Trajectory Similarity Search with Multi-level Semantics

Jianbing Zheng[1], Shuai Wang[1], Cheqing Jin[1](✉), Ming Gao[1], Aoying Zhou[1], and Liang Ni[2]

[1] East China Normal University, Shanghai, China
{jbzheng,51215903058}@stu.ecnu.edu.cn,
{cqjin,mgao,ayzhou}@dase.ecnu.edu.cn
[2] Nanjing University of Science and Technology, Nanjing, China

Abstract. With the widespread popularity of intelligent mobile devices, massive trajectory data have been captured by mobile devices. Although trajectory similarity search has been studied for a long time, most existing work merely considers spatial and temporal features or single-level semantic features, thus insufficient to support complex scenarios. Firstly, we define multi-level semantics trajectory to support flexible queries for more scenarios. Secondly, we present a new "spatial + multi-level semantic" trajectory similarity query, and then propose a framework to find k most similar ones from a trajectory database efficiently. Finally, to hasten query processing, we build a multi-layer inverted index for trajectories, design 4 light-weight pruning rules, and propose an adaptive updating method. The thorough experimental results show that our approach works efficiently in extensive and flexible scenarios.

Keywords: Multi-level semantics · Inverted index · Trajectory similar query

1 Introduction

With the popularization and development of ubiquitous computing and positioning technology, users' location information is frequently collected from daily life by mobile intelligent devices. The massive trajectory data not only reflect a person's daily behavior, but also indicate the activity pattern of a user group or even the whole city. Therefore, trajectory analysis is involved in many fields, such as precision marketing, statistical analysis and policy making.

Similarity search is a typical issue in trajectory data management. Given a query trajectory, the goal is to find one or more trajectories close to it. Currently, most of the existing works that mainly focus on spatial and temporal features of trajectory, are lack of effective utilization of trajectories' semantics. In fact, with the popularity of intelligent devices and applications, more and more data beyond space and time are collected and converted into semantic information. Trajectory similarity search that integrates semantics is more meaningful in real life, which therefore supports more scenarios [1]. Moreover, since the semantic attributes of trajectory points are hierarchical, it can be defined as a specific attribute name, or an attribute level above the specific attribute name. In most cases, although semantic attributes may not be identical at the current level, they may be similar at higher levels. Therefore, comparing the hierarchical semantic similarity between trajectories has wide application, such as transportation modes and trip purpose. Take Fig. 1 as an example (the X-axis and Y-axis are longitude and latitude respectively), although the four trajectories are far away and not similar for spatial dimension, they are similar in transportation mode and trip purpose for semantic dimension.

Transportation Modes Classification and Planning. Multi-level semantic trajectories can be queried and applied in a variety of ways, which cannot be realized simultaneously by single-level semantics. For example, the four trajectories represent four students going to school, where Tr_1 and Tr_3 take the bus, Tr_4 takes the subway, and Tr_2 drives to school. If someone wants to choose a favorite mode from all possible public transportation modes to school, the high-level semantics, "Public Transportation", can be used to query all the trajectories in transportation mode. Moreover, integrating multi-level semantics, the trajectories with the same destination in the same region can be found as an alternate path. For example, Tr_1 and Tr_4 are close in space, and both public transport. When there are few buses, students on Tr_1 (bus) can choose Tr_4 (subway) to go to school in the same area through the query of high-level semantics, "Public Transportation", and space.

Trip Purpose Classification and Planning. With the combination of low and high level semantic tags, more personalized services can be provided. For example, when travelling to Beijing, we not only want to visit the Palace Museum and the Great Wall (low-level semantic tag), but also taste Beijing's special food and theater (high-level semantic tag), which cannot be achieved simultaneously with single-level semantics. In this example, with the high-level semantic tag ("Special Food"), multiple trajectories passing through different specialty restaurants will be found for people to choose.

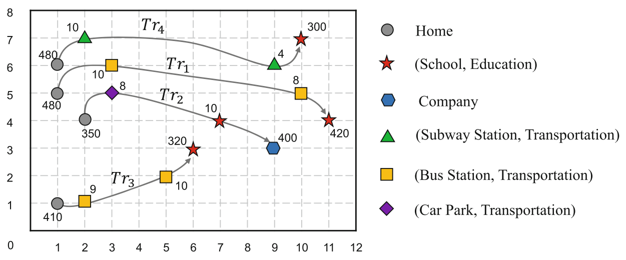


Fig. 1. An example of multi-level semantic trajectories.

However, there are several challenges in trajectory data analysis. Since massive trajectory data are collected by mobile devices every day, it is challenging to convert discrete trajectory points into stay points and then attach accurate semantic tags. In addition, previous studies only considered single-level semantics, and Jaccard Index was usually used to evaluate semantic similarity of trajectories. However, since Jaccard Index no longer works for multi-level semantics, how to define an appropriate similarity evaluation method is also challenging. Furthermore, processing large-scale trajectory data raises another challenge. The conventional methods, which compare trajectories one by one, is quite expensive. Therefore, it is critical to design efficient pruning methods and feasible indexes for optimization.

In this paper, we propose a similar trajectory search framework integrating hierarchical semantic features. We first mine stay points from raw trajectories, then choose appropriate multi-level semantic tags for each stay point to generate semantic trajectories. To evaluate the difference between a pair of trajectories, we define an efficient trajectory distance based on time, space and semantics. Meanwhile, we introduce spatial and semantic indexes, and propose several lightweight pruning rules to optimize query processing. To sum up, we make the following contributions in this paper:

- We introduce a new way to evaluate trajectory distance based on spatio-temporal and hierarchical semantic features.
- We present an efficient query processing method, which relies on indexes and four pruning rules.
- We verify the performance of our proposed method on two different real datasets.

The rest of our paper is organized as follows. Section 2 reviews related work in recent years. Section 3 defines some important concepts. Section 4 introduces our novel framework in detail. Section 5 verifies the performance of our proposed methods through experiments. Section 6 summarizes the paper briefly.

2 Related Work

Trajectory similarity research, a foundational task of trajectory data management, plays an important role in many fields, such as traffic management, urban planning and intelligent recommendation.

Spatio-Temporal Similarity of Trajectory. Early studies mainly focused on the measurement of spatio-temporal similarity of trajectory [2,3]. To deal with the case that trajectory points are not aligned, Ta et al. proposed a similarity measurement method of bidirectional mapping, and then generated a feature code for each trajectory for rapid pruning [4]. In [5], Shang et al. combined time and space linearly to calculate trajectory similarity, and proposed a two-stage algorithm to support parallel retrieval of spatial and temporal dimensions, which improves query efficiency.

Semantic Similarity of Trajectory. With the emergence of the blending of location and text, since more and more trajectories carry text information, more researches focus on trajectory similarity search with semantics [6,7]. In [8], Zheng et al. proposed fuzzy keyword query for semantic trajectory. Given a set of query keywords, by calculating the pairwise semantic edit distance, top-k trajectories with the smallest distance are returned.

Spatial and Semantics Similarity of Trajectory. He work that combines space and semantics has also been studied in recent years [9,10]. [11] proposed a top-k spatial keyword activity trajectory query method, aiming to find a set of trajectories that both geographically close and semantically meet the query requirement. Compared with the existing fuzzy keyword query, this query can find more similar trajectories. In [12], Chen et al. proposed a divide-and-conquer algorithm to deduce the boundary of spatial similarity and textual similarity between two trajectories, which realizes trajectory pruning without calculating the exact value of trajectory similarity and improves the query efficiency.

Although the trajectory similarity search based on spatio-temporal dimensions is intuitive, how to reflect the semantic attribute is challenging. Also, it is relatively unitary to compute trajectory similarity only from semantic dimension. Therefore, the combination of space and semantics is more effective and suitable for more scenarios. But most existing researches only consider single level semantics, without in-depth consideration of the hierarchy of semantics. And the existing method of calculating trajectory distance based on single-level semantics is not suitable for multi-level semantics. Consequently, we define a new way to compute the distance between trajectories, based on which an efficient method by integrating several pruning rules is given.

Index of Trajectory. In addition, to improve query efficiency, it is necessary to build indexes in both spatial and semantic dimensions. [13] described the trajectory representation and storage, and lists the indexing methods for spatial text trajectory data: quadtree, R-tree, grid index and Z-order curve. Space-efficient index representations and processing frameworks are crucial for trajectory data, and the majority of trajectory search solutions [14] rely on an R-tree [15], which store all points from the raw trajectories. Since trajectory datasets such as T-drive [16] often contain millions of points, the R-tree must manage an enormous number of maximum bounding rectangles (MBR), which have prohibitive memory cost in practice. Simpler Grid-index solutions are sometimes more appropriate in such scenarios [17]. Therefore, we combine grid index and inverted index to establish spatial index and semantic index.

3 Problem Definition

In this section, we define the query formally, along with some core concepts.

Definition 1. Trajectory (Tr). A trajectory Tr is a series of n points ordered by time, $Tr = (p_1, p_2, ..., p_n)$, where $p_i = (t_i, loc_i)$ means that the moving object is located at loc_i at time t_i.

Definition 2. Stay Point (SP). A stay point $s = (loc, len, c)$ represents that a moving object has stayed at loc for len time, and c is a label to represent the corresponding semantics.

As aforementioned in Sect. 1, each stay point may have concrete semantics, e.g., restaurant, cinema and university. The semantics can be structured hierarchically. For example, through the POI information of the map and manual correction, we define the multi-level semantics of Tsinghua University as (educational institution, school, university, Tsinghua University). Furthermore, some special multi-level semantics may be customized for different persons, such as home and working area, as illustrated in Fig. 2.

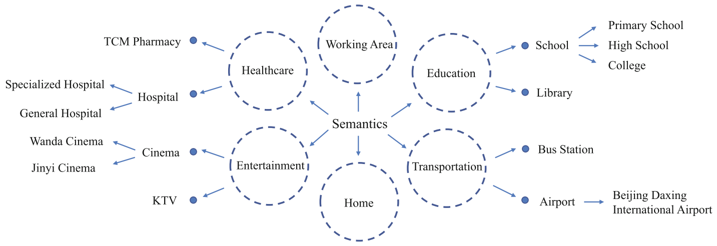


Fig. 2. An example of semantics.

Definition 3. Semantic Trajectory (ST). A semantic trajectory ST is a series of chronological ordered stay points of a moving object, $ST = (s_1, s_2, ..., s_m)$.

To compute the distance between a pair of semantic trajectories A and B (Spatial Semantic distance, S^2D), we integrate both spatial distance (SpD) and semantic distance (SeD) at the same time, as shown below.

$$S^2D(A,B) = \beta \cdot SpD(A,B) + (1-\beta) \cdot SeD(A,B) \tag{1}$$

where $\beta \in [0,1]$ leverages the importance of each feature. We set β a greater value if more attention is paid to space, and vice versa. The spatial distance, $SpD(A,B)$, is computed as normalized Euclidean distance.

$$SpD(A,B) = \begin{cases} \sum_{i=1}^{|A|} \|A.loc_i, B.loc_i\|, & |A| \le |B| \\ MAX\_DIST, & |A| > |B| \end{cases} \tag{2}$$

$$\|A.loc_i, B.loc_i\| = \frac{dis(A.loc_i, B.loc_i) - minDis}{maxDis - minDis + 1} \tag{3}$$

where $dis(A.loc_i, B.loc_i)$ represents the Euclidean distance between point $A.loc_i$ and $B.loc_i$, $maxDis$ and $minDis$ are the maximum and minimum values of the distance between track A and B.

According to Eq. (2), if the length of B is smaller than that of A, we simply return MAX_DIST, a specific BIG value. Otherwise, we compare the first $|A|$ pairs of points. Note that $SpD(A, B) = SpD(B, A)$ only if $|A| = |B|$. If $|A| \neq |B|$, $SpD(A, B) \neq SpD(B, A)$. In comparison, the semantic distance, SeD, is a bit more complex, since the two labels may be identical at any level, or totally different at all levels. Hence, the semantic difference between two labels, $sd(c_1, c_2)$, and the overall semantic distance between two semantic trajectories, $SeD(A, B)$, are defined below.

$$sd(c_1, c_2) = \begin{cases} 0, & c_1 = c_2 \\ 1 - \alpha^g, & \text{identical in the upper } g\text{-th level} \\ 1, & \text{different at all levels} \end{cases} \tag{4}$$

$$SeD(A, B) = \begin{cases} \sum_{i=1}^{|A|} sd(A.c_i, B.c_i), & |A| \leq |B| \\ MAX\_DIST, & |A| > |B| \end{cases} \tag{5}$$

where $\alpha \in (0, 1)$ in Eq. (4) reflects the decay rate. $\forall c_1, c_2, sd(c_1, c_2) \in [0, 1]$. We use α to control the decay rate of semantic distance if two labels are identical at the g-th upper level, i.e., a smaller α value means higher decay rate.

Definition 4. Spatial Semantic Similar query (*S^2Sim query*). Given a set of semantic trajectories $\Phi = \{ST_1, ST_2, ..., ST_g\}$, a query trajectory q, an adjustment coefficient β and an integer k, $S^2Sim(\Phi, q, \beta, k)$ query returns a subset Φ', such that (1) $|\Phi'| = k$, and (2) for any $ST_i \in \Phi'$ and $ST_j \in \Phi \setminus \Phi'$, $S^2D(ST_i, q) \leq S^2D(ST_j, q)$.

Take Fig. 1 as an example, suppose ST_1 is a query trajectory, $k = 1$ and $\alpha = 0.7$. Table 1 illustrates S^2D values between ST_1 and others in different β. If we consider space and semantics equally ($\beta = 0.5$), we return ST_4 because $S^2D(ST_1, ST_4) = 0.5 \times 8.74 + 0.5 \times 0.6 = 4.67$ is the minimal one. Similarly, if we only focus on semantics ($\beta = 0$) or space ($\beta = 1$) respectively, the query result is ST_3 or ST_2 accordingly.

Table 1. S^2D between ST_1 and other three trajectories in different β.

ST	SpD	SeD	S^2D			$wSpD$	$wSeD$	S^2D		
			$\beta = 0$	$\beta = 0.5$	$\beta = 1$			$\beta = 0$	$\beta = 0.5$	$\beta = 1$
ST_2	7.81	2.3	2.3	5.06	**7.81**	1.857	0.536	0.536	1.197	1.857
ST_3	20.02	0	**0**	10.01	20.02	4.504	0	**0**	2.252	4.504
ST_4	8.74	0.6	0.6	**4.67**	8.74	1.833	0.005	0.005	**0.919**	**1.833**

$$wSpD(A,B) = \begin{cases} \sum_{i=1}^{|A|} w_i \cdot \|A.loc_i, B.loc_i\|, & |A| \le |B| \\ MAX\_DIST, & |A| > |B| \end{cases} \tag{6}$$

$$wSeD(A,B) = \begin{cases} \sum_{i=1}^{|A|} w_i \cdot sd(A.c_i, B.c_i), & |A| \le |B| \\ MAX\_DIST, & |A| > |B| \end{cases} \tag{7}$$

As a moving object may stay at a stay point for different time length, the stay time length also acts as an important factor, which results in a weighted version. Equation (6) and (7) depict the weighted version of distance, where the i-th weight is $w_i = \frac{\min(A.len_i, B.len_i)}{\sum_{j=1}^{\min(|A|,|B|)} \min(A.len_j, B.len_j)}$.

The right part of Table 1 shows the weighted distance, where the stay time length of each stay point comes from in Fig. 1. If we consider space and semantics equally ($\beta = 0.5$), we return ST_4 because $S^2D(ST_1, ST_4) = 0.5 \times 1.833 + 0.5 \times 0.005 = 0.919$ is the minimal one. Similarly, if we focus on semantics ($\beta = 0$) or space ($\beta = 1$) respectively, the query result is ST_3 or ST_4 accordingly.

4 Framework

In this section, we detail our framework, shown in Fig. 3. Our framework contains three parts: semantic trajectory generation, index construction and query processing. In the first part, stay points are generated and then tagged with concrete semantics. In index construction part, we build spatial index and semantic index to quickly retrieve trajectories. Furthermore, we propose several pruning rules to quickly update the candidate set. Finally, we return top-k similar trajectories.

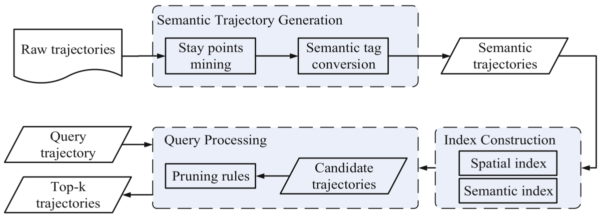


Fig. 3. Processing framework.

4.1 Semantic Trajectory Generation

To generate semantic trajectory, we mine stay points from raw trajectories, and then set appropriate labels. Algorithm 1 describes the steps to generate semantic trajectory. The first point is initialized as the left endpoint of a segment (line 1). If a segment $(p_l, ..., p_{r-1})$ satisfies the conditions $\|loc_r, loc_l\| > \delta_{dis}$ and $t_r - t_l \geq \delta_t$, it means $\|loc_{r-1}, loc_l\| \leq \delta_{dis}$ and we treat the mean of this segment as a stay point and put it into st (lines 3–6). Otherwise, if $t_r - t_l < \delta_t$, we ignore this segment and update the left endpoint (line 7). After traversing all points, if $l < |Tr|$, it means $\|loc_{|Tr|}, loc_l\| \leq \delta_{dis}$. And if $t_{|Tr|} - t_l \geq \delta_t$, we also treat the mean of the segment $(p_l, ..., p_{|Tr|})$ as a stay point and put it into st (lines 10–12).

After getting stay points, we select the nearest point of interest (POI) as semantic label. Finally, the raw trajectory Tr is transformed into a semantic trajectory st.

Algorithm 1. GenST

```
Input: $Tr$: raw trajectory, $\delta_{dis}$: distance threshold, $\delta_t$: temporal threshold;
Output: $st$: semantic trajectory;
 1: $l \leftarrow 1$, $st \leftarrow \emptyset$;
 2: for $r = 2$ to $|Tr|$ do
 3:     if $\|loc_r, loc_l\| > \delta_{dis}$ then
 4:         if $t_r - t_l \geq \delta_t$ then
 5:             $s.loc \leftarrow$ the center of points $p_l, ..., p_{r-1}$, $s.len \leftarrow t_r - t_l$, $st.add(s)$;
 6:         end if
 7:         $l \leftarrow r$;
 8:     end if
 9: end for
10: if $l < |Tr|$ and $t_{|Tr|} - t_l \geq \delta_t$ then
11:     $s.loc \leftarrow$ the center of points $p_l, ..., p_{|Tr|}$, $s.len \leftarrow t_{|Tr|} - t_l$, $st.add(s)$;
12: end if
13: return $st$;
```

4.2 Query Processing

The most straightforward solution to find similar trajectories is to scan the whole trajectory database, and compute the distance to the query trajectory one by one, which is, however, inefficient for large-scale dataset. To improve query efficiency, it is necessary to build indexes in both spatial and semantic dimensions, and propose efficient light-weight pruning rules, which is the basic idea of this paper.

Index Construction. We have combined grid index and inverted index to establish spatial index and semantic index. According to the proposed trajectory similarity formula, we calculate the spatial and semantic distance point by point. Therefore, we put the i-th point of each trajectory together to construct hierarchical spatial and semantic index. The final number of levels is determined by the longest trajectory in the dataset.

For spatial dimension, we use grid index to handle trajectories, where the whole space is divided into multiple basic cells with same size, and each trajectory is mapped to the corresponding cell. Let a and b denote two arbitrary points in the space, function $mindis(cell(a), cell(b))$ computes the minimal distance between two cells containing a and b respectively. Note that $mindis(cell(a), cell(b)) = 0$ if they are at the neighbor cell. For semantic dimension, we build an inverted index for semantic information. And each node is represented as (c, trs), where c refers to the label and trs is a set of trajectories. Since the label is structured hierarchically, function $upper(c_i, g)$ gets the label in the g-th upper level. For example, $upper$ (bus station, 1) = transportation, since transportation is the highest level of bus station, as shown in Fig. 2.

Index Maintenance. When a new trajectory is added to the database, it is unnecessary to rebuild the existing index, but only to update it incrementally. Spatial index and semantic index have the same essential structure, which are both multilevel inverted index.

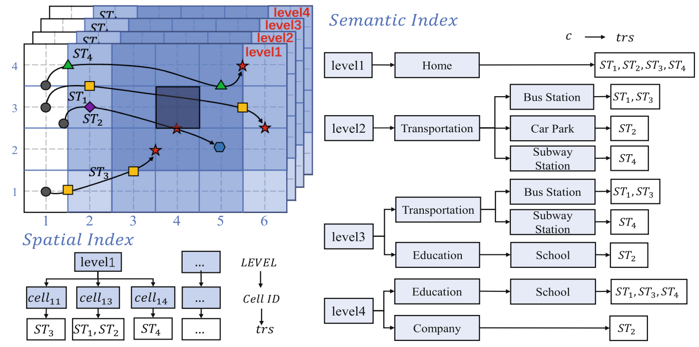


Fig. 4. An example of spatial and semantic indexes.

Figure 4 illustrates the spatial and semantic indexes for the trajectories in Fig. 1. The maximum number of four trajectories is 4, so the spatial index and semantic index have 4 levels respectively. The left part is spatial index, where the space is divided into 24 cells, each with a size of 2 km × 2 km. The minimal distances between $cell_{43}$ and $cell_{42}$, $cell_{23}$ and $cell_{21}$ (calling $mindis$ function) are 0, 2 and $2\sqrt{2}$, respectively. The right part is semantic index, and each level is a tree-based structure, which satisfies the hierarchy of semantics. One trajectory may fall on many nodes, and one node will contain many trajectories. So, we can easily get trajectories using one semantic label. At level 2, ST_1 and ST_3 will be quickly extracted via "Bus Station".

Pruning Rules. Besides indexing, it is necessary to devise light-weight pruning rules to hasten query processing. In other words, given partial information, we are capable of judging whether a trajectory is a candidate or not. The main idea is that the distance between two trajectories can be accumulated point by point. Consequently, if the partial distance based on one or several stay point(s) is evaluated exceeding a given threshold, the whole trajectory can be filtered safely.

Pruning Rule 1 (Spatial Pruning). Given an upper bound τ, an adjustment parameter β, a query semantic trajectory q, and an arbitrary semantic trajectory r, $S^2D(q,r) > \tau$ if there exist positions $i_1, i_2, ..., i_d$, such that

$$\sum_{j=1}^{d} mindis(cell(q.loc_{i_j}), cell(r.loc_{i_j})) > \frac{\tau}{\beta} \tag{8}$$

Proof. According to Eq. (2), we have:

$$SpD(q,r) \geq \sum_{j=1}^{d} \parallel q.loc_{i_j}, r.loc_{i_j} \parallel \geq \sum_{j=1}^{d} mindis(cell(q.loc_{i_j}), cell(r.loc_{i_j})) > \frac{\tau}{\beta}$$

Integrating with Eq. (1), we have $S^2D(q,r) \geq \beta \cdot SpD(q,r) > \tau$. □

Pruning Rule 2 (Semantic Pruning). Given an upper bound τ, an adjustment parameter β, a query semantic trajectory q, and an arbitrary semantic trajectory r, $S^2D(q,r) > \tau$ if there exist positions $i_1, i_2, ..., i_d$, such that

$$\sum_{j=1}^{d} sd(q.c_{i_j}, r.c_{i_j}) > \frac{\tau}{1-\beta} \tag{9}$$

Proof. According to Eq. (5), we have:

$$SeD(q,r) \geq \sum_{j=1}^{d} sd(q.c_{i_j}, r.c_{i_j}) > \frac{\tau}{1-\beta}$$

Integrating with Eq. (1), we have $S^2D(q,r) \geq (1-\beta) \cdot SeD(q,r) > \tau$. □

Framework. We propose a query framework which is **T**op-**k** similar trajectories query based on **I**ndexes and **P**runing (TKIP). In Algorithm 2, we get spatial and semantic candidate sets according to Algorithm 3 and 4 at first, then we get top-k similar trajectories from the candidate set. Among them, the selection of spatial candidate set needs to calculate the nearest spatial distance. In order to improve the calculation efficiency, we generate cell distance coordinate pair set *DcSet* in advance and arrange them in ascending order of distance. Specifically, the *DcSet* contains two elements, the cell distance value *dist*, and its corresponding

Algorithm 2. TKIP

```
Input: $ISP$: spatial index, $ISE$: semantic index, $q$: query trajectory, $DcSet$, $\beta$, $k$;
Output: $\Psi$: top-$k$ similar trajectories;
1: $\tau \leftarrow \infty$, $\Psi \leftarrow \emptyset$, $g \leftarrow 0$, $step \leftarrow 0$, $\sigma \leftarrow 0$, $lb_1 \leftarrow 0$, $lb_2 \leftarrow 0$;
2: while $lb_1 \cdot \beta + lb_2 \cdot (1-\beta) \leq \tau$ do
3:     if $\beta$=1 then
4:         $S \leftarrow GetSpCandidate(q, ISP, step, DcSet)$, $step \leftarrow step + 1$;
5:     else if $\beta$=0 then
6:         $S \leftarrow GetSeCandidate(q, ISE, g)$, $g \leftarrow g + 1$;
7:     else
8:         $S_1 \leftarrow GetSpCandidate(q, ISP, step, DcSet)$;
9:         $S_2 \leftarrow GetSeCandidate(q, ISE, g)$;
10:        $S \leftarrow S_1 \cup S_2$, $step \leftarrow step + 1$, $\sigma \leftarrow \sigma + (1-\beta)$, $g \leftarrow g + \lfloor \sigma \rfloor$;
11:    end if
12:    $S \leftarrow$ remove trajectories in S which length less than $|q|$;
13:    if $S \neq \emptyset$ then
14:        $\Psi \leftarrow k$ nearest trajectories in $\Psi \cup S$, $\tau \leftarrow \max(\{S^2D(q,r) | r \in \Psi\})$;
15:    end if
16:    $lb_1 \leftarrow DcSet_{step}.dis$, $lb_2 \leftarrow$semantic distance of the g-th level;
17: end while
18: return $\Psi$;
```

coordinate pair *CoorSet*. And the *CoorSet* contains the cell distance between X-axis and Y-axis formed as $(\Delta x, \Delta y)$.

Algorithm 2 details the steps to process TKIP. We first initialize upper bound τ, candidate set Ψ, the level of multi-level semantics g, current distance *step*, adjusting parameter σ, spatial lower bound lb_1 and semantic lower bound lb_2 (line 1). For each iteration, we use the pruning rules to determine whether to terminate the algorithm (line 2). If not, we should determine the value of parameter β at first. If $\beta = 0$ or 1, then we only need to get a candidate set of one dimension, spatial or semantic, and update the corresponding parameters *step* or g (lines 3–6). Otherwise, both the spatial and semantic candidate sets are generated to get the final candidate set, and the parameter *step*, σ and g are updated accordingly (lines 7–10). Furthermore, we remove the trajectories in S whose trajectory length is less than $|q|$ (line 12). If candidate set S is not empty, we keep the k nearest trajectories in Ψ and update threshold τ (lines 13–15). At the end of each iteration, we update the lower bounds lb_1 and lb_2 (line 16). Finally, Ψ only contains the top k similar trajectories.

Algorithm 3 details the steps to get the spatial candidate set of similar trajectories. We first initialize spatial candidate set S_1 (line 1). For i from 1 to $|q|$, we get the cell id of the i-th point in q and all the cell id in the i-th level of ISP. According to the *DcSet*, we get the set of coordinate pairs corresponding to the current step, generate the corresponding candidate cells based on the query cell and put them in set DS. Next we take the intersection of cells which unmarked in *DcSet* and *HCI* as the candidate set and put them in S_1 (lines 2–7). Finally, we mark the trajectories in S_1 in the index to prevent re-access (line 8). The

Algorithm 3. GetSpCandidate

```
Input: $q$: query trajectory, $ISP$: spatial index, $step$: current distance step, $DcSet$;
Output: $S_1$: spatial trajectory candidate set;
1: $S_1 \leftarrow \emptyset$;
2: for $i$ from 1 to $|q|$ do
3:     $h \leftarrow cell(q.loc_i)$ , $HCI \leftarrow$ get all the cell id in the i-th level of $ISP$;
4:     $A \leftarrow DcSet_{step}.CoorSet$, $DS \leftarrow \{h.x + A.\Delta x, h.y + A.\Delta y\}$;
5:     $mTr \leftarrow$ unmarked trajectories in $DS \cap HCI$;
6:     $S_1 \leftarrow S_1 \cup mTr$;
7: end for
8: Mark the trajectories in $S_1$ in the index to prevent re-access;
9: return $S_1$;
```

Algorithm 4. GetSeCandidate

```
Input: $q$: query trajectory, $ISE$: semantic index, $g$;
Output: $S_2$: semantic trajectory candidate set;
1: $S_2 \leftarrow \emptyset$;
2: for $i$ from 1 to $|q|$ do
3:     $r \leftarrow upper(q.c_i, g)$ , $R \leftarrow$ Get $i$-th level of $ISE$;
4:     $Tr \leftarrow$ trajectories in the $g$-th level from $R$ which contians $r$ and unmarked;
5:     $S_2 \leftarrow S_2 \cup Tr$;
6: end for
7: Mark the trajectories in $S_2$ in the index to prevent re-access;
8: return $S_2$;
```

semantic candidate set is obtained in a manner similar to the spatial candidate set, as detailed in Algorithm 4.

When updating the candidate set, we use a priority queue to maintain the candidate set and its complexity is $O(\log k)$. Therefore, the time complexity is $O(L \cdot |S| \cdot (|q| + \log k))$, where $O(L)$ is the number of iterations and $O(|S| \cdot (|q| + \log k))$ is the cost of updating the candidate set.

4.3 Weighted Query Processing

Our proposed framework (Algorithm 2) still works for weighted query version (wTKIP), except the pruning rules should be modified accordingly. The complexity of this weighted version (wTKIP) is the same as the original version (TKIP). To simplify the formula and proof, let $\lambda_i = \frac{\min(q.len_i, r.len_i)}{q.length}$, where $q.length = \sum_{j=1}^{|q|} q.len_j$ means the total stay time length of q. According to Eq. (6) and (7), the weight w_i is computed as $\frac{\min(q.len_i, r.len_i)}{\sum_{j=1}^{\min(|q|,|r|)} \min(q.len_j, r.len_j)}$, where $\sum_{j=1}^{\min(|q|,|r|)} \min(q.len_j, r.len_j)$ is within $[\min(q.len_i, r.len_i), q.length]$. Thus, w_i is within $[\lambda_i, 1]$.

Pruning Rule 3 (Weighted Spatial Pruning). Given an upper bound τ, an adjustment parameter β, a query semantic trajectory q, and an arbitrary semantic trajectory r, $S^2D(q,r) > \tau$ if there exist positions $i_1, ..., i_d$, such that

$$\sum_{j=1}^{d} \lambda_{i_j} \cdot mindis(cell(q.loc_{i_j}), cell(r.loc_{i_j})) > \frac{\tau}{\beta} \tag{10}$$

Proof. According to Eq. (6), we have:

$$\begin{aligned} SpD(q,r) &\geq \sum_{j=1}^{d} w_{i_j} \cdot \parallel q.loc_{i_j}, r.loc_{i_j} \parallel \\ &\geq \sum_{j=1}^{d} w_{i_j} \cdot mindis(cell(q.loc_{i_j}), cell(r.loc_{i_j})) \\ &\geq \sum_{j=1}^{d} \lambda_{i_j} \cdot mindis(cell(q.loc_{i_j}), cell(r.loc_{i_j})) > \frac{\tau}{\beta} \end{aligned}$$

Integrating with Eq. (1), we have $S^2D(q,r) \geq \beta \cdot SpD(q,r) > \tau$. □

Pruning Rule 4 (Weighted Semantic Pruning). Given an upper bound τ, an adjustment parameter β, a query semantic trajectory q, and an arbitrary semantic trajectory r, $S^2D(q,r) > \tau$ if there exist positions $i_1, ..., i_d$, such that

$$\sum_{j=1}^{d} \lambda_{i_j} \cdot sd(q.c_{i_j}, r.c_{i_j}) > \frac{\tau}{1-\beta} \tag{11}$$

Proof. According to Eq. (7), we have:

$$SeD(q,r) \geq \sum_{j=1}^{d} w_{i_j} \cdot sd(q.c_{i_j}, r.c_{i_j}) \geq \sum_{j=1}^{d} \lambda_{i_j} \cdot sd(q.c_{i_j}, r.c_{i_j}) > \frac{\tau}{1-\beta}$$

Integrating with Eq. (1), we have $S^2D(q,r) \geq (1-\beta) \cdot SeD(q,r) > \tau$. □

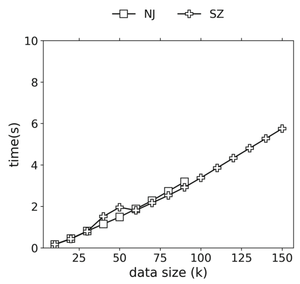
5 Experiments

In this section, we evaluate our proposed framework upon two real datasets. One dataset named NJ is the base-station access logs of 5,000 users in Nanjing from Oct. 10 to Oct. 31, 2020. Another dataset named SZ is the base-station access logs of 8,000 users in Suzhou from Mar. 10 to Mar. 31, 2021. The sizes of the two datasets are about 3.4 GB and 6.17 GB respectively. In addition, we use a real file with 982,200 POIs containing real location and the corresponding multi-level semantic information. All codes were written in C++. The experiments run on a PC with Windows 10 Pro, Intel(R) Core(TM) i7-8550U CPU @1.80 GHz, and 16 GB of RAM.

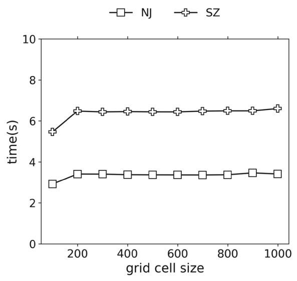
We transform the raw trajectories into semantic trajectories by setting $\delta_{dis} = 1\,\text{km}$ and $\delta_t = 900\,\text{s}$, then we can get 90,423 and 154,096 semantic trajectories respectively. As mentioned above, since no existing work considers spatial dimension and hierarchical semantics at the same time, we use LINEAR and weighted LINEAR (wLINEAR) as the baseline methods, which computes the distance between the query trajectory and other trajectories one by one. LINEAR directly traverses the trajectory set, calculates the distance using the trajectory distance formula S^2D defined in this paper, and returns the closest one. The difference between wLINEAR and LINEAR is that wLINEAR calculates the distance using the weighted trajectory distance formula S^2D.

We build spatial index and semantic index in this paper. Figure 5 illustrates the construction time and memory consumption of index under different factors and datasets.

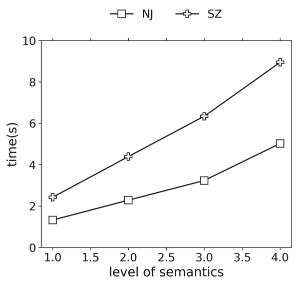
As shown in Fig. 5(a) and (d), with the increment of dataset size, the index construction time and memory consumption also rise accordingly. Figure 5(b) and (e) show that with the increase of grid cell size, the construction time of spatial index changes little, and the memory consumption tends to decrease due to the reduction of the number of cells that need to be placed in memory.



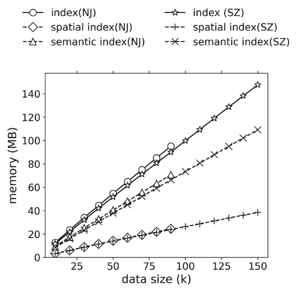
(a) Time cost of index construction



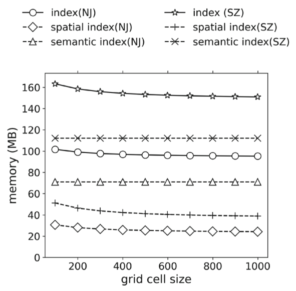
(b) Time cost upon different grid size



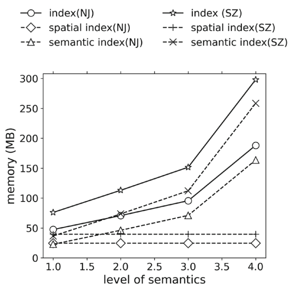
(c) Time cost upon different levels of semantics



(d) Space consumption of index construction



(e) Space consumption upon different grid size



(f) Space consumption upon different levels of semantics

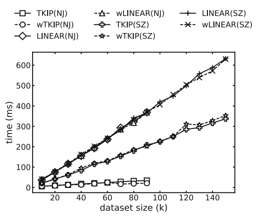
Fig. 5. Index construction time and memory on different factors.

Figure 5(c) and (f) illustrate that with the increment of semantic level, the construction time and memory consumption of semantic index will increase significantly. Moreover, the level of semantics will lead to different depth and number of nodes in semantic index. In conclusion, through the analysis of the above experiments, it is found that the index construction time will not increase significantly with the increase of semantic level or the number of grid cells, and basically remains unchanged. This shows that our index structure is reasonable and efficient.

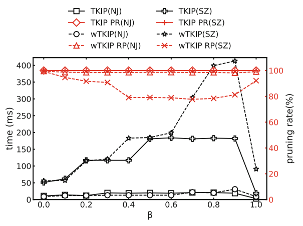
5.1 Performance Comparison

We report the performance of our framework by comparing with the baseline methods. We randomly select one query trajectory, and use our algorithms and baseline methods to process queries in different dataset size and record these query times.

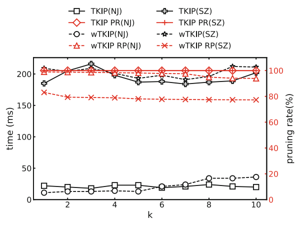
Figure 6(a) reports the performance comparison. We observe that TKIP and wTKIP are much faster than LINEAR and wLINEAR in different cases.



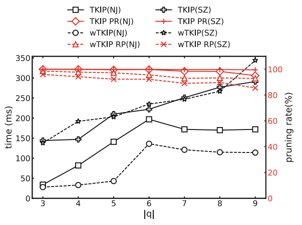
(a) dataset size



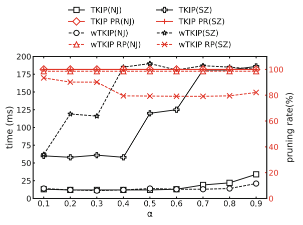
(b) β



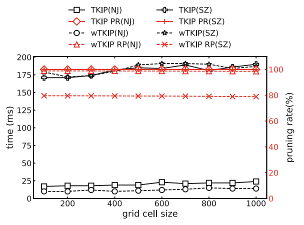
(c) k



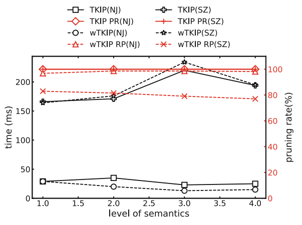
(d) $|q|$



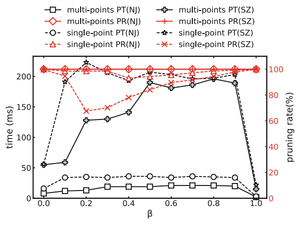
(e) α



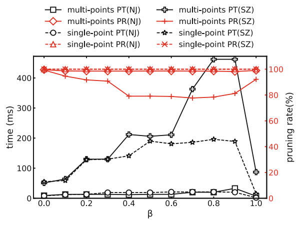
(f) grid cell size



(g) level of semantics



(h) TKIP



(i) wTKIP

Fig. 6. Query efficiency on different factors.

However, due to different data structures of SZ and NJ, the results presented by the two experimental datasets are not completely consistent. There is also slight difference in query performance between different query methods. When the NJ dataset size is 90k, TKIP and wTKIP only take about less than 50 ms, while LINEAR and wLINEAR take about 500 ms.

5.2 Parameter Analysis

Then, we investigate the impact of the parameters. We randomly select one query trajectory to process queries under different parameters, and record query time and pruning rate (PR).

Figure 6(b) shows that the influence of β on query efficiency and pruning rate under different experimental data. Query efficiency is highest when β=0 or 1, because only one distance, spatial or semantic distance, is concerned at this time, thus greatly reducing the query time. In other cases, the higher the β value, the lower the query efficiency. This is because spatial features will receive more attention when β increases, and the query efficiency of spatial indexes is lower than that of semantic indexes, because the pruning rules of semantic indexes are simpler than those of spatial indexes. Figure 6(c) shows that with the increase of k, the query efficiency decreases. The reason is that the greater the k value is, the greater the upper bound τ will be, which may reduce the pruning efficiency. Figure 6(d) shows that with the increase of $|q|$, the query efficiency may be reduced. Because the calculation time of the trajectory distance is a factor that affects the efficiency, which is proportional to the length of the trajectory.

In summary, we find that parameter settings do affect query time and pruning rate. Moreover, the query time is not only determined by the pruning rate, but also other factors, such as the time consumed by merging trajectories.

5.3 Analysis of Indexes and Pruning Rules

We first analyze the impact of different α and index structures on query efficiency, and then compare pruning rules, finally verify the effectiveness.

Figure 6(e) shows the query efficiency on different α. The result shows that a smaller α may reduce the query time. And the weighted query method is more deeply affected by α than the unweighted query method. Moreover, when $\alpha = 0.4$, there will be an inflection point when query time increases. In addition, Fig. 6(f) and (g) illustrate the query efficiency under different grid cell size and semantic level, respectively. We find that with the increase of grid cell size, query efficiency decreases, but the effect is insignificant. Due to the different data distribution of the two experimental datasets, the efficiency of weighted query based on the experimental data of SZ is generally low, mainly due to the low pruning rate. Moreover, the level of semantics has little influence on the query efficiency, whether weighted or unweighted.

Our pruning rules determine whether the partial distance based on one or several stay point(s) exceeds a threshold. Single point pruning only considers one point, while multi-points pruning considers as many as points as possible.

To compare the efficiency of single point and multi-points pruning, we record the query time and pruning rate under two pruning rules. Figure 6(h) and (i) illustrate the comparison of TKIP and wTKIP respectively. Figure 6(h) shows the query efficiency of multi-point pruning is higher than single-point pruning under TKIP. However, under wTKIP, as shown in Fig. 6(i), the weight of query method has a great influence on pruning methods.

6 Conclusion

In this paper, we study similarity search, a typical issue in trajectory data management. Unlike most existing works that only focus on spatio-temporal and single-level semantic features, we integrate multi-level semantic features and propose a framework to find top-k similar trajectories efficiently. We first build spatial and semantic indexes to quickly retrieve trajectories. To improve query efficiency, we also propose several light-weight pruning rules to filter invalid trajectories and update the candidate set continuously and an adaptive updating method of candidate set based on β value. In practice, parameter β is used to adjust spatial and semantic attention, so as to adapt to more scenarios. The experiments based on real datasets show that our approach is not only efficient, but also suitable for flexible scenarios.

In future, we intend to improve our index structure, and propose more pruning rules to filter out more illegal trajectories in time. Furthermore, we also intend to analyze the scenarios with different β value.

Acknowledgments. This work is partially supported by National Science Foundation of China (U1911203, 61877018 and U1811264).

References

1. Tao, Y., Papadias, D.: The MV3R-tree: a spatio-temporal access method for timestamp and interval queries. In: VLDB, pp. 431–440 (2001)
2. Wang, S., Ferhatosmanoglu, H.: PPQ-trajectory: spatio-temporal quantization for qerying in large trajectory repositories. Proc. VLDB Endow. **14**(2), 215–227 (2020)
3. Shang, Z., Li, G., Bao, Z.: DITA: distributed in-memory trajectory analytics. In: Proceedings of the 2018 International Conference on Management of Data, pp. 725–740 (2018)
4. Ta, N., Li, G., Xie, Y., Li, C., Hao, S., Feng, J.: Signature-based trajectory similarity join. IEEE TKDE **29**(4), 870–883 (2017)
5. Shang, S., Chen, L., Wei, Z., Jensen, C.S., Zheng, K., Kalnis, P.: Parallel trajectory similarity joins in spatial networks. VLDB J. **27**(3), 395–420 (2018)
6. Dai, Y., Shao, J., Wei, C., Zhang, D., Shen, H.T.: Personalized semantic trajectory privacy preservation through trajectory reconstruction. World Wide Web **21**(4), 875–914 (2018)
7. Kontarinis, A., Zeitouni, K., Marinica, C., Vodislav, D., Kotzinos, D.: Towards a semantic indoor trajectory model. In: EDBT/ICDT Workshops (2019)
8. Zheng, B., Yuan, N.J., Zheng, K., Xie, X., Sadiq, S., Zhou, X.: Approximate keyword search in semantic trajectory database. In: ICDE, pp. 975–986 (2015)

9. Zhao, K., Chen, L., Cong, G.: Topic exploration in spatio-temporal document collections. In: Proceedings of the 2016 International Conference on Management of Data, pp. 985–998 (2016)
10. Belesiotis, A., Skoutas, D., Efstathiades, C., Kaffes, V., Pfoser, D.: Spatio-textual user matching and clustering based on set similarity joins. VLDB J. **27**(3), 297–320 (2018). https://doi.org/10.1007/s00778-018-0498-5
11. Zheng, K., Zheng, B., Xu, J., Liu, G., Liu, A., Li, Z.: Popularity-aware spatial keyword search on activity trajectories. World Wide Web **20**(4), 749–773 (2016). https://doi.org/10.1007/s11280-016-0414-0
12. Chen, L., Shang, S., Jensen, C.S., Yao, B., Kalnis, P.: Parallel semantic trajectory similarity join. In: ICDE, pp. 997–1008 (2020)
13. Wang, S., Bao, Z., Shane Culpepper, J., Cong, G.: A survey on trajectory data management, analytics, and learning. ACM Comput. Surv. (CSUR) **54**(2), 1–36 (2021)
14. Chen, Z., Shen, H.T., Zhou, X., Zheng, Y., Xie, X.: Searching trajectories by locations-an efficiency study. In: SIGMOD, pp. 255–266 (2010)
15. Guttman, A.: R-trees: a dynamic index structure for spatial searching. ACM SIGMOD Rec. **14**(2), 45–47 (1984)
16. Yuan, J.: T-drive: driving directions based on taxi trajectories. In: GIS, pp. 99–108 (2010)
17. Zheng, K., Shang, S., Yuan, N.J., Yang, Y.: Towards efficient search for activity trajectories. In: ICD, pp. 230–241 (2013)