Distributed Semantic Trajectory Similarity Search

Shenghao Gong[†], Ziquan Fang[†], Lu Chen[†], Liu Liu[†], Yunjun Gao[†], Kai Zheng[♯], Gang Chen[†]
[†]*Zhejiang University, China*
[♯]*University of Electronic Science and Technology of China, China*
[†]*{gongshenghao, zqfang, luchen, liu2, gaoyj, cg}@zju.edu.cn* [♯]zhengkai@uestc.edu.cn

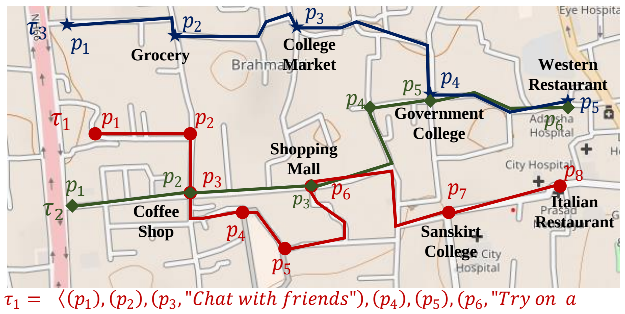
***Abstract*—The proliferation of location-based services enables the generation of massive semantic trajectories of moving objects. In this paper, we study the problem of semantic trajectory search, which is a fundamental functionality of trajectory analyses. The objective is to find "similar" trajectories of a query trajectory considering both spatial proximity and semantic similarity. Existing studies primarily employ keyword-based discrete semantic modeling, lacking the ability to capture broader semantic similarities like text descriptions. More importantly, these studies often develop search algorithms on a single machine, leading to constrained storage and processing capabilities.**

To this end, we propose an effective framework to perform distributed semantic trajectory similarity searches. Initially, we introduce a new method for representing semantic trajectories, utilizing geo-sequences, semantic sequences, and semantic embeddings. This approach enhances spatial-semantic awareness in trajectory modeling. Subsequently, we implement the query framework in Spark, employing a well-designed two-phase and computation-aware partitioning architecture. This is the first architecture that utilizes both semantic and spatial aspects of trajectories, as well as inter-trajectory distances, to guide the partitioning procedure. It allows for quick pruning of most partitions in the distributed cluster when processing a query trajectory. Within each local partition, we construct an ST-HNSW index to further accelerate queries. Our framework supports three popular trajectory distance measures – DTW, LCSS, and EDR. Extensive experiments using large-scale datasets demonstrate the significant superiority of our design, achieving efficiency improvements of up to 1 to 3 orders of magnitude compared to baselines and alternative designs.

I. Introduction

With the proliferation of GPS devices, a large number of trajectories of vehicles and individuals have been generated [58]. Retrieving similar trajectories of a query trajectory, called trajectory similarity search, is a fundamental functionality in trajectory databases [19], [34], [51]. Trajectory similarity search is useful in transportation and security fields, e.g., finding all vehicles that share similar routes to the query abnormal driving vehicle; finding close contacts to the patient's trajectory. Besides, it is a fundamental operator for many downstream analyses including clustering [59], pattern mining [14], and route planning [38]. Overall, the community has devoted efforts into spatial-aware trajectory search studies [21], [31], [33], [35], [60], [64], [65], [71], [74], e.g., searching for similar trajectories considering spatial perspective.

Recently, the explosion of social networks and online maps has augmented trajectories with semantic data [16], [77]. A semantic trajectory [44] is denoted by a sequence of locations where each location is associated with semantic label(s) like keywords and text descriptions. With semantic trajectories, we



$\tau_1 =$ ⟨$(p_1), (p_2), (p_3,$ *"Chat with friends"*$), (p_4), (p_5), (p_6,$ *"Try on a dress"*$), (p_7,$ *"Listen to a lecture"*$), (p_8,$ *"Delicious dinner"*$)$⟩

$\tau_2 =$ ⟨$(p_1), (p_2,$ *"Buy a coffee"*$), (p_3,$ *"Drive in a traffic jam"*$), (p_4), (p_5,$ *"Unload the sports equipment"*$), (p_6,$ *"Park a car"*$)$⟩

$\tau_3 =$ ⟨$(p_1), (p_2,$ *"Bump into a friend"*$), (p_3,$ *"Clothing procurement"*$), (p_4), (p_5,$ *"Cooked to perfection"*$)$⟩

Fig. 1. A Toy Example of Three Semantic Trajectories

can know not only where a user has been, but also what he/she has done. In this paper, we investigate offline semantic trajectory similarity search, which aims to find "similar" trajectories considering spatial-semantic perspective. In this work, we do not consider the temporal aspect of trajectories, as the semantic similarity is usually applied in time-insensitive applications, referring existing semantic trajectory analyses [16], [18], [76], [77]. Overall, semantic trajectory similarity search is important for location-based services, especially travel recommendation services. These offerings are extensively used in electronic maps and social platforms like Google Maps [1] and Foursquare [2]. Recently, there has been a multitude of semantic-based travel planning apps, e.g., Wanderlog [3], Rome2Rio [4], etc. Their behind idea is shown in Fig. 1, where three semantic trajectories exist, i.e., τ_1–τ_3. Although τ_1 and τ_2 are spatially close, they express distant semantics and are thus preferred by different groups. In contrast, τ_1 and τ_3 are semantically close and may be recommended to the same group like city tourists. Consequently, it is important to consider spatial proximity and semantic similarity when determining whether two trajectories are similar to each other.

However, when revisiting semantic trajectory analyses [16], [29], [48], [75], [75], [77], we identify three unsolved challenges in terms of diverse semantic modelings, large-amount data processing, and query efficiency, as detailed below.

Challenge I: How to model diverse semantic meanings? As shown in Fig. 1, semantic trajectories exhibit diverse meanings. For example, τ_1 records someone who chronically "*Chat with friends*" at the Coffee Shop, "*Try on a dress*" at the Shopping Mall, "*Listen to a lecture*" at the Sanskirt College, and had dinner at the Italian Restaurant while commenting

"*Delicious dinner*". As observed, two types of semantics exist, i.e., keywords and text descriptions, where the former records users' locations and the latter records users' activities. Note that, even if τ_1 and τ_2 go through the same location, it is possible to express different user activities. For example, τ_1 records someone who "*Chat with friends*" at the Coffee Shop while τ_2 "*Buy a coffee*" without meeting anyone. Besides, with the development of 5G technologies, semantic information has become increasingly diverse. However, existing studies [7], [11], [16], [18], [25], [55], [75], [76] mostly focus on keyword-based semantic modelings and cannot effectively model text descriptions. In the above example, existing methods directly model the activity "*Chat with friends*" with discrete keywords including "*Chat*", "*with*", and "*friends*". Leaving aside the disruption of semantic coherence of sentences, we cannot enumerate all keywords. **To address this challenge**, we propose a new representation method to model a semantic trajectory from its geo-sequence and semantic sequence, respectively. In a semantic sequence, we embed semantics into vectors, enabling robust semantic modeling. It is worth mentioning that, this is the first attempt to combine the embedding technique and raw spatial trajectory representation to perform spatial and semantic aware trajectory similarity searches. Finally, a linear combination method [7], [11], [25], [55] is adopted to combine the spatial and semantic similarity into a spatial-semantic similarity metric, while supporting popular trajectory distance measures including DTW [66], LCSS [57], and EDR [15].

Challenge II: How to design an efficient distributed semantic trajectory similarity search framework? Compared to raw trajectories that consist of locations, semantic trajectories introduce semantic events, leading to a great increase in data sizes. Taking the widely used trajectory datasets T-drive [68] and GeoLife [78] as examples, enriching them with semantics leads to 10 $\times$ storage increase. Even if we have embedded semantics as vectors, the storage size still increases more than 6 $\times$. Even worse, owing to the intricacy involved in semantic distance computation, the calculation of similarity between semantic trajectories entails higher complexity. According to our experimental study, after incorporating semantic information into T-drive, the processing time required for a random single query on a single machine exceeds 2,000 seconds. As a result, semantic trajectories may easily exceed the storage capacity and computational capability in a single machine, necessitating a distributed and scalable framework. However, most of the existing semantic trajectory studies deal with small data in a centralized environment. Although there are proposals [46], [51], [60], [67], [73], [74] for distributed similarity search over spatial trajectories, they do not consider the semantic aspect. There are also studies [13], [43], [61], [62] for distributed vector search, which cannot be directly employed due to their inability to leverage spatial information of trajectories. Overall, the key to building a distributed query framework is an effective partitioning strategy that considers both spatial and semantic aspects, so that on a query trajectory, most of the partitions in the cluster can be quickly pruned, which is challenging and none of the previous studies can achieve. **To address this challenge**, we propose a two-phase and computation-aware partitioning architecture over Spark [70]. In contrast to existing trajectory partitioning methods that typically use IDs [60], round-robin [74], or methods based on separated trajectory points and Minimum Bounding Rectangles (MBRs) [46], [51], [67], [73] to perform direct trajectory partitioning, we leverage both the fine-grained semantic and spatial features of trajectories, as well as the inter-trajectory distances to enable computation-aware distributed partitioning.

Challenge III: How to accelerate the query process? With the above computation-aware distributed data partitioning, the input trajectory dataset is assigned to different partitions. Given a query trajectory, it will be broadcasted into relevant partitions for parallel searching. This brings the idea of designing an effective index in each partition for local pruning. Existing studies typically build inverted indexes over keywords and manage trajectories with multiple inverted files. In each file, they build traditional trajectory indexes, including tree indexes [24], [26], [42], [45], [54] and grid indexes [12], [81]. However, there are two limitations. First, the inverted indexes cannot model semantics like text descriptions. Second, the traditional indexes cannot reduce the time complexity of scanning a partition, but only reduce the number of trajectories to be evaluated. For instance, although the grid indexes split the search process into grids, the number of trajectories in each grid is still dependent on the whole dataset. **To address this challenge**, we extend the HNSW index [40]), which was proposed for vector retrieval, for semantic trajectory similarity search. Specifically, in each data partition, we organize trajectories into a nearest neighbor graph by connecting items that are close in spatial-semantic distance with edges. Based on that, we build a semantic trajectory oriented HNSW index, called ST-HNSW, where the query starts at a node in the graph and iteratively traverses its neighboring nodes. Consequently, the query time depends on the search steps. Compared to prior studies that integrate classic spatial indexes with inverted indexes, ST-HNSW offers much better query efficiency. Besides, we also develop a series of optimization techniques on ST-HNSW for further query acceleration.

In this paper, we revisit semantic trajectory similarity search from semantic modeling, distributed partitioning, and optimized query. Overall, we make the below contributions.

- We propose a robust semantic trajectory representation method. Then, we adopt a linear combination way to evaluate spatial-semantic aware inter-trajectory similarities, while supporting three trajectory distance measures.
- We build a distributed semantic trajectory search framework. We propose a two-phase and computation-aware partitioning strategy so that given a query trajectory, most of the irrelevant partitions can be directly pruned.
- We develop the ST-HNSW index in each data partition for local query acceleration. Besides, a series of optimization techniques are designed to further speed up local queries.
- Extensive experiments on large-scale datasets demonstrate the effectiveness, efficiency, and scalability of the proposed framework over competitors.

TABLE I
STATISTICS OF RECENT SEMANTIC TRAJECTORY SEARCHES

Methods	Data Size	#measures	Processing	Publication
`PSTSJ` [16]	8 GB	1	Standalone	ICDE 20
`At2vec` [36]	0.3 GB	1	Standalone	TKDE 22
`T`k`Sk` [77]	3 GB	1	Standalone	WWW 17
`AKQST` [75]	0.3 GB	1	Standalone	ICDE 15
`Ours`	**377 GB**	**3**	**Distributed**	–

We release code at https://github.com/ZJU-DAILY/DSTSS. Due to space limitation, we put the complete manuscript here[1].

II. RELATED WORK

(A) Semantic Trajectory Representation. Existing semantic trajectory representation methods can be categorized into individually keywords-tagged representation [16], [18], [75]–[77] and globally keywords-tagged representation [37], [50], [82]. In an individual-based trajectory, a set of keywords is assigned to each point. In contrast, a global-based trajectory assigns keywords to the entire trajectory. Nevertheless, these methods possess limitations. First, they can only capture semantics based on keywords, disregarding other forms of semantics such as text. Second, the global-based representation fails to preserve the order of semantics, while the individual-based representation leads to redundant storage [29]. Last but not the least, keyword-based representation typically necessitates the construction of an inverted index using keywords [16]. In that case, as the number of keywords increases, the redundancy escalates. There are also works to attach keywords to road network constrained trajectories [29], [30], which is not the scope of this paper. Recently, [10], [20] introduced an approach for semantic trajectory similarity calculation that totally does not consider the spatial aspect in measuring inter-trajectory similarity. In this paper, we propose a new representation by decoupling the semantic trajectory into geographical and semantic sequences, respectively. This not only resolves the issue of redundant storage but also preserves the sequential characteristics.

(B) Spatial Keyword Queries. Given a semantic trajectory dataset $\mathcal{T}$, a query GPS point p, and a set of keywords $\mathcal{K}$, the objective is to retrieve semantic trajectories from the $\mathcal{T}$ that are both spatially close to point p and semantically similar to $\mathcal{K}$. The community typically employs inverted indexes, which can be classified into spatial-first [18], [75]–[77] and keyword-first [16], [47], [48]. The spatial-first indexes organize trajectory points using spatial indexes such as grids [12], [81] and R-tree [26]. Then, they construct an inverted file based on keywords at the bottom unit in the spatial index. In contrast, the keyword-first indexes create inverted indexes based on keywords and then establish spatial indexes. Despite the efforts, it is inefficient to deploy them into semantic trajectory similarity search. First, the query input only includes a single GPS point and a set of keywords, rather than a complete trajectory. Second, the keyword-based indexes are unable to evaluate the similarity of synonymous terms since they rely on exact keyword matches. In this paper, we perform semantic trajectory similarity searches considering its entire sequence while supporting text semantic modeling.

[1]https://github.com/ZJU-DAILY/DSTSS

TABLE II
SYMBOLS AND DESCRIPTIONS

Notation	Description
τ	A semantic trajectory of a moving object
GS	A geographical sequence of a semantic trajectory
SS	A semantic sequence of a semantic trajectory
$e, \boldsymbol{v}$	A semantic event and its embedding
$d(\cdot,\cdot)$	The pairwise distance between two elements
$\mathcal{D}(\cdot,\cdot)$	The spatial-semantic distance between trajectories
$\mathcal{D}^{(G)}$	The inter-distance between geo-sequences
$\mathcal{D}^{(S)}$	The inter-distance between semantic sequences
α	The weight between spatial and semantic aspects

(C) Semantic Trajectory Similarity Searches. The objective of the similarity search is to retrieve trajectories that are similar to a specified query trajectory. There are two types of solutions. The first type [10], [16], [20], [47], [48], [75], [77] relies on keyword-based inverted indexes. Specifically, they utilize the inverted indexes to create global indexes for semantic trajectories and then construct local spatial indexes such as R-trees. Ultimately, the similarity search is performed on these local indexes. However, they suffer from storage redundancy and exhibit lower performance. The second type [7], [11], [25], [30], [36], [63], [72] is based on deep learning. Specifically, they employ deep models like Seq2seq [53] and Attention [56] to embed the whole semantic trajectories into vectors. Then, they perform searches in the vector space. However, the learning-based methods fail to provide interpretable results. To the best of our knowledge, there is no work that utilizes a distributed approach for semantic-spatial trajectory similarity search except [48]. However, it still relies on keyword-based inverted index and the pruning method lacks efficiency.

The main statistics of the related semantic trajectory search methods are **summarized** in Table I, from which we find that, we propose the first distributed framework that supports large-scale data sizes and more trajectory distance measures.

III. SEMANTIC TRAJECTORY MODELING

Table II lists the frequently used symbols.

A. Semantic Trajectory Representation

When reviewing existing semantic trajectory representation methods, there are **individual-based** forms [16], [18], [75]–[77] and **global-based** forms [37], [50], [82]. An individual-based semantic trajectory is a time-ordered sequence of points, i.e., $\tau = \langle p_1, p_2, ..., p_n \rangle$, where each point p_i $(1 \leq i \leq n)$ contains a location l_i and an optional keyword set $\mathcal{K}_i$. Specifically, $\mathcal{K}_i = \{key_1, key_2, ...\}$ is used to describe the attributes of the corresponding location p_i. Fig. 2(a) shows two individual-based semantic trajectories, τ_1 and τ_2. In contrast, the global-based semantic trajectory representation methods directly assign a global keyword set $\mathcal{K}$ to the entire trajectory.

As observed, both individual-based and global-based representations rely on **raw keywords**. On the one hand, the semantics are not limited to keywords but also contain semantics such as text descriptions, which cannot be precisely delivered by finite keywords. On the other hand, as shown in Fig. 2(b), existing studies [16], [76], [77] typically enumerate all keywords to build inverted indexes, resulting in massive redundancy. To this end, we propose a new semantic trajectory representation manner. We first introduce the semantic event.

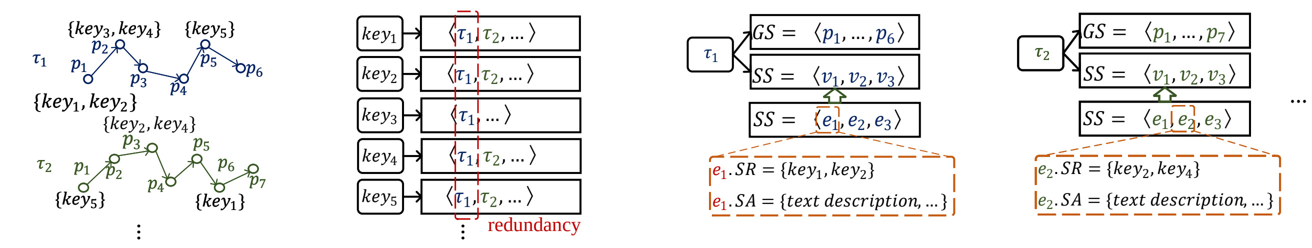

$\{key_3, key_4\}$
$\{key_5\}$
τ_1
p_1 p_2 p_3 p_4 p_5 p_6
$\{key_1, key_2\}$
$\{key_2, key_4\}$
τ_2
p_1 p_2 p_3 p_4 p_5 p_6 p_7
$\{key_5\}$
$\{key_1\}$
key_1 $\langle \tau_1, \tau_2, ... \rangle$
key_2 $\langle \tau_1, \tau_2, ... \rangle$
key_3 $\langle \tau_1, ... \rangle$
key_4 $\langle \tau_1, \tau_2, ... \rangle$
key_5 $\langle \tau_1, \tau_2, ... \rangle$
redundancy
τ_1
$GS = \langle p_1, ..., p_6 \rangle$
$SS = \langle v_1, v_2, v_3 \rangle$
$SS = \langle e_1, e_2, e_3 \rangle$
$e_1.SR = \{key_1, key_2\}$
$e_1.SA = \{text\ description, ...\}$
τ_2
$GS = \langle p_1, ..., p_7 \rangle$
$SS = \langle v_1, v_2, v_3 \rangle$
$SS = \langle e_1, e_2, e_3 \rangle$
$e_2.SR = \{key_2, key_4\}$
$e_2.SA = \{text\ description, ...\}$
...
(a) Individual-based semantic trajectories (b) Inverted-based modeling (c) Our methods

Fig. 2. Semantic Trajectory Representation

Definition 1. (Semantic Event, e**)** *A semantic event e is a tuple, $e = (SR, SA)$. $SR = \{key_1, key_2, ...\}$ is a keyword set describing the attributes of a semantic region. $SA = \{a_1, a_2, ...\}$ is a set of activities that happened at SR.*

According to Definition 1, a semantic region *SR* refers to a geo-area that is specified by a center point and a radius parameter. It expresses semantics (e.g., a shopping malls and its surroundings), similar to **individual-based** representation. In the real world, the keyword set $\mathcal{K}$ is collected from online maps. Simultaneously, $SA = \{a\}$ refers to semantic activities in *SR*, which are usually collected from check-in social networks. For instance, a can denote text descriptions such as users' activity phrases and comment sentences.

Semantic data acquisition. Semantic events can be easily obtained in large quantities from reality by utilizing the check-in feature on social platforms and combining it with the API of online maps to acquire semantic region information. Taking the first semantic event of τ_1 in Fig. 1 as an example, when someone posts a tweet with the content "$Chat\,with\,friends$" and the description obtained from querying the coordinates on an online map API is "$College$", this event can be modeled as $\tau_1.SS.e_1 = (\{\text{"}College\text{"}\}, \{\text{"}Chat\,with\,friends\text{"}\})$. However, the existing mainstream **keyword-based** methods [36], [63], [72], [76], [77] can only model this event with independent keywords, i.e., $\tau_1.SS.e_1 = \{\text{"}College\text{"}, \text{"}Chat\text{"}, \text{"}with\text{"}, \text{"}friends\text{"}\}$, which lacks coherent sentence meaning.

Following existing semantic trajectory studies [16], [18], [75]–[77], semantic similarity is typically applied in offline applications. In such scenarios, the temporal information of semantic events usually has little impact. For example, in the above example of "$Chat\,with\,friends$", whether it occurs in the morning or afternoon, it generally does not significantly alter its behavioral characteristics. In this paper, the precise timestamp information is not included in trajectory modeling, leaving a future study in the community.

Definition 2. (Semantic Sequence, ***SS*****)** *A semantic sequence SS of a trajectory τ is consists of m time-ordered semantic events, i.e., $\tau.SS = \langle e_1, e_2, ..., e_m \rangle$, where e is defined above.*

Definition 3. (Geographical Sequence, ***GS*****)** *A geographical sequence GS of a trajectory τ is consists of n time-ordered points, i.e., $\tau.GS = \langle p_1, p_2, ..., p_n \rangle$, where p is a 2-dimensional (x, y) location.*

Definition 4. (Semantic Trajectory, τ**)** *A semantic trajectory is denoted by $\tau = \langle GS, SS \rangle$, where GS denotes τ's geographic sequence and SS denotes τ's semantic sequence.*

Motivation. Given a raw trajectory, we model it from geographical and semantic aspects. This is because, the spatial points and semantic events are not strictly one-to-one correspondence: not every point has its semantic event and not every semantic event corresponds to a point. By decoupling the spatial aspect from the semantic aspect, we can flexibly describe users' mobility features.

In the rest of paper, if there are no special instructions, we refer to **semantic trajectory** as **trajectory**.

To enable robust semantic modeling, we apply FastText [6] to embed *SR*s and *SA*s into d-dimensional vectors, i.e., $SR \rightarrow \overrightarrow{SR} \in \mathbb{R}^d$, $SA \rightarrow \overrightarrow{SA} \in \mathbb{R}^d$. Then, for each semantic event e_i, we obtain its embedding $\boldsymbol{v_i} = \overrightarrow{e_i.SR} + \overrightarrow{e_i.SA}$. Recall Definition 2, given a semantic sequence $SS = \langle e_1, e_2, ..., e_m \rangle$, we transform it into a vector sequence $SS = \langle \boldsymbol{v_1}, \boldsymbol{v_2}, ..., \boldsymbol{v_m} \rangle$. Fig. 2(c) shows the process. Note that, FastText is a mature tool for fast text embedding and can be easily replaced with any other text embedding methods.

Semantic event distance. We use $\boldsymbol{v}[i]$ to denote the i-th dimension of $\boldsymbol{v}$ $(1 \leq i \leq d)$. For two semantic events e_a and e_b, as well as their vectors v_a and v_b, the semantic event distance is $d(e_a, e_b) = \sqrt{\sum_{i=1}^{d}(v_a[i] - v_b[i])^2}$. By default, we treat semantic sequences as embedded sequences.

Benifits analyses. Through the utilization of embedding-based semantic information modeling, semantic events are converted into vector representations, thereby reducing the computation of distances between them. This approach offers a more effective means of capturing the actual meaning of semantic events compared to keyword-based matching methods. Additionally, by exclusively embedding semantic events, the sequential order of geographic and semantic sequences is retained, enabling the application of sequence distance metrics.

B. Trajectory Distance Functions

The focus of our work is not to investigate different similarity measures. Instead, we adopt the well-known DTW [66], LCSS [57], and EDR [15] for spatial-semantic distance.

Definition 5. (Trajectory Distance Function) *Given two trajectories τ_i and τ_j, their inter-distance $\mathcal{D}(\tau_i, \tau_j)$ is defined as the linear combination of $\mathcal{D}^{(G)}(\tau_i, \tau_j)$ and $\mathcal{D}^{(S)}(\tau_i, \tau_j)$:*

$$\mathcal{D}(\tau_i, \tau_j) = \alpha \cdot \mathcal{D}^{(G)}(\tau_i.GS, \tau_j.GS) + (1-\alpha) \cdot \mathcal{D}^{(S)}(\tau_i.SS, \tau_j.SS) \tag{1}$$

where $\mathcal{D}^{(G)}(\cdot, \cdot)$ denotes the distance between their geo-sequences, $\mathcal{D}^{(S)}(\cdot, \cdot)$ denotes the distance between their semantic sequences, and α balances the weight between them. $\mathcal{D}$ denotes DTW [66], LCSS [57], or EDR [15].

Next, we use LCSS [57] to illustrate the spatial-semantic trajectory distance computation. Given two trajectories $\tau_a = \langle GS_a, SS_a \rangle$ and $\tau_b = \langle GS_b, SS_b \rangle$, the LCSS$^{(G)}$ and LCSS$^{(S)}$ distances between τ_a and τ_b are computed below.

$$\mathrm{LCSS}^{(\mathrm{G})}(GS_a, GS_b) = \begin{cases} 0, \text{ if } n = 0 \text{ or } n' = 0 \\ 1 + \mathrm{LCSS}^{(\mathrm{G})}(R(GS_a), R(GS_b)), \\ \quad \text{if } d(H(GS_a), H(GS_b)) \leq \varepsilon \ \& \ ||GS_a| - |GS_b|| \leq \delta \\ max(\mathrm{LCSS}^{(\mathrm{G})}(R(GS_a), GS_b), \mathrm{LCSS}^{(\mathrm{G})}(GS_a, R(GS_b))), \\ \quad \text{otherwise} \end{cases} \tag{2}$$

$$\mathrm{LCSS}^{(\mathrm{S})}(SS_a, SS_b) = \begin{cases} 0, \text{ if } m = 0 \text{ or } m' = 0 \\ 1 + \mathrm{LCSS}^{(\mathrm{S})}(R(SS_a), R(SS_b)), \\ \quad \text{if } d(H(SS_a), H(SS_b)) \leq \varepsilon \ \& \ ||SS_a| - |SS_b|| \leq \delta \\ max(\mathrm{LCSS}^{(\mathrm{S})}(R(SS_a), SS_b), \mathrm{LCSS}^{(\mathrm{S})}(SS_a, R(SS_b))), \\ \quad \text{otherwise} \end{cases} \tag{3}$$

where $R(\cdot)$ refers to the sequence after removing the first element. $H(\cdot)$ refers the sequence's head. ε and δ refer to the distance threshold and order gap threshold of two elements. The difference between LCSS$^{(G)}$ and LCSS$^{(S)}$ is $d(\cdot, \cdot)$, which denotes **point-to-point** distance in LCSS$^{(G)}$ and **vector-to-vector** distance in LCSS$^{(S)}$. Based on a linear combination of LCSS$^{(G)}$ and LCSS$^{(S)}$, we evaluate the distance between τ_a and τ_b considering both spatial and semantic aspects. The computations of DTW [66] and EDR [15] measures are omitted, as they share a similar process.

C. Problem Statements

Note that, we consider a trajectory of the two-dimensional free space since it represents the most common scenarios. It is straightforward to extend our solutions to road networks.

Definition 6. (Problem Definition) *Given a set of trajectories $\mathcal{T}$, a query trajectory τ_q, a positive integer k, a distance function $\mathcal{D}(\cdot, \cdot)$, and an approximate guarantee ϵ, a semantic trajectory similarity search query returns a set of trajectories $\mathcal{R} \subset \mathcal{T}$, where $|\mathcal{R}| = k$ and $\forall \tau_i \in \mathcal{R}, \forall \tau_j \in \mathcal{T} - \mathcal{R}, \mathcal{D}(\tau_j, \tau_q)| = (1 + \epsilon) \cdot \mathcal{D}(\tau_i, \tau_q)$.*

Note that, if $\epsilon > 0$, we perform approximated similarity searches; if $\epsilon = 0$, we perform exact similarity searches. "$| =$" means "$\geq$" when $\mathcal{D}(\cdot, \cdot)$ denotes the DTW and EDR measures, and it means "$\leq$" when $\mathcal{D}(\cdot, \cdot)$ denotes the LCSS measure. This is because the smaller the distance between τ_i and τ_j, the more similar they are in DTW and EDR, while a larger LCSS distance means that they are more similar.

D. Alternative Baselines

Before introducing our framework, we detail three distributed baselines by extending existing methods.

<u>`ID-Force`:</u> It partitions the dataset based on trajectory IDs, i.e., trajectories are distributed into partitions via a HashPartitioner [41]. Then, a local index is constructed in each partition, where existing techniques such as HNSW [40] can be employed. Based on that, each query τ_q is broadcasted

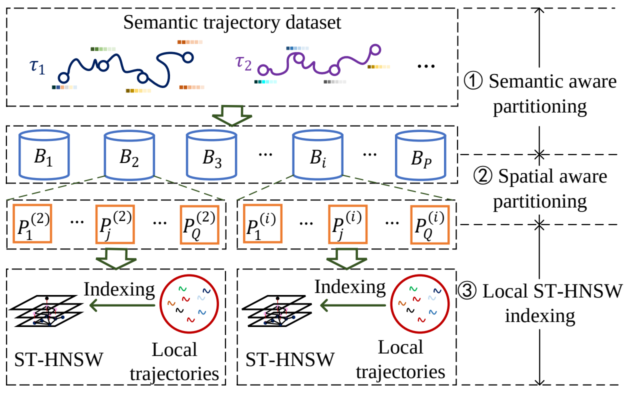


Fig. 3. The Framework Overview

to every partition, and all local results are merged to obtain the final results. While this distributed extension is simple, its pruning capability is limited, as each partition is scanned.

<u>`Dis-PSTSJ`:</u> It is a distributed variant of PSTSJ [16] that supports parallel semantic trajectory similarity computation. We extend PSTSJ as follows. First, all trajectories are organized via an inverted list, in which each file comprises a subset of the dataset. With HashPartitioner [41], we distribute inverted files across partitions. During the search, each query τ_q is dispatched to partitions where the inverted file satisfies the keyword constraints. This approach distributes trajectories leveraging keyword similarities. However, its pruning ability is still limited as the distributed inverted list solely exploits semantic features without considering spatial features. Actually, a query trajectory typically contains multiple keywords and needs to check multiple partitions to avoid result missing.

<u>`DITA-S`:</u> It is an extension of a well-known distributed spatial trajectory similarity search framework DITA [51]. We first employ DITA for spatial-aware similarity search, followed by a brute-force for semantic-aware similarity search. This two-phase filter returns similar trajectories of a query trajectory considering both spatial and semantic aspects.

IV. The Framework Overview

The key to building an efficient distributed framework lies in data partitioning so that on a query trajectory, most of the partitions can be quickly skipped. Unlike the above baselines simply utilizing IDs or keywords for coarse-grained partitioning, we leverage both semantic and spatial features, as well as inter-trajectory distance simultaneously.

Fig. 3 gives the framework overview, including ①semantic aware partitioning, ②spatial aware partitioning, and ③local ST-HNSW indexing. Specifically, the input trajectories are partitioned into P <u>**data batches**</u> via semantic features, where trajectories that are closer in the semantic distance are grouped in the same batch. For each data batch, we further distribute it into Q <u>**data partitions**</u> via spatial features, where trajectories that are closer in the spatial distance are grouped in the same partition. Based on the above two-phase and computation-aware partitioning, data batches or data partitions that are distant in distance can be excluded from distance computation with the query trajectory τ_q. Finally, in each local partition, we build an ST-HNSW index to speed up local queries.

V. Two-Phase and Computation-Aware Partitioning

Given a raw trajectory dataset $\mathcal{T} = \{\tau_1, \tau_2, ..., \tau_n\}$ where each τ consists of a geo-sequence and a semantic sequence, we first distribute $\mathcal{T}$ into P data batches, i.e., $\mathcal{T} \rightarrow \mathcal{T}^{'} = \{B_1, B_2, ..., B_P\}$. Then we distribute each B_i $(1 \leq i \leq P)$ into Q data partitions, i.e., $B_i = \{P_1^{(i)}, P_2^{(i)}, ..., P_Q^{(i)}\}$, where $P_j^{(i)}$ $(1 \leq j \leq Q)$ denotes the j-th data partition of the i-th data batch. Overall, $\mathcal{T} \rightarrow \mathcal{T}^{'} \rightarrow \mathcal{T}^{''} = \{P_1^{(1)}, ..., P_Q^{(1)}; P_1^{(2)}, ..., P_Q^{(2)}; ...; P_1^{(P)}, ..., P_Q^{(P)}\}$.

A. Semantic Aware Partitioning

As shown in Fig. 4, given $\mathcal{T} = \{\tau_1, ..., \tau_n\}$, its semantic event vectors consist of $\mathcal{V} = \{v | v = \overrightarrow{e.SR} + \overrightarrow{e.SA}\ \&\ e \in \tau.SS\ \&\ \tau \in \mathcal{T}\}$. Then, we cluster $\mathcal{V}$ into P groups, i.e., $G_1, ..., G_P$. Based on that, we partition $\mathcal{T}$ into P data batches, i.e., $\mathcal{T} \rightarrow \mathcal{T}^{'} = \{\hat{B_1}, ..., \hat{B_P}\}$, where $\hat{B_i} = \{\tau | \tau \in \mathcal{T}\ \&\ \exists v \in \tau.SS\ \&\ v \in G_i\}$ and $1 \leq i \leq P$. Obviously, this grouping-based partitioning approach reduces the storage redundancy of the classic inverted-based partitioning methods.

1) DPiSAX-based partitioning: Specifically, we adopt DPiSAX [62] for semantic event vector clustering, which uses the iSAX encoding to compress vectors into sequences consisting of binary strings. According to the previous work [32], given two semantic vectors $v_a \in \mathbb{R}^d$ and $v_b \in \mathbb{R}^d$, as well as their iSAX embeddings $iSAX(v_a) = \langle x_1, ..., x_\omega \rangle$ and $iSAX(v_b) = \langle y_1, ..., y_\omega \rangle$, the distance between them is

$$d(iSAX(v_a), iSAX(v_b)) = \sqrt{\frac{d}{\omega}}\sqrt{\sum_{i=1}^{\omega} d(x_i, y_i)} \tag{4}$$

where $d(x_i, y_i)$ denotes the L-2 distance between two iSAX elements, ω is the length of the $iSAX$ sequences, and $\omega \ll d$.

Lemma 1. *Given two semantic event vectors $v_a \in \mathbb{R}^d$ and $v_b \in \mathbb{R}^d$ with their iSAX representations $iSAX(v_a)$ and $iSAX(v_b)$, $d(v_a, v_b) = \sqrt{\sum_{i=1}^{d}(v_a[i] - v_b[i])^2}$ is the vector distance. Then we have $d(v_a, v_b) \geq d(iSAX(v_a), iSAX(v_b))$.*

Proof. The proof can be found elsewhere [32]. □

According to Lemma 1, given two semantic events, their $iSAX$ distance is the lower bound of their semantic event distance and thus can be used for pruning unnecessary inter-vector computation. Besides, the computation of $d(iSAX(v_a), iSAX(v_b))$ is far more efficient than the computation of $d(v_a, v_b)$ due to $\omega \ll d$. Based on the vector clustering results, we distribute the dataset into P data batches, i.e., $\mathcal{T} \rightarrow \mathcal{T}^{'} = \{\hat{B_1}, ..., \hat{B_P}\}$.

Consequently, in each data batch $\hat{B}$, all trajectories share the same iSAXs, i.e., $\forall v_a, v_b \in \hat{B}, iSAX(v_a) = iSAX(v_b)$. Thus, we refer to the iSAX code of $\hat{B}$ as $iSAX(\hat{B})$. Then, we define the distance between a trajectory and a data batch.

Definition 7. (Trajectory-to-Batch Distance) *Given a trajectory τ and a data batch $\hat{B}$, we define their distance as*

$$TB(\tau, \hat{B}) = \min_{v \in \tau.SS} d(iSAX(v), iSAX(\hat{B})) \tag{5}$$

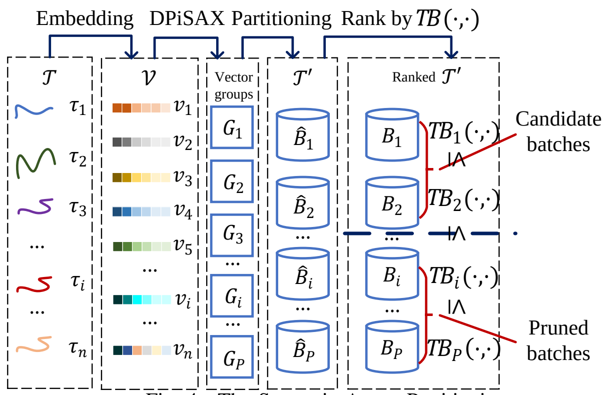

Embedding
DPiSAX Partitioning
Rank by $TB(\cdot,\cdot)$
$\mathcal{T}$
$\mathcal{V}$
Vector groups
$\mathcal{T}'$
Ranked $\mathcal{T}'$
τ_1 τ_2 τ_3 τ_i τ_n
v_1 v_2 v_3 v_4 v_5 v_i v_n
G_1 G_2 G_3 G_i G_P
$\hat{B}_1$ $\hat{B}_2$ $\hat{B}_i$ $\hat{B}_P$
B_1 B_2 B_i B_P
$TB_1(\cdot,\cdot)$ $TB_2(\cdot,\cdot)$ $TB_i(\cdot,\cdot)$ $TB_P(\cdot,\cdot)$
Candidate batches
Pruned batches

Fig. 4. The Semantic Aware Partitioning

2) Data batch pruning in Search: Based on Definition 7, given a query trajectory τ_q and a data batch $\hat{B}$, we are inspired to derive the relations between $TB(\tau_q, \hat{B})$ and the inter-trajectory semantic distance $\mathcal{D}^{(S)}(\tau_q, \tau_i)$ to prune all trajectories $\tau_i \in \hat{B}$.

Given two sequences $G = \langle g_1, g_2, ..., g_n \rangle$ and $S = \langle s_1, s_2, ..., s_m \rangle$, the LCSS distance between them via Equ. 2 or 3 is reformulated as

$$\begin{aligned} LCSS(G, S) &= \sum_{c=1}^{C} \mathcal{W}^c(g_i, s_j), \mathcal{W}^c(g_i, s_j) \\ &= \begin{cases} 1 & \text{if } d(g_i, s_j) \leq \varepsilon\ \&\ |i-j| \leq \delta \\ 0 & otherwise \end{cases} \end{aligned} \tag{6}$$

where C denotes the number of matched element pairs from G and S, and $\mathcal{W}^c(\cdot, \cdot)$ denotes the weight of c-th matched pair. If G and S are semantic sequences, $d(g_i, s_j)$ denotes vector-to-vector distance; if G and S are geo-sequences, $d(g_i, s_j)$ denotes point-to-point distance. DTW and EDR can be reformulated similarly.

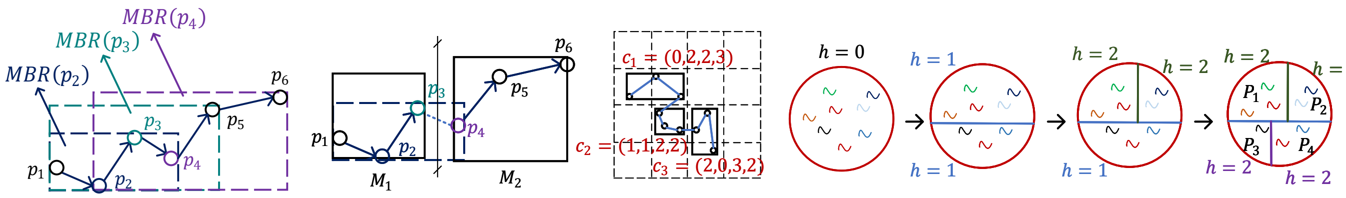
$$\begin{aligned} DTW(G, S) &= \sum_{c=1}^{C} \mathcal{W}^c(g_i, s_j), \mathcal{W}^c(g_i, s_j) \\ &= \begin{cases} d(g_i, s_j) & \text{if } |i-j| \leq \delta \\ 0 & otherwise \end{cases} \end{aligned} \tag{7}$$

$$\begin{aligned} EDR(G, S) &= \sum_{c=1}^{C} \mathcal{W}^c(g_i, s_j), \mathcal{W}^c(g_i, s_j) \\ &= \begin{cases} 1 & \text{if } d(g_i, s_j) \geq \varepsilon\ \&\ |i-j| \leq \delta \\ 0 & otherwise \end{cases} \end{aligned} \tag{8}$$

We unify the weight $\mathcal{W}$ of the above distance functions.

$$\mathcal{W}(d(\cdot,\cdot)) = \begin{cases} d(\cdot,\cdot) & \text{if } \mathcal{D} \text{ is } DTW \\ \begin{cases} 1 & \text{if } d(\cdot,\cdot) \leq \varepsilon \\ 0 & otherwise \end{cases} & \text{, if } \mathcal{D} \text{ is } LCSS \\ \begin{cases} 1 & \text{if } d(\cdot,\cdot) \geq \varepsilon \\ 0 & otherwise \end{cases} & \text{, if } \mathcal{D} \text{ is } EDR \end{cases} \tag{9}$$

where $d(\cdot, \cdot)$ denotes the element-pair distance.



(a) Envelope (b) Segmentation (c) Encoding (d) Partitioning

Fig. 5. The Spatial Aware Partitioning

Lemma 2. *Given a query trajectory τ_q, a trajectory measure $\mathcal{D}$ (i.e., DTW, LCSS, or EDR), and a semantic aware partitioned trajectory dataset $\mathcal{T}' = \{\hat{B}_1, ..., \hat{B}_P\}$. Then $\forall \hat{B} \in \mathcal{T}'$, $\forall \tau \notin \bigcup_{\hat{B}' \in \mathcal{T}' \& TB(\tau_q, \hat{B}') < TB(\tau_q, \hat{B})} \hat{B}'$, we have $\mathcal{D}(\tau_q.SS, \tau.SS) \mid= \mathcal{W}(TB(\tau_q, \hat{B}))$.*

Proof. Taking $\mathcal{D}$ is DTW as an example. According to Equations 7 and 9, the inter-trajectory distance is determined by the distance between one or several item pairs among the sequences. Thus, $\forall \hat{B} \in \mathcal{T}'$ and $\forall \tau \in \hat{B}$, $\mathcal{D}(\tau_q.SS, \tau.SS) \geq \mathcal{W}(\min_{v_i \in \tau.SS} \min_{v_j \in \tau_q.SS} d(v_i, v_j))$. When $\tau \notin \bigcup \hat{B}'$ where $\hat{B}' \in \mathcal{T}'$ & $TB(\tau_q, \hat{B}') < TB(\tau_q, \hat{B})$, according to Definition 7, we can derive that $\forall v_i \in \tau.SS$, $TB(\tau_q, \hat{B}) \leq \min_{v_j \in \tau_q.SS} d(iSAX(v_i), iSAX(v_j))$, that is $TB(\tau_q, \hat{B}) \leq \min_{v_i \in \tau.SS} \min_{v_j \in \tau_q.SS} d(iSAX(v_i), iSAX(v_i))$. Based on Lemma 1, $\min_{v_i \in \tau.SS} \min_{v_j \in \tau_q.SS} d(iSAX(v_i), iSAX(v_i)) \leq \min_{v_i \in \tau.SS} \min_{v_j \in \tau_q.SS} d(v_i, v_j)$. Overall, we have $\mathcal{W}(TB(\tau_q, \hat{B})) \leq \mathcal{D}(\tau_q.SS, \tau.SS)$. EDR and LCSS measures share a similar proof process and thus are omitted. □

Based on Lemma 2, we show how to prune irrelevant data batches. Given a query trajectory τ_q, a trajectory distance function $\mathcal{D}$ (i.e., DTW, LCSS, or EDR), and a semantic aware partitioned dataset $\mathcal{T}'$, we rank $\mathcal{T}'$ such that $TB(\tau_q, B_i) \leq TB(\tau_q, B_j)$ when $i \leq j$. Here, $1 \leq i \leq P, 1 \leq j \leq P$, $B_i \in \mathcal{T}'$, and $B_j \in \mathcal{T}'$. Fig. 4 shows the ranking process. Assuming the current search batch is B_i, if $\mathcal{W}(TB(\tau_q, B_i))$ satisfies the query criteria, e.g., $\mathcal{W}(TB(\tau_q, B_i))$ has exceeded the distance between τ_q and the farthest trajectory in returned trajectories on the scanned data batches B_j ($1 \leq j \leq i$), the data batches ranked behind B_i can be pruned.

B. Spatial Aware Partitioning

Fig. 5 shows the spatial aware partitioning, involving: (1) envelope, (2) segmentation, (3) encoding, and (4) partitioning.

1) Envelope:

Definition 8. (MBR Sequence) *Given a geo-sequence $GS = \langle p_1, p_2, ..., p_n \rangle$ and an order constraint $\delta \in \mathbb{Z}$, we transform GS into an MBR sequence $MS = \langle MBR(p_1), MBR(p_2), ..., MBR(p_n) \rangle$, where $MBR(p_i) = (x_i^l, y_i^l, x_i^u, y_i^u)$ that is computed as*

$$\begin{aligned} x_i^l &= \min\{x_j | i-\delta \leq j \leq i+\delta\}, \\ y_i^l &= \min\{y_j | i-\delta \leq j \leq i+\delta\}, \\ x_i^u &= \max\{x_j | i-\delta \leq j \leq i+\delta\}, \\ y_i^u &= \max\{y_j | i-\delta \leq j \leq i+\delta\} \end{aligned} \tag{10}$$

As shown in Fig. 5(a), when $\delta = 2$, the blue dashed box is the MBR of p_2, the green dashed box is the MBR of p_3, and the purple dashed box is the MBR of p_4.

Definition 9. (MBR-to-MBR Distance) *Given two MBRs $M_i = MBR(x_i^l, y_i^l, x_i^u, y_i^u)$ and $M_j = MBR(x_j^l, y_j^l, x_j^u, y_j^u)$, their inter distance is $d(M_i, M_j) = \sqrt{d_x^2 + d_y^2}$, where d_x and d_y are*

$$d_x = \begin{cases} 0 & \text{if } [x_i^l, x_i^u] \cap [x_j^l, x_j^u] \neq \varnothing \\ \min(x_j^u - x_i^l, x_i^u - x_j^l) & \text{otherwise} \end{cases} \tag{11}$$

$$d_y = \begin{cases} 0 & \text{if } [y_i^l, y_i^u] \cap [y_j^l, y_j^u] \neq \varnothing \\ \min(y_j^u - y_i^l, y_i^u - y_j^l) & \text{otherwise} \end{cases} \tag{12}$$

Definition 10. (MBR-sequence to MBR-sequence Distance) *Given two geo-sequences GS_a and GS_b as well as their MBR sequences MS_a=$\langle MBR(p_1^a)$, ..., $MBR(p_m^a) \rangle$ and MS_b=$\langle MBR(p_1^b)$, ..., $MBR(p_n^b) \rangle$, assume that $m \leq n$, we define their inter-distance as*

$$\begin{aligned} MSD(MS_a, MS_b) = &\sum_{i=1}^{m} \mathcal{W}(d(MBR(p_i^a), MBR(p_i^b))) + \\ &\sum_{i=m+1}^{\min(n, m+\delta)} \mathcal{W}(d(MBR(p_m^a), MBR(p_i^b))) \end{aligned} \tag{13}$$

Lemma 3. *Given GS_a and GS_b, as well as their MBR sequences MS_a and MS_b, then we have $\mathcal{D}(GS_a, GS_b) \mid= MSD(MS_a, MS_b)$.*

Proof. Taking $\mathcal{D}$ is DTW as an example. Given $p_i \in GS_a$ & $p_j \in GS_b$, we get $d(p_i, p_j) \geq d(MBR(p_i), MBR(p_j))$. Assume that the length of GS_a and GS_b are m and n, respectively, where $m \leq n$, $\mathcal{D}(GS_a, GS_b) = \sum_{c=1}^{C} \mathcal{W}^c \geq \sum_{i=1}^{m} \mathcal{W}$ $(d(MBR(p_i^a), MBR(p_i^b)))$ + $\sum_{i=m+1}^{\min(n,m+\delta)} \mathcal{W}$ $(d(MBR(p_m^a), MBR(p_i^b)))$. Then $\mathcal{D}(GS_a, GS_b) \geq MSD(MS_a, MS_b)$. The above proof still holds when $n \leq m$. EDR and LCSS measures share a similar proof and thus are omitted. □

Based on Lemma 3, given two trajectories, we first envelope them into MBR sequences. Then, we compute the distance of their MBR sequences, which can be used to prune unnecessary geo-sequence spatial distance computation. However, given a trajectory, different points may share the same MBR. To accelerate the distance computation between MBR sequences, given an MBR sequence $MS = \langle MBR(p_1), MBR(p_2), ..., MBR(p_n) \rangle$, we segment MS as below.

2) Segmentation:

Definition 11. (MBR Sequence Segmentation) *Given a geo-sequence $GS = \langle p_1, p_2, ..., p_n \rangle$, its MBR sequence MS =*

$\langle MBR(p_1), MBR(p_2), ..., MBR(p_n)\rangle$, and a segment length $\mathcal{L}$, we segment MS into a sequence MS' that consists of $\frac{n}{\mathcal{L}}$ MBRs, i.e., $MS' = \langle M_1, M_2, ..., M_{\frac{n}{\mathcal{L}}}\rangle$, where $M_i = (sx_i^l, sy_i^l, sx_i^u, sy_i^u)$ ($1 \le i \le \frac{n}{\mathcal{L}}$) is computed as

$$\begin{aligned} sx_i^l &= \min_{j=\mathcal{L}(i-1)+1}^{\mathcal{L}i} x_j^l, \quad sy_i^l = \min_{j=\mathcal{L}(i-1)+1}^{\mathcal{L}i} y_j^l \\ sx_i^u &= \max_{j=\mathcal{L}(i-1)+1}^{\mathcal{L}i} x_j^u, \quad sy_i^u = \max_{j=\mathcal{L}(i-1)+1}^{\mathcal{L}i} y_j^u \end{aligned} \tag{14}$$

Overall, given a geo-sequence *GS* of p points, we first envelope *GS* into an MBR sequence *MS* of p MBRs, i.e., $GS = \{p_1, p_2, ..., p_n\} \to MS = \{MBR(p_1), MBR(p_2), ..., MBR(p_n)\}$. Then, we segment *MS* into a simplified MBR sequence that consists of $\frac{n}{\mathcal{L}}$ MBRs, i.e., $MS \to MS' = \{M_1, M_2, ..., M_{\frac{n}{\mathcal{L}}}\}$, where the length of each MBR is $\mathcal{L}$. As shown in Fig. 5(b), 2 MBRs are enough to bound the geo-sequence.

3) Encoding:

Definition 12. (MBR Sequence Encoding) *Given a geo-sequence GS of n points, its segmented MBR-sequence MS' = $\langle M_1, ..., M_i, ..., M_{\frac{n}{\mathcal{L}}}\rangle$, and a geo-span (x_l, x_r, y_d, y_u), we encode MS' into $\mathtt{MS} = \langle c_1, ..., c_i, ..., c_{\frac{n}{\mathcal{L}}}\rangle$, where c_i=(c_i^l, c_i^d, c_i^r, c_i^u) = $(\lfloor\frac{sx_i^l - x_l}{(x_r - x_l)/2^\omega}\rfloor, \lfloor\frac{sy_i^l - y_d}{(y_u - y_d)/2^\omega}\rfloor, \lceil\frac{sx_i^u - x_l}{(x_r - x_l)/2^\omega}\rceil, \lceil\frac{sy_i^u - y_d}{(y_u - y_d)/2^\omega}\rceil)$ is the encoded code of M_i and ω controls the length of the binary code.*

Specifically, for each MBR M_i in MS', we compute its code c_i=(c_i^l, c_i^d, c_i^r, c_i^u), where c_i^l and c_i^d denote the bottom left location of M_i while c_i^r and c_i^u denote top right location. In Fig. 5(c), when $\omega = 2$, we split the space into 4 × 4 grids. The trajectory is segmented to an MBR sequence that consists of 3 MBRs, where the codes of MBRs are $c_1 = (0, 2, 2, 3)$, $c_2 = (1, 1, 2, 2)$, and $c_3 = (2, 0, 3, 2)$.

Definition 13. (Encoded Inter MBR-sequence Distance) *Given two encoded MBR-sequences $\mathtt{MS}_a = \langle c_1^a, ..., c_{sa}^a\rangle$ and $\mathtt{MS}_b = \langle c_1^b, ..., c_{sb}^b\rangle$, assume $sa \le sb$, the distance between $\mathtt{MS}_a$ and $\mathtt{MS}_b$ is*

$$\begin{aligned} EMSD(\mathtt{MS}_a, \mathtt{MS}_b) = \sum_{i=1}^{sa} \mathcal{W}(d(c_i^a, c_i^b)) + \\ \sum_{i=sa}^{\min(sa+\lceil\delta/\mathcal{L}\rceil, sb)} \mathcal{W}(d(c_{sa}^a, c_i^b)) \end{aligned} \tag{15}$$

Lemma 4. *Given two MBR sequences MS_a and MS_b of two geo-sequences, as well as their segmented and encoded MBR sequences $\mathtt{MS}_a$ and $\mathtt{MS}_b$. When $\mathtt{MS}_a$ and $\mathtt{MS}_b$ have the same fixed length $\mathcal{L}$ of MBR segment, then we have $MSD(MS_a, MS_b)$ $| = \mathcal{L} \cdot EMSD(\mathtt{MS}_a, \mathtt{MS}_b)$.*

Proof. Taking $\mathcal{D}$ is DTW as an example. Given a geo-sequence GS_a with length m, as well as its segmented and encoded MBR sequence $\mathtt{MS}_a$ with length *sa*, according to Equation 14, $\forall i \in [1, sa]$ and $\forall j \in [\mathcal{L}(i-1)+1, \mathcal{L}i]$, we have $[x_j^l, x_j^u] \subseteq [sx_i^l, sx_i^u]$ and $[y_j^l, y_j^u] \subseteq [sy_i^l, sy_i^u]$. Then we derive that $MBR(p_j^a) \subseteq M_i = (sx_i^l, sy_i^l, sx_i^u, sy_i^u)$. Similarly, according to definition 12, we have $M_i \subseteq c_i^a$. Therefore $MBR(p_j^a) \subseteq c_i^a$. For another geo-sequence GS_b with length b, as well as its segmented and encoded MBR sequence $\mathtt{MS}_b$ with length *sb*, we also derive that $MBR(p_j^b) \subseteq c_i^b$. Assume that $m{<}n$, then $\forall i \in [1, sa]$ and $\forall j \in [\mathcal{L}(i-1)+1, \mathcal{L}i]$, $d(MBR(p_j^a), MBR(p_j^b)) \ge d(c_i^a, c_i^b)$, i.e., $\sum_{i=\mathcal{L}(i-1)+1}^{\mathcal{L}i} d(MBR(p_j^a), MBR(p_j^b)) \ge \mathcal{L} \cdot d(c_i^a, c_i^b)$. Therefore, we have $\sum_{i=1}^{sa} \sum_{j=\mathcal{L}(i-1)+1}^{\mathcal{L}i} d(MBR(p_j^a), MBR(p_j^b))$ $\ge \sum_{i=1}^{sa} \mathcal{L} \cdot d(c_i^a, c_i^b)$, i.e., $\sum_{i=1}^{m} d(MBR(p_i^a), MBR(p_i^b)) \ge \mathcal{L} \sum_{i=1}^{sa} d(c_i^a, c_i^b)$. Similarly, we derive that $\sum_{i=m+1}^{\min(m+\delta,n)} d(MBR(p_m^a), MBR(p_i^b)) \ge \mathcal{L} \sum_{i=sa}^{\min(sa+\lceil\delta/\mathcal{L}\rceil, sb)} d(c_m^a, c_i^b)$. Then $MSD(GS_a, GS_a) = \sum_{i=1}^{m} \mathcal{W}(d(MBR(p_i^a), MBR(p_i^b)))$ + $\sum_{i=m+1}^{\min(m+\delta,n)} \mathcal{W}(d(MBR(p_m^a), MBR(p_i^b))) \ge \mathcal{L} \cdot \sum_{i=1}^{sa} \mathcal{W}(d(c_i^a, c_i^b))$ + $\sum_{i=sa}^{\min(sa+\lceil\delta/\mathcal{L}\rceil, sb)} \mathcal{W}(d(c_{sa}^a, c_i^b))$. Therefore, we have $MSD(GS_a, GS_b) \ge \mathcal{L} \cdot EMSD(\mathtt{MS}_a, \mathtt{MS}_b)$. EDR and LCSS share a similar proof process and are omitted. □

4) Partitioning:

Definition 14. (Partition Code of an MBR) *Given an encoded and segmented MBR sequence $\mathtt{MS}$, for each MBR $c_i = (c_i^l, c_i^d, c_i^r, c_i^u) \in \mathtt{MS}$, the partition code of c_i is defined as $pc_i = (pc_i^l, pc_i^d, pc_i^r, pc_i^u)$, where $pc_i^l = c_i^l \gg (\omega - 1)$, $pc_i^d = c_i^d \gg (\omega - 1)$, $pc_i^r = c_i^r \gg (\omega - 1)$, $pc_i^u = c_i^u \gg (\omega - 1)$.*

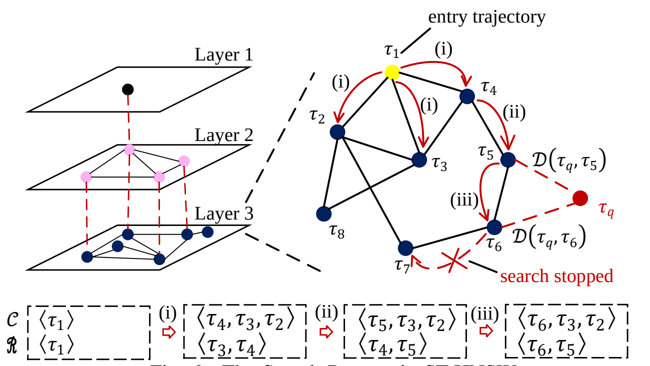
Given a geo-sequence *GS*, its MBR sequence $\mathtt{MS} = \langle c_1, ..., c_i, ..., c_{\frac{n}{\mathcal{L}}}\rangle$, we encode its h ($h \le \frac{n}{\mathcal{L}}$) MBRs using Definition 4.12. Then, we get the partition code for *GS*, i.e., $PC(GS) = \langle pc_1, ..., pc_i, ..., pc_h\rangle$. Based on that, we partition each batch B into Q partitions, i.e., $B = \{P_1, P_2, ..., P_Q\}$, such that for a data partition P_i ($1 \le i \le Q$), all its trajectories share the same code, i.e., $\forall \tau_a, \tau_b \in P_i$ & $\tau_a \ne \tau_b$, $PC(\tau_a.GS)$ = $PC(\tau_b.GS)$. Thus, we refer to the code of P_i as $PC(P_i)$. In Fig. 5(d), we repeatedly partition the largest until the specified partition number Q is achieved. After each partitioning, h is increased by 1, so h is referred to the partition depth.

5) Data partition pruning in Search:

Definition 15. (Trajectory-to-Partition Distance) *Given a data partition P with its partition code $PC(P) = \langle pc_1, ..., pc_d\rangle$, a trajectory geo-sequence GS of n points, as well as its segmented and encoded MBR sequence $\mathtt{MS} = \langle c_1, ..., c_{\frac{n}{\mathcal{L}}}\rangle$, where $pc_i = (pc_i^l, pc_i^d, pc_i^r, pc_i^u)$. Assuming the max length of the segmented MBR sequences in P is smax and the minimum length is smin, when $d{<}\min(\frac{n}{\mathcal{L}}, smin)$, we define the distance between geo-sequence GS and partition P*

$$\begin{aligned} TP(GS, P) = \sum_{i=1}^{d} \mathcal{W}(d(c_i, pc_i)) + (smax - d)\mathcal{W}(0), where \\ d(c_i, pc_i) = d(c_i, MBR(2^{\omega-1}pc_i^l, 2^{\omega-1}pc_i^d, 2^{\omega}pc_i^r, 2^{\omega}pc_i^u)) \end{aligned} \tag{16}$$

Lemma 5. *Given a query trajectory τ_q, a trajectory distance function $\mathcal{D}$ (i.e., DTW, LCSS, or EDR), and a set of the spatially partitioned data partitions on a data batch (i.e., $B = \{P_1, ..., P_Q\}$). Assume the length of the MBR segment of $\tau_q.GS$ is $\mathcal{L}$. Then $\forall P \in B, \forall \tau \in P$, we have $\mathcal{D}(\tau_q.GS, \tau.GS)$ $| = \mathcal{L} \cdot TP(\tau_q.GS, P)$.*


entry trajectory
Layer 1
Layer 2
Layer 3
τ_1
τ_2
τ_3
τ_4
τ_5
τ_6
τ_7
τ_8
τ_q
(i)
(ii)
(iii)
$\mathcal{D}(\tau_q, \tau_5)$
$\mathcal{D}(\tau_q, \tau_6)$
search stopped
$\mathcal{C}$ $\langle\tau_1\rangle$ (i) $\langle\tau_4, \tau_3, \tau_2\rangle$ (ii) $\langle\tau_5, \tau_3, \tau_2\rangle$ (iii) $\langle\tau_6, \tau_3, \tau_2\rangle$
$\mathcal{R}$ $\langle\tau_1\rangle$ $\langle\tau_3, \tau_4\rangle$ $\langle\tau_4, \tau_5\rangle$ $\langle\tau_6, \tau_5\rangle$
Fig. 6. The Search Process in ST-HNSW

Proof. Recall the partition code of a spatial partition P is $PC(P) = \langle pc_1, ..., pc_d \rangle$, where $pc_i = (pc_i^l, pc_i^d, pc_i^r, pc_i^u)$. $\forall \tau \in P$, $PC(\tau.GS) = PC(P)$. Let $\texttt{MS}_q$ and $\texttt{MS}$ denote the encoded MBR-sequence of $\tau_q.GS$ and $\tau.GS$, respectively. According to Definitions 12 and 14, $\forall i \in [1, d]$, assume $c_i = \texttt{MS}.c_i = MBR(c_i^l, c_i^d, c_i^r, c_i^u)$ and $c_i^q = \texttt{MS}_q.c_i$, we derive $[\ c_i^l, c_i^r\ ] \subseteq [\ 2^{\omega-1}pc_i^l, 2^{\omega}pc_i^r\ ]$ and $[\ c_i^d, c_i^u\ ] \subseteq [\ 2^{\omega-1} pc_i^d, 2^{\omega}pc_i^u\ ]$. Then $c_i \subseteq MBR(2^{\omega-1}pc_i^l,\ 2^{\omega-1}pc_i^d,\ 2^{\omega}pc_i^r,\ 2^{\omega}pc_i^u)$. Therefore, $d(c_i^q,\ MBR(2^{\omega-1}pc_i^l,\ 2^{\omega-1}pc_i^d,\ 2^{\omega}pc_i^r,\ 2^{\omega}pc_i^u)) \leq d(c_i^q,\ c_i)$. According to Definition 15, we have $d(c_i^q,\ MBR(2^{\omega-1}pc_i^l,\ 2^{\omega-1}pc_i^d,\ 2^{\omega}pc_i^r,\ 2^{\omega}pc_i^u)) = d(c_i^q,\ pc_i)$. Then $d(c_i^q,\ c_i) \geq d(c_i^q,\ pc_i)$. Assume the lengths of $\tau_q.\texttt{MS}$ and $\tau.\texttt{MS}$ are sq and s, respectively. When $\min(sq, s) = sq$, we have $\sum_{i=1}^{sq} \mathcal{W}(\ d(c_i^q,\ c_i)) + \sum_{i=sq+1}^{\min(sq+\lceil\delta/\mathcal{L}\rceil, s)} \mathcal{W}((c_i^q, c_i)) \mid = \sum_{i=1}^{d} \mathcal{W}(\ d(c_i^q,\ bc_i)) + (smax\text{-}d)\mathcal{W}(0)$, which means $EMSD(\texttt{MS}_q, \texttt{MS}) \mid = TP(\tau_q.GS, P)$. When $\min(sq, s) = s$, the same result are obtained. According to Lemmas 3 and 4, we have $\mathcal{D}(\tau_q.GS, \tau.GS) \mid = MSD(\tau_q.GS, \tau.GS) \mid = \mathcal{L}\ EMSD(\texttt{MS}_q, \texttt{MS}) \mid = \mathcal{L}\ TP(\tau_q.GS, P)$, that is $\mathcal{D}(\tau_q.GS, \tau.GS) \mid = \mathcal{L}TP(\tau_q.GS, P)$. □

VI. Local Index and Query Process

To accelerate the query process in each partition, we build the local ST-HNSW index.

A. Local ST-HNSW Index

1) HNSW Index: Hierarchical Navigable Small World [40] is a graph structure for vector similarity search. Specifically, HNSW uses a hierarchical structure to guide the search and employs a probabilistic insertion to balance search efficiency and quality. Overall, HNSW outperforms many classic methods, e.g., NSW [39], FLANN [22], VP-tree [8], and FALCONN [5]. This is the first proposal to deploy HNSW into semantic trajectory similarity search.

2) ST-HNSW Construction: The construction of a semantic-trajectory-oriented HNSW index (ST-HNSW) follows HNSW. In ST-HNSW, each trajectory is treated as a node and inserted into the hierarchical neighbor graph. When inserting a trajectory, there is a certain probability of entering a specific layer in the graph. During the insertion, the graph finds the M trajectories nearest to the trajectory being inserted, and the connections are established between them. Overall, the building and searching in ST-HNSW involve k nearest neighbor (kNN) search in a specific layer. As shown in Fig. 6, each layer in ST-HNSW represents a progressively larger NSW (Navigable Small World) [39], with the bottom layer being a complete NSW that contains all trajectories.

3) ST-HNSW Search: We detail how to perform search in a single layer. Fig. 6 shows an example of $k = 2$. Specifically, two priority queues are maintained: $\mathcal{C}$ that stores trajectories in ascending order of the distance to τ_q and $\mathcal{R}$ that stores query results in descending order of distance. We perform kNN as follows: (i) Starting from the entry trajectory τ_1, all neighbor trajectories τ_2, τ_3, and τ_4 of the current trajectory are traversed and enqueued into $\mathcal{C}$. The closer trajectories τ_3 and τ_4 are added to $\mathcal{R}$; (ii) The neighbor trajectory τ_5 of τ_4 is traversed and enqueued into $\mathcal{C}$ and the trajectory τ_5 is added to $\mathcal{R}$; (iii) The neighbor trajectory τ_6 of τ_5 is traversed and enqueued into $\mathcal{C}$ and the trajectory τ_6 is added to $\mathcal{R}$; (iv) The distance from the trajectory at the front of $\mathcal{C}$ to the target trajectory is not less than the distance from the trajectory at the front of $\mathcal{R}$ to the target τ_q, the query stops, and $\{\tau_5, \tau_6\}$ is returned.

B. Query Process

1) Two-Phase based Search: Given a query trajectory τ_q and a dataset $\mathcal{T}$, based on the semantic aware partitioning, we obtain P sorted data batches, i.e., $\mathcal{T} \rightarrow \mathcal{T}' = \{B_1, ..., B_P\}$ such that $TB(\tau_q, B_i) \leq TB(\tau_q, B_j)$ when $i \leq j$. Then, based on the spatial aware partitioning, we distribute each data batch $B_i \in \mathcal{T}'$ $(1 \leq i \leq P)$ into Q data partitions, i.e., $B_i = \{P_1^{(i)}, ..., P_Q^{(i)}\}$. Overall, there are $P \times Q$ data partitions.

Definition 16. (Bound Distance) *Given a query trajectory τ_q, a ranked batches $\mathcal{T}' = \{B_1, ..., B_P\}$ where $B_i = \{P_1^{(1)}, ..., P_Q^{(P)}\}$ $(1 \leq i \leq P)$ contains Q partitions, as well as the segment length $\mathcal{L}$. The bound distance between τ_q and $P_j^{(i)} (1 \leq j \leq Q, P_j^{(i)} \in B_i)$ is*

$$\begin{aligned} BD(\tau_q, P_j^{(i)}) = &\alpha \cdot \mathcal{W}(TB(\tau_q.SS, B_{i-1})) \\ &+ (1-\alpha) \cdot \mathcal{L} \cdot TP(\tau_q.GS, P_j^{(i)}) \end{aligned} \tag{17}$$

Lemma 6. *Given a query trajectory τ_q, a distance function $\mathcal{D}$, a ranked semantic-partitioned data batches $\mathcal{T}' = \{B_1, ..., B_P\}$ where each data batch $B_i = \{P_1^{(i)}, ..., P_Q^{(i)}\}$ contains Q data partitions, we then have $\forall \tau \in P_k^{(i)} (1 \leq k \leq Q) \& \tau \notin \bigcup_{j<i} B_j$, $\mathcal{D}(\tau_q, \tau) \geq BD(\tau_q, P_k^{(i)})$.*

Proof. The proof is straightforward by Lemmas 2 and 5. □

Based on Lemma 6, we can use *BD* to prune unnecessary data batches and data partitions. Specifically, during similarity search, if $BD(\tau_q, P_k^{(i)})$ has exceeded the distance between τ_q and the farthest trajectory in returned trajectories on searched data partitions, the remaining data partitions in B_i are pruned. Note that, in a local data partition, we directly use ST-HNSW search (Section 6.1.3).

2) Optimized ST-HNSW based Search: Although the two-phase based search enables pruning many unnecessary data partitions, in each local partition, it directly uses ST-HNSW search. However, ST-HNSW computes the distance between two trajectories in both construction and search phases, which cost cannot be ignored. Therefore, we extend HNSW into weighted HNSW and deploy triangle inequality for optimization. Specifically, when inserting trajectories into ST-HNSW, the distances $\mathcal{D}(\cdot,\cdot)$ between trajectories are saved as the edge weight of the graph.

Recall Equation 1, if $\mathcal{D}^{(G)}(\cdot,\cdot)$ and $\mathcal{D}^{(S)}(\cdot,\cdot)$ satisfy triangle inequality, $\mathcal{D}(\cdot,\cdot)$ satisfies triangle inequality. Thus, during the search in each layer of ST-HNSW, we combine weighted edges and triangle inequality to prune inter-trajectory computations.

Lemma 7. *During the process of finding the top k nearest neighbors of the query trajectory τ_q, assuming the current search node is τ_c (i.e., the first trajectory in the candidate queue $\mathcal{C}$ for traversing), τ_n is a neighbor of τ_c and $\mathcal{R}$ is the set of current top k nearest neighbors of τ_q. When $|\mathcal{D}(\tau_c,\tau_n)-\mathcal{D}(\tau_c,\tau_q)|\ | = \max_{\tau\in\mathcal{R}}\mathcal{D}(\tau,\tau_q)$, then $\tau_n \notin \mathcal{R}$.*

Proof. According to triangle inequality, $\mathcal{D}(\tau_n,\tau_q)\ | = |\mathcal{D}(\tau_c,\tau_n)-\mathcal{D}(\tau_c,\tau_q)|$. So $\mathcal{D}(\tau_n,\tau_q)\ | = \max_{\tau\in\mathcal{R}}\mathcal{D}(\tau,\tau_q)$. Then $\tau_n \notin \mathcal{R}$. □

Lemma 8. *During the process of finding the top k nearest neighbors of the query trajectory τ_q, assuming the current search node is τ_c (i.e., the first trajectory in the candidate queue $\mathcal{C}$ for traversing), τ_n is a neighbor of τ_c and $\mathcal{R}$ is the set of current top k nearest neighbors of τ_q. When $BD(\tau_c,\tau_n)\ | = \mathcal{D}(\tau_c,\tau_q)$ and $BD(\tau_c,\tau_n)-\mathcal{D}(\tau_c,\tau_q)\ | = \max_{\tau\in\mathcal{R}}\mathcal{D}(\tau,\tau_q)$, then $\tau_n \notin \mathcal{R}$.*

Proof. According Lemma 6, $\mathcal{D}(\tau_c,\ \tau_n)\ | = BD(\tau_c,\tau_n)$. Assume that $\mathcal{D}(\tau_c,\tau_n) = BD(\tau_c,\tau_n)+\Delta\mathcal{D}$, $\Delta\mathcal{D}\geq 0$, according to triangle inequality, we have $\mathcal{D}(\ \tau_c,\ \tau_q)+\mathcal{D}(\ \tau_n,\ \tau_q)\ | = \mathcal{D}(\ \tau_c,\ \tau_n)$. Then $\mathcal{D}(\ \tau_n,\ \tau_q)\ | = \mathcal{D}(\ \tau_c,\ \tau_n)-\mathcal{D}(\ \tau_c,\ \tau_q)$, i.e., $\mathcal{D}(\ \tau_n,\ \tau_q)\ | = BD(\ \tau_c,\ \tau_n)+\Delta\mathcal{D}-\mathcal{D}(\tau_c,\tau_q)$. Therefore $\mathcal{D}(\ \tau_n,\ \tau_q)\ | = BD(\tau_c,\tau_n)-\mathcal{D}(\tau_c,\tau_q)$. Then $\mathcal{D}(\tau_n,\tau_q)\ | = \max_{\tau\in\mathcal{R}}\mathcal{D}(\ \tau,\tau_q)$. So we derive $\tau_n \notin \mathcal{R}$. □

In ST-HNSW, a fixed entry may increase search steps because the distance between the entry point and its neighbors is far. We use a grid index to find a suitable entry point in advance. Specifically, we divide the space into grids, where each grid contains all trajectories within that region. During searches, we use the grid index to find the adjacent grid regions to the query trajectory, and then select a suitable entry point within that region. This method reduces the number of search steps and improves ST-HNSW efficiency.

VII. Experiment

We use research questions below to guide experiments:

RQ1. How does the semantic modeling measure perform?
RQ2. How does the distributed framework perform?
RQ3. How useful of the proposed pruning technology?
RQ4. How robust is the framework for hyperparameters?
RQ5. Can this framework support other similarity queries?

A. Setup

Datasets. We use two real datasets, T-Drive [68], [69] and GeoLife [78]–[80], as well as a synthetic dataset, Brinkhoff [9]. Specifically, i) T-drive contains taxi trajectories of Beijing, during a period of 7 days; ii) GeoLife contains people trajectories that involve various transportation modes, during a period of more than three years; and iii) Brinkhoff is generated on real road networks of San Joaquin. Due to the long duration of the original trajectories, we divided the trajectories into multiple trajectories at regular time intervals.

TABLE III
Statistics of the Datasets Used

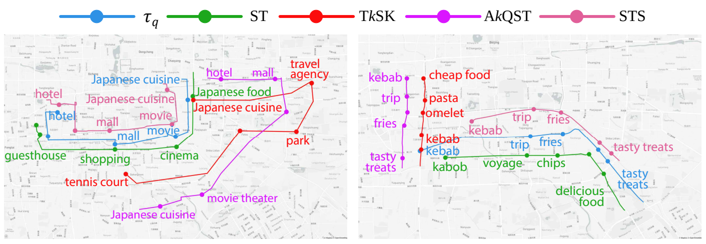
Attributes	T-drive	Brinkhoff	GeoLife
#trajectories	3,347,280	3,478,080	1,886,640
#GPS points	152,358,760	176,997,340	497,539,560
#keywords	133,404,220	155,205,860	436,085,220
#descriptions	77,256,139	70,798,936	257,178,733
Disk usage	179.24 GB	377.91 GB	115.16 GB

TABLE IV
Parameter Ranges and Default Values

Parameters	Range
Spatial-Semantic weight α	0.25, **0.50**, 0.75, 1.00
Segment length $\mathcal{L}$ of geo-sequence	2, 4, **6**, 8
Grid length δ_g ($\times 10^3$ m) on T-drive	2.25, **4.5**, 6.75, 9
Grid length δ_g ($\times 10^4$) on Brinkhoff	1.5, **3**, 4.5, 6
Grid length δ_g ($\times 10^3$ m) on GeoLife	2.5, **5**, 7.5, 10
Ratio P/Q	0.5, **1**, 1.5, 2
Search result size k	5, 10, 20, **50**
# trajectories O_r (%)	25, 50, 75, **100**
# parallelism $N = P \times Q$	48, 72, 96, **120**

Following previous works [16], [75]–[77], we augment raw trajectory datasets with semantics. By using the coordinates of the check-in data from Weibo and a collection of real-life tweets [49], we associated them with the corresponding coordinates of the above datasets. Table III summarizes the statistics of the datasets used.

As there are no manually annotated ground truths for trajectory similarity tasks, existing effectiveness studies are categorized into i) Downsampling [35]. This involves taking a real trajectory and applying various downsamplings, resulting in a set of trajectories that can be utilized as query trajectories and ground truth. ii) Detouring [17], [30]. It is designed for creating ground truth data under road networks. It entails selecting a (sub)trajectory on the road network as the query trajectory and generating detour trajectories with the same starting and ending points to serve as ground truth. iii) Measure-based [23], [27], [64]. They utilize trajectory distance measures to perform a search, with the outcomes obtained serving as the ground truth. Both downsampling and detouring entail the synthetic trajectories, rather than using the original trajectories as the query and ground truth. Consequently, they are unable to properly represent the distribution of trajectories. Thus, we adopt the measure-based methods, which are widely



(a) T-Drive (b) GeoLife

Fig. 7. Expressiveness of Different Semantic-aware Distance Measures

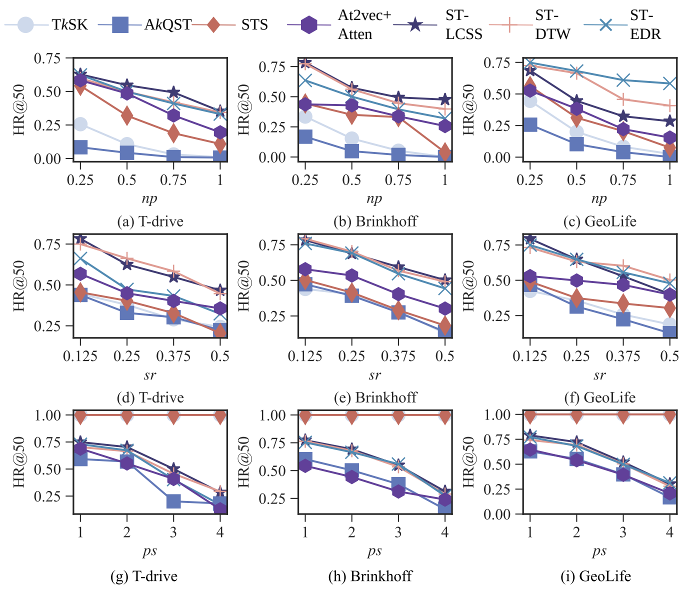


Fig. 8. Trajectory Measure Robustness Evaluation

used in trajectory measure quality studies.

Parameters. Table IV lists parameters. Specifically, α balances the importance of spatial versus semantic similarity; $\mathcal{L}$ denotes the segment length of segmented MBR sequences; δ_g denotes the grid length in the grid optimization for ST-HNSW; P and Q are the numbers of batches and partitions; P/Q balances the partition numbers of semantic and spatial partitions; k is used in the search; O_r denotes the percentage of moving objects w.r.t. all objects, and N denotes the number of data partitions after two-layer partitioning. The approximate guarantee ϵ is set to 0 as default.

Baselines. We propose a new semantic modeling method, based on which, we build the distributed query framework. Therefore, we evaluate our methods from two dimensions, i.e., A) semantic modeling and B) search processing.

- *A) Semantic modeling comparison.* We compare the proposed semantic trajectory representation manner termed ST (Eq. 1, supported by LCSS, DTW, and EDR) with existing semantic modeling measures in terms of effectiveness and robustness. Specifically, we compare ST with i) TkST [77] that combines global keyword set similarity and spatial similarly, ii) AkQST [75] that combines individual keyword similarity and spatial similarity, iii) STS [16] that combines GPS points with keyword annotations, and iv) At2vec+Atten [36] that use neural networks to capture spatial-semantic similarities.
- *B) Search comparison.* We refer to our method as the DSTSS (Distributed Semantic Trajectory Similarity Search). To evaluate the usefulness of the local optimization on ST-HNSW, we remove that from DSTSS, resulting in a variant called DSTSS w/o LO. Then, we compare these two methods with i) ID-Force (Section 3.4) that uses HashPartitioner and local HNSW-ST index to achieve distributed searches; ii) Dis-PSTSJ (Section 3.4) that is a distributed implementation of PSTSJ [16] that uses invert indexes for pruning; (iii) DITA-S (Section 3.4) that is an extension to semantic trajectory search based on DITA [51]; iv) a distributed implementation of At2vec+Atten [36], and v) LSTS-Join [48] that is the one and only existing work that was designed for distributed semantic trajectory similarity joins, which builds a hierarchical index on Spark.

Evaluation metrics. We use five metrics. i) **HR@k**. It denotes the proportion of results that are true positives. The ground truth is the search results via brute-force search. ii) **Running time**. It refers to index construction or search process. iii) **Memory usage.** It denotes the memory resource occupation during searches. iv) C_m. It denotes the ratio of distance calculations after pruning to the calculations in brute-force search. v) ***SD.*** It denotes the standard deviation of the number of trajectories in each partition. All the experiments were conducted on a cluster consisting of 6 nodes. Each node is equipped with two 12-core processors (Intel Xeon E5-2620 v3 2.40 GHz), 64GB RAM, Ubuntu 14.04.3, and Spark 3.1.1.

B. Semantic Modeling Study (RQ1)

To verify the effectiveness of different semantic modeling methods, we select dense areas from two real-life datasets, respectively, and visualize the most similar (i.e., Top-1) trajectory to a query trajectory. The results are shown in Fig. 7, where the blue one denotes the query trajectory τ_q. Note that ST returns the same trajectory when supported by LCSS, DTW, and EDR. At2Vec+Atten returns the same trajectory as ST. Overall, we have the following observations: (i) TkSK returns a semantic-similar while spatial faraway trajectory, as TkSK mainly considers the global semantic similarities. (ii) Although AkQST considers both spatial and semantic aspects, it relies on keyword representation, thus it returns a trajectory that significantly deviates from the query trajectory. (iii) STS cannot identify semantically synonymous keywords in trajectories, as STS is based on raw representations. (iv) The search performance of At2vec+Atten is unstable. (v) Our proposed ST effectively finds the most spatial-semantic similar trajectory in the two real-life datasets. Note that utilizing LCSS, EDR, and DTW in ST typically yields identical returns, since these methods compute trajectory similarity via point-to-point matching, as detailed in Section 3. It is precisely because of their stability that they have been widely used in trajectory similarity studies, which indirectly proves the rationality of the framework's support for these metrics.

Following previous studies [28], [52], we use three noising parameters, i.e., noising-probability *np*, noising-sampling *sr*, and noising-shifting *sp*, to evaluate the robustness. These parameters are used to disturbance datasets [28]. Specifically, we compare the query results returned from the raw dataset and those returned from the dataset after perturbation processing. Here, we use the HR@50 metric. The results are shown in Fig. 8. First, for all types of perturbations, HR@50 decreases with the increase of the perturbation degree. Second, for noise and sampling perturbations, our similarity measure is more robust than the other measures, as we use vector representations while the other measures use raw keyword forms. Finally, for the point shifting perturbation, TkSK and STS are not affected, while our measure shows better robustness.

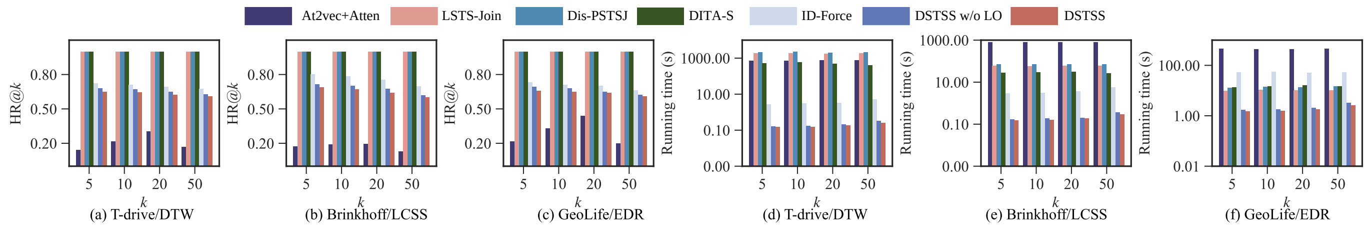


Fig. 9. Effectiveness and Efficiency Evaluation vs. k

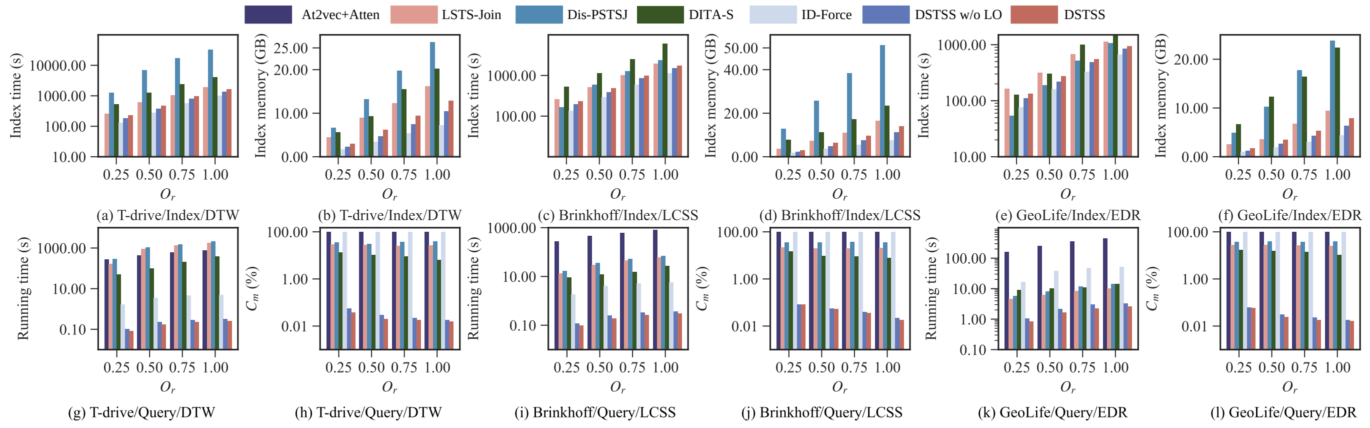


Fig. 10. Scalability Evaluation vs. Data sizes O_r

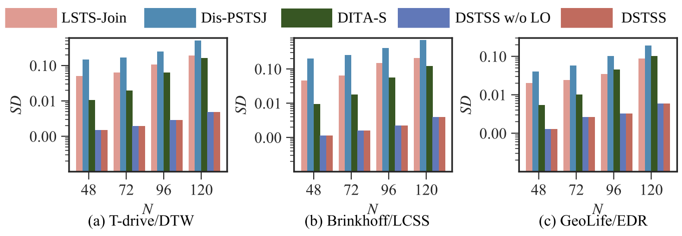


Fig. 11. Workload Evaluation

C. Search Processing Study (RQ2 & RQ3)

First, we evaluate the search effectiveness, as the adopted local ST-HNSW search is based on approximation processing. We compare the proposed two methods, i.e., `DSTSS` and `DSTSS w/o LO`, with `ID-Force`, `Dis-PSTSJ`, `DITA-S`, `At2vec+Atten`, and `LSTS-Join`, in terms of HR@k at different k values. The results are shown in Figs. 9(a)–(c). Compared with the exact search methods (i.e., `Dis-PSTSJ`, `DITA-S`, and `LSTS-Join`), our methods show a lower hit ratio. The sole reduction in hit ratio can be attributed to the ST-HNSW based search. Compared with the learning-based approximate search methods (i.e., `At2vec+Atten`), our methods show better search quality.

Then, we study the search efficiency, corresponding to the above effectiveness evaluation. The results are shown in Figs. 9(d)–(f). As observed, our methods outperform baselines up to 1–3 orders of magnitude, due to the effective computation-aware partitioning. With the increase of k, the query time of baselines increases while the query time of our methods keeps stable, which shows the superiority of our methods. Another observation is, `DSTSS` continuously performs better than `DSTSS w/o LO`, proving the usefulness of the local optimization techniques on ST-HNSW. Our methods trade a slight effectiveness drop for much better efficiency, i.e., 1–3 orders of magnitude efficiency improvements.

Next, we study the scalability performance under different data sizes O_r. The results are shown in Fig. 10, where we study both index construction and query phases. In the index construction phase, i.e., Figs. 10(a)–(f), `Dis-PSTSJ` and `LSTS-Join` require more construction time and memory usage because they need to build a large inverted index of keywords, resulting in a lot of overlapping between inverted files. Second, when O_r increases, both index construction time and memory usage increase, as there are more trajectories. Note that `At2Vec+Atten` requires massive model training time (e.g., 8.55 hours on T-drive), which is omitted here. In the query phase, i.e., Figs. 10(g)–(l), `At2vec+Atten` requires the max query time because it requires tensor calculation. Second, `Dis-PSTSJ`, `DITA-S`, and `LSTS-Join` also require a long query time because they need to traverse and calculate similarity scores for a large set of trajectory candidates. Third, when O_r increases, the query time gradually increases. Finally, `DSTSS` and `DSTSS w/o LO` show significantly less query time by reducing a large number of unnecessary similarity calculations. Last but not the least, Figs. 10(h), 10(j), 10(l) show the corresponding pruning process, where our methods show a more powerful pruning ability than baselines.

Finally, we evaluate the workload by computing the computational standard deviation (*SD*). The results are shown in Fig. 11. As expected, our partition strategy achieves much better balancing, i.e., 1–2 orders of magnitude gains.

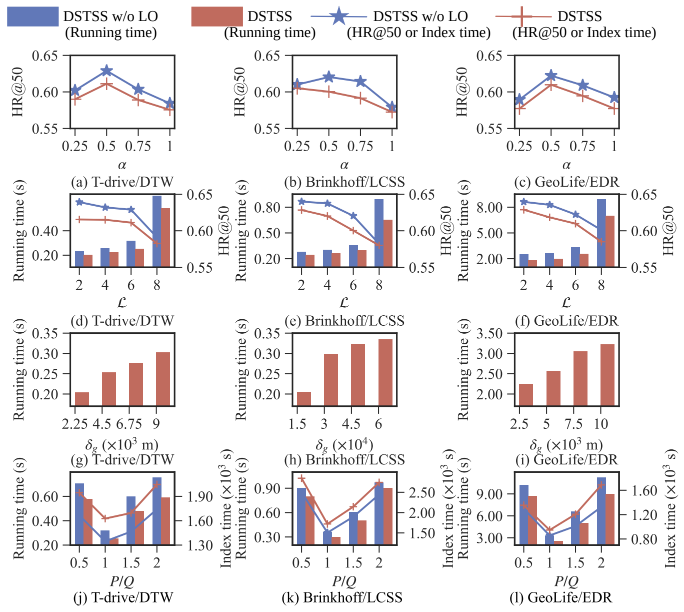


Fig. 12. Parameter Sensitivity Study

Algorithm 1: Threshold-based Similarity Search

```
Input: a set of queries $Q$, a distance function $\mathcal{D}$, the
       distance threshold $\theta$, and a two-phase partitioned
       trajectory dataset $\mathcal{T}'' = Map(Int, List(List()))$ that
       stores data batches and data partitions
1  $R_\tau(Int, List()) \leftarrow Null$ // a map to store the results for each
     query
2  for each query $\tau_q \in Q$ do
3      $R_\tau[\tau_q.ID] \leftarrow$ List()
4      for each batch $B \in \mathcal{T}''$ do
5          for each partition $P \in B$ do
6              if $\theta \mid = BD(\tau_q, P)$ then
7                  // Definition 16 and Lemma 6
8                  LocalRes $\leftarrow$ Threshold-based ST-HNSW
                     Search
9                  for $\tau \in LocalRes$ do
10                     $R_\tau[\tau_q.ID]$.add($\tau$)
11         if $\alpha \cdot TB(\tau_q, B) \mid = \theta$ then
12             // Lemma 2
13             break
```

D. *Parameter Sensitivity Study (RQ4)*

We also explore the effects of spatial-semantic weight α, segment length $\mathcal{L}$, grid length δ_g, and P/Q.

Effect of α. Figs. 12(a)–12(c) report the HR@50 of `DSTSS` and `DSTSS w/o LO` by varying α. As α increases, HR@50 first increases and then decreases. When α is small, the effect of the semantic similarity is small; when α is large, the effect of the spatial similarity is small. Thus, when α takes an appropriate weight between 0 and 1, HR@50 reaches its peak.

Effect of $\mathcal{L}$**.** Figs. 12(d)–12(f) show the effect of varying $\mathcal{L}$. As $\mathcal{L}$ increases, the length of the segmented geographic sequence becomes smaller, the lower bound of the encoded geographical sequence distance becomes looser, the pruning effect of spatial partitioning becomes worse, and the load becomes more unbalanced, resulting in the decline of HR@50 and the increase of query time.

Effect of δ_g**.** Figs. 12(g)–12(i) plot the results by varying δ_g. As δ_g increases, the query time slightly increases. When δ_g is very large, the effect of grid optimization on HNSW becomes weak, because it is difficult for the grid to locate entry points close to the query.

Effect of P/Q**.** Figs. 12(j)–12(l) plot the results by varying P/Q. When one of P or Q is very small while the other is large, both running time and index time will increase. This is because a partitioner with a smaller number of partitions has reduced pruning capabilities and the load becomes more imbalanced, resulting in performance degradation. When $P = Q$, we yield the best performance.

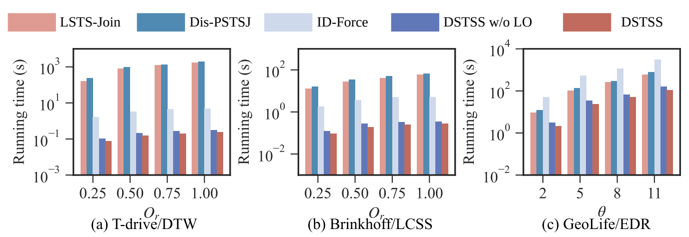
E. *Threshold-based Similarity Search (RQ5)*

In this subsection, we show that our framework supports another important trajectory query, i.e., threshold-based similarity search.

Definition 17. (Threshold-based Similarity Search, TSS) *Given a set of trajectories $\mathcal{T}$, a query trajectory τ_q, a distance function $\mathcal{D}(\cdot,\cdot)$, and a distance threshold θ, TSS returns a set of trajectories $\mathcal{R} \subset \mathcal{T}$, where $\forall \tau \in \mathcal{R}, \theta \mid = \mathcal{D}(\tau, \tau_q)$.*

Specifically, by replacing the input parameters k and ϵ in Algorithm 1 with θ, and substituting $(1+\epsilon)\cdot bsfDist$ with θ in lines 7 and 13, our method can be seamlessly extended to support the above TSS problem. In addition, in the Local Search process, the search result $\mathcal{R}$ is transformed into an unbounded set, and the insertion condition is adjusted to satisfy the threshold θ. The pseudo-code for TSS is shown in Algorithm 2. It takes a set Q of query trajectories, a distance function $\mathcal{D}$, the distance threshold θ, and a trajectory dataset $\mathcal{T}'' = Map(Int, List(\mathcal{L}ist()))$ as input. Lines 2–3 initialize a result list for each query. Then, lines 4–7 calculate the bound distance *BD* between a query trajectory and a data partition. If the bound distance does not violate Lemma 6, we dispatch this query into this partition and perform a threshold-based ST-HNSW search (lines 8–10). Simultaneously, if the *TB* distance between the query trajectory to the current data batch violates Lemma 2, we directly prune all data partitions from this data batch (lines 11–13).

We also perform a set of experiments by varying the parameters θ and O_r. The results are shown in Fig. 13. As observed, our methods outperform baselines up to 1–3 orders of magnitude, due to the similar computation and pruning process with kNN search. With the increase of O_r, the running time of all methods increases. However, as θ increases, the running time grows rapidly because it requires evaluating and returning a large number of trajectories. This leads to a deterioration in the pruning effect of all methods, while our method, although experiencing a performance decline, still maintains great superiority over the baselines.



(a) T-drive/DTW (b) Brinkhoff/LCSS (c) GeoLife/EDR

Fig. 13. Threshold-based Similarity Search Evaluation

VIII. Conclusions

This paper introduces a distributed framework for semantic trajectory similarity search. We propose a new representation method for robust semantic trajectory modeling. We also present a computation-aware two-phase partitioning, allowing efficient pruning of most data partitions during query processing. Within each partition, we incorporate the ST-HNSW index to further enhance speed. Experiments using both real and synthetic datasets demonstrate the superior performance of our methods. In the future, it would be intriguing to include additional types of semantics, such as geo-photos and videos, or incorporate a temporal perspective into semantic analyses.

References

[1] https://support.google.com/mymaps.
[2] https://foursquare.com.
[3] https://wanderlog.com.
[4] https://www.rome2rio.com.
[5] A. Andoni and I. P. Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In R. A. Servedio and R. Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 793–801. ACM, 2015.
[6] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguistics*, 5:135–146, 2017.
[7] T. Boonchoo, X. Ao, and Q. He. Multi-aspect embedding for attribute-aware trajectories. *Symmetry*, 11(9):1149, 2019.
[8] L. Boytsov and B. Naidan. Learning to prune in metric and non-metric spaces. In C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 1574–1582, 2013.
[9] T. Brinkhoff. A framework for generating network-based moving objects. *GeoInformatica*, 6(2):153–180, 2002.
[10] C. Cai and D. Lin. Find another me across the world - large-scale semantic trajectory analysis using spark. *IEEE Trans. Knowl. Data Eng.*, 35(9):8905–8918, 2023.
[11] H. Cao, F. Xu, J. Sankaranarayanan, Y. Li, and H. Samet. Habit2vec: Trajectory semantic embedding for living pattern recognition in population. *IEEE Trans. Mob. Comput.*, 19(5):1096–1108, 2020.
[12] V. P. Chakka, A. Everspaugh, and J. M. Patel. Indexing large trajectory data sets with SETI. In *First Biennial Conference on Innovative Data Systems Research, CIDR 2003, Asilomar, CA, USA, January 5-8, 2003, Online Proceedings*. www.cidrdb.org, 2003.
[13] M. Chatzakis, P. Fatourou, E. Kosmas, T. Palpanas, and B. Peng. Odyssey: A journey in the land of distributed data series similarity search. *Proc. VLDB Endow.*, 16:1140–1153, 2023.
[14] L. Chen, Y. Gao, Z. Fang, X. Miao, C. S. Jensen, and C. Guo. Real-time distributed co-movement pattern detection on streaming trajectories. *Proc. VLDB Endow.*, 12(10):1208–1220, 2019.
[15] L. Chen, M. T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In F. Özcan, editor, *Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005*, pages 491–502. ACM, 2005.
[16] L. Chen, S. Shang, C. S. Jensen, B. Yao, and P. Kalnis. Parallel semantic trajectory similarity join. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*, pages 997–1008. IEEE, 2020.
[17] Y. Chen, X. Li, G. Cong, Z. Bao, C. Long, Y. Liu, A. K. Chandran, and R. Ellison. Robust road network representation learning: When traffic patterns meet traveling semantics. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, pages 211–220, 2021.
[18] G. Cong, H. Lu, B. C. Ooi, D. Zhang, and M. Zhang. Efficient spatial keyword search in trajectory databases. *CoRR*, abs/1205.2880, 2012.
[19] J. Dai, B. Yang, C. Guo, and Z. Ding. Personalized route recommendation using big trajectory data. In J. Gehrke, W. Lehner, K. Shim, S. K. Cha, and G. M. Lohman, editors, *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, pages 543–554. IEEE Computer Society, 2015.
[20] D. R. de Almeida, C. de Souza Baptista, and F. G. de Andrade. Similarity search on semantic trajectories using text processing. *ISPRS Int. J. Geo Inf.*, 11(7):412, 2022.
[21] J. Ding, J. Fang, Z. Zhang, P. Zhao, J. Xu, and L. Zhao. Real-time trajectory similarity processing using longest common subsequence. In *21st IEEE International Conference on High Performance Computing and Communications; 17th IEEE International Conference on Smart City; 5th IEEE International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2019, Zhangjiajie, China, August 10-12, 2019*, pages 1398–1405, 2019.
[22] Y. Fang, R. Cheng, W. Tang, S. Maniu, and X. S. Yang. Scalable algorithms for nearest-neighbor joins on big trajectory data. In *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016*, pages 1528–1529. IEEE Computer Society, 2016.
[23] Z. Fang, Y. Du, X. Zhu, D. Hu, L. Chen, Y. Gao, and C. S. Jensen. Spatio-temporal trajectory similarity learning in road networks. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 347–356, 2022.
[24] E. Fredkin. Trie memory. *Commun. ACM*, 3:490–499, 1960.
[25] C. Gao, Z. Zhang, C. Huang, H. Yin, Q. Yang, and J. Shao. Semantic trajectory representation and retrieval via hierarchical embedding. *Inf. Sci.*, 538:176–192, 2020.
[26] A. Guttman. R-trees: A dynamic index structure for spatial searching. In B. Yormark, editor, *SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, USA, June 18-21, 1984*, pages 47–57. ACM Press, 1984.
[27] P. Han, J. Wang, D. Yao, S. Shang, and X. Zhang. A graph-based approach for trajectory similarity computation in spatial networks. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pages 556–564, 2021.
[28] D. Hu, L. Chen, H. Fang, Z. Fang, T. Li, and Y. Gao. Spatio-temporal trajectory similarity measures: A comprehensive survey and quantitative study. *CoRR*, abs/2303.05012, 2023.
[29] H. Issa and M. L. Damiani. Efficient access to temporally overlaying spatial and textual trajectories. In C. Chow, P. P. Jayaraman, and W. Wu, editors, *IEEE 17th International Conference on Mobile Data Management, MDM 2016, Porto, Portugal, June 13-16, 2016*, pages 262–271. IEEE Computer Society, 2016.
[30] J. Jiang, D. Pan, H. Ren, X. Jiang, C. Li, and J. Wang. Self-supervised trajectory representation learning with temporal regularities and travel semantics. In *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*, pages 843–855, 2023.
[31] S. Kanda, K. Takeuchi, K. Fujii, and Y. Tabei. Succinct trit-array trie for scalable trajectory similarity search. In *SIGSPATIAL '20: 28th International Conference on Advances in Geographic Information Systems, Seattle, WA, USA, November 3-6, 2020*, pages 518–529, 2020.
[32] E. J. Keogh. Exact indexing of dynamic time warping. In *Proceedings of 28th International Conference on Very Large Data Bases, VLDB 2002, Hong Kong, August 20-23, 2002*, pages 406–417. Morgan Kaufmann, 2002.
[33] E. Leal, L. Gruenwald, J. Zhang, and S. You. Tksimgpu: A parallel top-k trajectory similarity query processing algorithm for gpgpus. In *2015 IEEE International Conference on Big Data (IEEE BigData 2015), Santa Clara, CA, USA, October 29 - November 1, 2015*, pages 461–469, 2015.

[34] R. Li, H. He, R. Wang, S. Ruan, T. He, J. Bao, J. Zhang, L. Hong, and Y. Zheng. Trajmesa: A distributed nosql-based trajectory data management system. *IEEE Trans. Knowl. Data Eng.*, 35(1):1013–1027, 2023.

[35] X. Li, K. Zhao, G. Cong, C. S. Jensen, and W. Wei. Deep representation learning for trajectory similarity computation. In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*, pages 617–628, 2018.

[36] A. Liu, Y. Zhang, X. Zhang, G. Liu, Y. Zhang, Z. Li, L. Zhao, Q. Li, and X. Zhou. Representation learning with multi-level attention for activity trajectory similarity computation. *IEEE Trans. Knowl. Data Eng.*, 34(5):2387–2400, 2022.

[37] K. Liu, B. Yang, S. Shang, Y. Li, and Z. Ding. MOIR/UOTS: trip recommendation with user oriented trajectory search. In *2013 IEEE 14th International Conference on Mobile Data Management, Milan, Italy, June 3-6, 2013 - Volume 1*, pages 335–337. IEEE Computer Society, 2013.

[38] Z. Luo, L. Li, M. Zhang, W. Hua, Y. Xu, and X. Zhou. Diversified top-k route planning in road network. *Proc. VLDB Endow.*, 15(11):3199–3212, 2022.

[39] Y. Malkov, A. Ponomarenko, A. Logvinov, and V. Krylov. Approximate nearest neighbor algorithm based on navigable small world graphs. *Inf. Syst.*, 45:61–68, 2014.

[40] Y. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(4):824–836, 2020.

[41] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis. Dremel: Interactive analysis of web-scale datasets. *Proc. VLDB Endow.*, 3(1):330–339, 2010.

[42] M. A. Nascimento and J. R. O. Silva. Towards historical r-trees. In K. M. George and G. B. Lamont, editors, *Proceedings of the 1998 ACM symposium on Applied Computing, SAC'98, Atlanta, GA, USA, February 27 - March 1, 1998*, pages 235–240. ACM, 1998.

[43] T. Palpanas. The parallel and distributed future of data series mining. *2017 International Conference on High Performance Computing & Simulation (HPCS)*, pages 916–920, 2017.

[44] C. Parent, S. Spaccapietra, C. Renso, G. L. Andrienko, N. V. Andrienko, V. Bogorny, M. L. Damiani, A. Gkoulalas-Divanis, J. A. F. de Macêdo, N. Pelekis, Y. Theodoridis, and Z. Yan. Semantic trajectories modeling and analysis. *ACM Comput. Surv.*, 45(4):42:1–42:32, 2013.

[45] D. Pfoser, C. S. Jensen, and Y. Theodoridis. Novel approaches to the indexing of moving object trajectories. In A. E. Abbadi, M. L. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, and K. Whang, editors, *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 395–406. Morgan Kaufmann, 2000.

[46] J. Qin, L. Ma, and Q. Liu. DFTHR: A distributed framework for trajectory similarity query based on hbase and redis. *Inf.*, 10(2):77, 2019.

[47] J. B. Rocha-Junior, O. Gkorgkas, S. Jonassen, and K. Nørvåg. Efficient processing of top-k spatial keyword queries. In D. Pfoser, Y. Tao, K. Mouratidis, M. A. Nascimento, M. F. Mokbel, S. Shekhar, and Y. Huang, editors, *Advances in Spatial and Temporal Databases - 12th International Symposium, SSTD 2011, Minneapolis, MN, USA, August 24-26, 2011, Proceedings*, volume 6849 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2011.

[48] ruijie tian, H. Zhai, J. Li, weishi zhang, fei wang, and H. Liu. A Distributed Framework for Large-Scale Semantic Trajectory Similarity Join. 8 2022.

[49] T. Sahni, C. Chandak, N. R. Chedeti, and M. Singh. Efficient twitter sentiment classification using subjective distant supervision. In *9th International Conference on Communication Systems and Networks, COMSNETS 2017, Bengaluru, India, January 4-8, 2017*, pages 548–553. IEEE, 2017.

[50] S. Shang, R. Ding, B. Yuan, K. Xie, K. Zheng, and P. Kalnis. User oriented trajectory search for trip recommendation. In E. A. Rundensteiner, V. Markl, I. Manolescu, S. Amer-Yahia, F. Naumann, and I. Ari, editors, *15th International Conference on Extending Database Technology, EDBT '12, Berlin, Germany, March 27-30, 2012, Proceedings*, pages 156–167. ACM, 2012.

[51] Z. Shang, G. Li, and Z. Bao. DITA: distributed in-memory trajectory analytics. In G. Das, C. M. Jermaine, and P. A. Bernstein, editors, *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pages 725–740. ACM, 2018.

[52] H. Su, S. Liu, B. Zheng, X. Zhou, and K. Zheng. A survey of trajectory distance measures and performance evaluation. *VLDB J.*, 29(1):3–32, 2020.

[53] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112, 2014.

[54] Y. Tao and D. Papadias. Mv3r-tree: A spatio-temporal access method for timestamp and interval queries. In P. M. G. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamohanarao, and R. T. Snodgrass, editors, *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*, pages 431–440. Morgan Kaufmann, 2001.

[55] F. Valdés and R. H. Güting. A framework for efficient multi-attribute movement data analysis. *VLDB J.*, 28(4):427–449, 2019.

[56] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[57] M. Vlachos, D. Gunopulos, and G. Kollios. Discovering similar multidimensional trajectories. In R. Agrawal and K. R. Dittrich, editors, *Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, February 26 - March 1, 2002*, pages 673–684. IEEE Computer Society, 2002.

[58] S. Wang, Z. Bao, J. S. Culpepper, and G. Cong. A survey on trajectory data management, analytics, and learning. *ACM Comput. Surv.*, 54(2):39:1–39:36, 2022.

[59] S. Wang, Z. Bao, J. S. Culpepper, T. Sellis, and X. Qin. Fast large-scale trajectory clustering. *Proc. VLDB Endow.*, 13(1):29–42, 2019.

[60] D. Xie, F. Li, and J. M. Phillips. Distributed trajectory similarity search. *Proc. VLDB Endow.*, 10(11):1478–1489, 2017.

[61] D. E. Yagoubi, R. Akbarinia, F. Masseglia, and T. Palpanas. Dpisax: Massively distributed partitioned isax. *2017 IEEE International Conference on Data Mining (ICDM)*, pages 1135–1140, 2017.

[62] D. E. Yagoubi, R. Akbarinia, F. Masseglia, and T. Palpanas. Massively distributed time series indexing and querying. *IEEE Trans. Knowl. Data Eng.*, 32(1):108–120, 2020.

[63] C. Yang, L. Chen, H. Wang, and S. Shang. Towards efficient selection of activity trajectories based on diversity and coverage. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 689–696. AAAI Press, 2021.

[64] P. Yang, H. Wang, Y. Zhang, L. Qin, W. Zhang, and X. Lin. T3S: effective representation learning for trajectory similarity computation. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*, pages 2183–2188, 2021.

[65] D. Yao, H. Hu, L. Du, G. Cong, S. Han, and J. Bi. Trajgat: A graph-based long-term dependency modeling approach for trajectory similarity computation. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 2275–2285, 2022.

[66] B. Yi, H. V. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In S. D. Urban and E. Bertino, editors, *Proceedings of the Fourteenth International Conference on Data Engineering, Orlando, Florida, USA, February 23-27, 1998*, pages 201–208. IEEE Computer Society, 1998.

[67] H. Yuan and G. Li. Distributed in-memory trajectory similarity search and join on road network. In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*, pages 1262–1273, 2019.

[68] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 316–324, 2011.

[69] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: driving directions based on taxi trajectories. In *18th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2010, November 3-5, 2010, San Jose, CA, USA, Proceedings*, pages 99–108, 2010.

[70] M. A. Zaharia, R. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. E. Gonzalez, S. Shenker, and I. Stoica. Apache spark. *Communications of the ACM*, 59:56 – 65, 2016.

[71] H. Zhang, X. Zhang, Q. Jiang, B. Zheng, Z. Sun, W. Sun, and C. Wang. Trajectory similarity learning with auxiliary supervision and optimal matching. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3209–3215, 2020.

[72] Y. Zhang, A. Liu, G. Liu, Z. Li, and Q. Li. Deep representation learning of activity trajectory similarity computation. In E. Bertino, C. K. Chang, P. Chen, E. Damiani, M. Goul, and K. Oyama, editors, *2019 IEEE International Conference on Web Services, ICWS 2019, Milan, Italy, July 8-13, 2019*, pages 312–319. IEEE, 2019.

[73] Z. Zhang, X. Qi, Y. Wang, C. Jin, J. Mao, and A. Zhou. Distributed top-k similarity query on big trajectory streams. *Frontiers Comput. Sci.*, 13(3):647–664, 2019.

[74] B. Zheng, L. Weng, X. Zhao, K. Zeng, X. Zhou, and C. S. Jensen. REPOSE: distributed top-k trajectory similarity search with local reference point tries. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*, pages 708–719. IEEE, 2021.

[75] B. Zheng, N. J. Yuan, K. Zheng, X. Xie, S. W. Sadiq, and X. Zhou. Approximate keyword search in semantic trajectory database. In J. Gehrke, W. Lehner, K. Shim, S. K. Cha, and G. M. Lohman, editors, *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, pages 975–986. IEEE Computer Society, 2015.

[76] K. Zheng, S. Shang, N. J. Yuan, and Y. Yang. Towards efficient search for activity trajectories. In C. S. Jensen, C. M. Jermaine, and X. Zhou, editors, *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*, pages 230–241. IEEE Computer Society, 2013.

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

[78] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W. Ma. Understanding mobility based on GPS data. In *UbiComp 2008: Ubiquitous Computing, 10th International Conference, UbiComp 2008, Seoul, Korea, September 21-24, 2008, Proceedings*, pages 312–321, 2008.

[79] Y. Zheng, X. Xie, and W. Ma. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.*, 33(2):32–39, 2010.

[80] Y. Zheng, L. Zhang, X. Xie, and W. Ma. Mining interesting locations and travel sequences from GPS trajectories. In *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*, pages 791–800, 2009.

[81] P. Zhou, D. Zhang, B. Salzberg, G. Cooperman, and G. Kollios. Close pair queries in moving object databases. In C. Shahabi and O. Boucelma, editors, *13th ACM International Workshop on Geographic Information Systems, ACM-GIS 2005, November 4-5, 2005, Bremen, Germany, Proceedings*, pages 2–11. ACM, 2005.

[82] C. Zhu, J. Xu, C. Liu, P. Zhao, A. Liu, and L. Zhao. Efficient trip planning for maximizing user satisfaction. In M. Renz, C. Shahabi, X. Zhou, and M. A. Cheema, editors, *Database Systems for Advanced Applications - 20th International Conference, DASFAA 2015, Hanoi, Vietnam, April 20-23, 2015, Proceedings, Part I*, volume 9049 of *Lecture Notes in Computer Science*, pages 260–276. Springer, 2015.