Distributed Semantic Trajectory Similarity Search

Shenghao Gong[†], Ziquan Fang[†], Lu Chen[♯], Liu Liu[♯], Yunjun Gao[♯]
[†]School of Software, Zhejiang University, Ningbo, China
[♯]College of Computer Science, Zhejiang University, Hangzhou, China
[†♯]{gongshenghao, zqfang, luchen, liu2, gaoyj}@zju.edu.cn

ABSTRACT

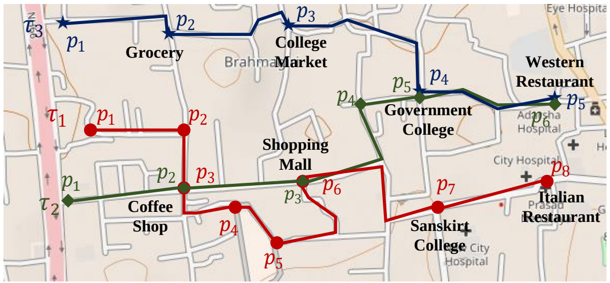
With the proliferation of location-based services, mobile applications have generated a massive amount of semantic trajectory data of moving objects. In this paper, we consider the problem of semantic trajectory search, which is a fundamental functionality of many trajectory analysis tasks. The objective is to find "similar" trajectories of a query trajectory while considering both spatial proximity and semantic similarity. However, most of the existing studies rely on keyword-based discrete semantic modeling while cannot capture general and complex semantics such as text descriptions. Besides, they tend to design search algorithms in a single machine, resulting in limited storage and processing capacity.

To this end, we propose a distributed framework to perform robust semantic trajectory similarity searches. First, we present a new semantic trajectory representation method, which utilizes geo-sequences, semantic sequences, and semantic embeddings of trajectories to enable effective spatial-semantic aware trajectory modeling. Then, we implement the query framework in Spark via a carefully designed two-phase partitioning architecture, which leverages both the semantic and spatial domains of trajectories, as well as the inter-trajectory distances to generate computation-aware partitions so that on a query trajectory, most of the partitions in the distributed cluster can be quickly pruned. Besides, in each local partition, we build a ST-HNSW index with constituent optimizations for query acceleration. Our framework supports three popular trajectory distance measures such as DTW, LCSS, and EDR. Extensive experiments have demonstrated the excellent superiority (e.g., up to 1–3 orders of magnitude efficiency improvement) achieved by our design, compared to other methods and designed alternatives.

PVLDB Reference Format:
Shenghao Gong, Ziquan Fang, Lu Chen, Liu Liu, Yunjun Gao. Distributed Semantic Trajectory Similarity Search. PVLDB, X(X): X - X, 2024.
doi:XX.XX/XXX.XX

1 INTRODUCTION

With the proliferation of GPS devices, a large number of trajectories of vehicles and individuals have been generated [48]. Retrieving similar trajectories of a query trajectory, called trajectory similarity search, is a fundamental functionality in trajectory databases [13, 24, 41]. 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 analysis tasks such as trajectory clustering [49], movement pattern mining [9], and route planning [28]. Due to its great benefits, the community has devoted efforts into spatial-aware trajectory search studies [14, 21, 23, 25, 50, 54, 55, 61, 64], e.g., searching for similar trajectories from a spatial perspective.



$\tau_1 =$ $\langle(p_1),(p_2),(p_3,\textit{"Chat with friends"}),(p_4),(p_5),(p_6,\textit{"Try on a dress"}),(p_7,\textit{"Listen to a lecture"}),(p_8,\textit{"Delicious dinner"})\rangle$

$\tau_2 =$ $\langle(p_1),(p_2,\textit{"Buy a coffee"}),(p_3,\textit{"Drive in a traffic jam"}),(p_4),(p_5,\textit{"Unload the sports equipment"}),(p_6,\textit{"Park a car"})\rangle$

$\tau_3 =$ $\langle(p_1),(p_2,\textit{"Bump into a friend"}),(p_3,\textit{"Clothing procurement"}),(p_4),(p_5,\textit{"Cooked to perfection"})\rangle$

Figure 1: A Toy Example of Three Semantic Trajectories

Recently, the explosion of social networks (e.g., Twitter) and maps (e.g., Google map) has augmented trajectories with semantic data [11, 67]. A semantic trajectory [34] 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 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. Note that, the temporal information of trajectories is not considered, as the semantic similarity is usually applied in time-insensitive applications like travel planning and group analysis, following most existing semantic trajectory analyses [11, 12, 66, 67]. Semantic trajectory similarity search is important for location-based services. Fig. 1 shows three semantic trajectories, i.e., τ_1–τ_3. As observed, although τ_1 and τ_2 are spatially close, they express distant semantics, and 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. Overall, it is important to consider spatial proximity and semantic similarity when determining whether two trajectories are similar to each other. However, when revisiting existing semantic trajectory similarity analyses [11, 65, 66], we identify three unsolved challenges, including diverse semantic modelings, large-amount semantic trajectory processing, and query acceleration, as detailed below.

Proceedings of the VLDB Endowment, Vol. X, No. X ISSN 2150-8097.
doi:XX.XX/XXX.XX

Challenge I: How to model the diverse semantic trajectories? 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 while, "*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, there are two types of semantics, i.e., keywords and text descriptions, where the former records users' location information and the latter records users' activity information. Note that, even if τ_1 and τ_2 go through the same location, it is possible to express different user activities. In the above 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. To the best of our knowledge, existing studies [3, 6, 11, 12, 17, 45, 65, 66] focus on keyword-based semantic modelings and cannot effectively model text descriptions, which restrains themselves from general semantic extraction. In the above example, existing keyword-based semantic modeling methods directly model the activity information "*Chat with friends*" with discrete keywords, i.e., "*Chat*", "*with*", "*friends*". Leaving aside the disruption of semantic coherence of phrases, sentences, and paragraphs, we cannot enumerate all possible 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. Finally, a linear combination method [3, 6, 17, 45] is adopted to combine the spatial and semantic similarity into a spatial-semantic similarity metric, while supporting existing popular trajectory distance measures including DTW [56], LCSS [47], and EDR [10].

Challenge II: How to support distributed trajectory similarity search over a large set of semantic trajectories? Compared to the raw trajectories that consist of locations, semantic trajectories introduce semantic events, leading to a great increase in data sizes. Taking the widely used GPS trajectory datasets T-drive [58] and GeoLife [68] as examples, enriching them with corresponding semantics leads to 10 × increase in terms of storage occupation. Even if we embed semantics as vectors in the above semantic modeling phase, the storage size still increases more than 6 ×. Even worse, owing to the intricacy involved in semantic distance computation, the calculation of similarity between semantic trajectories entails higher computational complexity, compared to that of raw trajectories. As a result, semantic trajectories may easily exceed the storage capacity and computational capability of 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 [36, 41, 50, 57, 63, 64] for distributed similarity search over spatial trajectories, they do not consider the semantic aspect. There are also studies [8, 33, 51, 52] for distributed vector search, which cannot be directly employed for our problem due to their inability to leverage spatial information of trajectories. The absence of either spatial or semantic aspects will render distributed methods incapable of effectively partitioning trajectories for spatial-semantic similarity computations. In this paper, the key to building an efficient distributed query framework is an effective partitioning strategy that considers both spatial and semantic aspects, so that on a query (semantic) trajectory, most of the data partitions in the cluster of machines can be quickly pruned, which is challenging and none of the previous studies can achieve. To address this challenge, we propose a two-phase partitioning architecture over Spark [60], which leverages both the 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 in each data partition? With the above two-phase distributed data partitioning architecture, the input semantic trajectory dataset is assigned to different data partitions. Given a query trajectory, it will be broadcasted into relevant partitions for parallel intra-partition searching. This naturally brings the idea of designing an effective index in each partition for local querying and pruning. Existing studies typically build inverted indexes over keywords and utilize traditional trajectory indexes in inverted files for query pruning, including tree indexes [16, 18, 32, 35, 44] and grid indexes [7, 71]. However, there are two limitations. First, the inverted indexes rely on keywords and cannot model semantics like text descriptions of Fig. 1. Second, the traditional trajectory indexes cannot reduce the time complexity of scanning a partition, but only reduce its 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. As the data volume grows, the number of trajectories in each grid also increases. To address this challenge, we borrow the HNSW index (Hierarchical Navigable Small World [30]) that was proposed for vector retrieval and extend that 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 better efficiency. Based on the ST-HNSW, we develop a series of effective optimization techniques for further query procedure acceleration.

To address the above three challenges, we revisit semantic trajectory similarity search in terms of semantic modeling, distributed partitioning, and optimized query. i) First, we propose a new representation method to model semantic trajectories from spatial and semantic aspects, respectively. Based on that, we adopt a linear combination manner to evaluate inter-trajectory distances considering both spatial and semantic aspects, while supporting three popular trajectory measures. ii) Second, we build a distributed framework over Spark. By a carefully designed two-phase partitioning strategy, trajectories are distributed into computation-aware data partitions, so that on a query trajectory, most of the irrelevant partitions are effectively pruned. iii) Finally, a ST-HNSW index (with constituent optimization techniques) is built into each partition to accelerate local queries. This paper makes the following contributions.

- We propose a new and 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.

Table 1: Statistics of Recent Semantic Trajectory Searches

Methods	Data Size	#measures	Processing	Publication
PSTSJ [11]	8 GB	1	Standalone	ICDE 20
At2vec [26]	0.3 GB	1	Standalone	TKDE 22
T*k*Sk [67]	3 GB	1	Standalone	WWW 17
AKQST [65]	0.3 GB	1	Standalone	ICDE 15
Ours	**377 GB**	**3**	**Distributed**	–

- We build a distributed semantic trajectory search framework. We propose a two-phase and pruning-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 real and synthetic datasets demonstrate the effectiveness, efficiency, scalability, and parameter sensitivity of the proposed framework over competitors.

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

2 RELATED WORK

(A) Semantic Trajectory Representation. Research on the semantic trajectory representation can be categorized into individually keywords-tagged representation [11, 12, 65–67] and globally keywords-tagged representation [27, 40, 72]. 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 [20]. Last but not the least, keyword-based representation typically necessitates the construction of an inverted index using keywords [11]. In that case, as the number of keywords increases, the redundancy escalates. In this paper, we propose a new semantic representation by decoupling the semantic trajectory into geographical and semantic sequence, respectively. This not only resolves the issue of redundant storage but also preserves the sequential characteristics. Furthermore, we employ embedding techniques to encapsulate the text semantics.

(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 [12, 65–67] and keyword-first [11, 37, 38]. The spatial-first indexes organize trajectory points using spatial indexes such as grids [7, 71] and R-tree [18]. 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.

Table 2: 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 semantic trajectories
$\mathcal{D}^{(G)}$	The inter-distance between geo-sequences
$\mathcal{D}^{(S)}$	The inter-distance between semantic sequences
α	A weight parameter 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 [11, 37, 38, 65, 67] 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 [3, 6, 17, 26, 53, 62] is based on deep learning. Specifically, they employ deep models like Seq2seq [43] and Attention [46] 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 [38]. However, it still relies on a keyword-based inverted index and the proposed pruning method lacks efficiency.

(D) The main statistics of the related semantic trajectory search methods are **summarized** in Table 1, from which we find that, we propose the first distributed semantic trajectory framework that supports much larger data sizes and more trajectory measures.

3 SEMANTIC TRAJECTORY MODELING

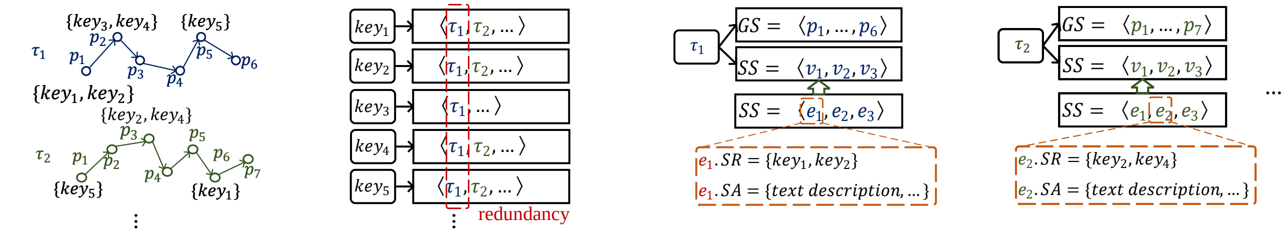
We first detail how to represent semantic trajectories. Then, we define trajectory distance functions considering both spatial and semantic aspects. Finally, we give problem formulation as well as baseline solutions. Table 2 lists the frequently used symbols.

3.1 Semantic Trajectory Representation

When reviewing existing semantic trajectory representation methods, there are **individual-based** forms [11, 12, 65–67] and **global-based** forms [27, 40, 72]. 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 representation methods rely on **raw keywords** to express the semantics, which is insufficient. 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 [11, 66, 67] typically enumerate all keywords and build inverted indexes to manage trajectories, resulting in massive redundancy. To this end, we propose a new semantic trajectory representation manner. We first introduce the semantic event.

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


$\{key_3, key_4\}$
$\{key_5\}$
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

Figure 2: Semantic Trajectory Representation

Definition 3.1. **(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 characterizing activities that semantically happened at SR.*

According to Definition 3.1, 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 [26, 53, 62, 66, 67] 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.

In this paper, given a trajectory, we then model its spatial aspect and semantic aspect via a geographical sequence and a semantic sequence, respectively. Note that, referring existing semantic trajectory studies [11, 12, 65–67], semantic similarity is typically applied in offline and time-insensitive 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. Therefore, in this paper, the precise timestamp information is not included in trajectory modeling, leaving a future study in the community.

Definition 3.2. **(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 below.*

Definition 3.3. **(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 3.4. **(Semantic Trajectory,** τ**)** *A semantic trajectory of a moving object 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 the paper, if there are no special instructions, we refer to **semantic trajectory** as **trajectory** for simplification.

To enable robust semantic modeling, we use representation learning methods. Specifically, we apply FastText [2] to embed SRs and SAs 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 $\boldsymbol{v_i} = \overrightarrow{e_i.SR} + \overrightarrow{e_i.SA}$ and $\boldsymbol{v_i} \in \mathbb{R}^d$. Recall Definition 3.2, 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.

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 $d(e_a, e_b) = \sqrt{\sum_{i=1}^{d}(v_a[i] - v_b[i])^2}$. In the rest of the paper, we treat semantic sequences as embedded sequences as default.

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 in trajectory similarity calculations.

3.2 Trajectory Distance Functions

The focus of our work is not to investigate different similarity search measures. Instead, we adopt the well-known DTW [56], LCSS [47], and EDR [10] for spatial-semantic trajectory distance evaluations.

Definition 3.5. **(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 $\mathcal{D}^{(S)}(\tau_i, \tau_j)$:*

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

where $\mathcal{D}^{(G)}(\cdot, \cdot)$ computes the distance between their geo-sequences, $\mathcal{D}^{(S)}(\cdot, \cdot)$ computes the distance between their semantic sequences, and α balances the weight between spatial and semantic aspects. $\mathcal{D}$ denotes DTW [56], LCSS [47], or EDR [10].

Next, we use LCSS [47] to illustrate the spatial-semantic trajectory distance computation. Given two trajectories $\tau_a = \langle GS_a, SS_a \rangle$ and $\tau_b = \langle GS_b, SS_b \rangle$, where $GS_a = \langle p_1^a, p_2^a, ..., p_n^a \rangle$, $GS_b = \langle p_1^b, p_2^b, ..., p_{n'}^b \rangle$, $SS_a = \langle v_1^a, v_2^a, ..., v_m^a \rangle$, and $SS_b = \langle v_1^b, v_2^b, ..., v_{m'}^b \rangle$, The LCSS$^{(G)}$ and LCSS$^{(S)}$ distances between τ_a and τ_b are computed below.

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

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

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 $\mathrm{LCSS}^{(G)}$ and $\mathrm{LCSS}^{(S)}$ is $d(\cdot,\cdot)$, which denotes **point-to-point** distance in $\mathrm{LCSS}^{(G)}$ and **vector-to-vector** distance in $\mathrm{LCSS}^{(S)}$. Based on a linear combination of $\mathrm{LCSS}^{(G)}$ and $\mathrm{LCSS}^{(S)}$, we evaluate the distance between τ_a and τ_b considering both spatial and semantic aspects. The computations of DTW [56] and EDR [10] measures are omitted, as they share a similar process.

3.3 Problem Statements

In this paper, we consider a moving object of the two-dimensional free space since it represents the most common scenarios. It is straightforward to extend our solutions to road networks.

Definition 3.6. **(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 the case of the first two measures, while a larger LCSS distance between τ_i and τ_j means that they are more similar.

3.4 Baselines

Before introducing our framework, we detail two distributed baselines by extending classic standalone methods.

`ID-Force`: It partitions the dataset based on trajectory IDs. Specifically, trajectories are distributed into data partitions via a HashPartitioner [31]. Then, a local index is constructed in each partition, where existing standalone techniques such as HNSW [30] can be employed. Based on that, each query τ_q is broadcasted 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.

`Dis-PSTSJ`: It is a distributed variant of PSTSJ [11] that supports parallel semantic trajectory similarity computation. We extend PSTSJ as follows. First, all trajectories are organized via an inverted list based on keywords, in which each file comprises a subset of the dataset. With HashPartitioner [31], we distribute inverted files across partitions. Subsequently, each partition manages its trajectories based on the Minimal Bounding Rectangle (MBR). During the search process, 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.

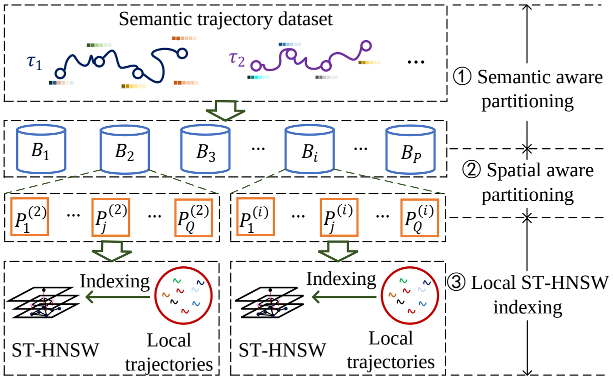


Figure 3: The Framework Overview

4 THE FRAMEWORK OVERVIEW

The key to building an efficient distributed search framework lies in data partitioning so that on a query trajectory, most of the data partitions can be quickly skipped. Unlike the above baselines that simply utilize ID or semantic features for coarse-grained partitioning, we leverage both semantic and spatial features, as well as inter-trajectory distance simultaneously. Recall Equation 1, we aim to derive the bounds of inter-trajectory distance $\mathcal{D}(\cdot,\cdot)$ and utilize that to achieve computation-aware data partitioning.

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 **data batches** via semantic features, where trajectories that are closer in the semantic distance $\mathcal{D}^{(S)}$ are grouped in the same batch and those farther away are placed in different batches. For each data batch, we further distribute it into Q **data partitions** via spatial features, where trajectories that are closer in the spatial distance are grouped in the same partition. Based on the above two-phase 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 a ST-HNSW index to speed up local queries.

5 TWO-PHASE 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, $\mathcal{T} \rightarrow \mathcal{T}' \rightarrow \mathcal{T}'' = \{P_1^{(1)}, ..., P_Q^{(1)}; P_1^{(2)}, ..., P_Q^{(2)}; ...; P_1^{(P)}, ..., P_Q^{(P)}\}$.

5.1 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.

5.1.1 DPiSAX-based partitioning. Specifically, we adopt DPiSAX [52] for semantic event vector clustering, which uses the iSAX encoding to compress vectors into sequences consisting of binary strings. According to the previous work [22], 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 $iSAX(v_a)$ and $iSAX(v_b)$ 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 5.1. *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 semantic event distance. Then we have $d(v_a, v_b) \geq d(iSAX(v_a), iSAX(v_b))$.*

Proof. The proof can be found elsewhere [22]. □

According to Lemma 5.1, 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, for each data batch $\hat{B}$, all its assigned 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 5.2. **(Trajectory-to-Batch Distance)** *Given a trajectory τ and a data batch $\hat{B}$, we define their distance as*

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

5.1.2 Data batch pruning in Search. Based on Definition 5.2, 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

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

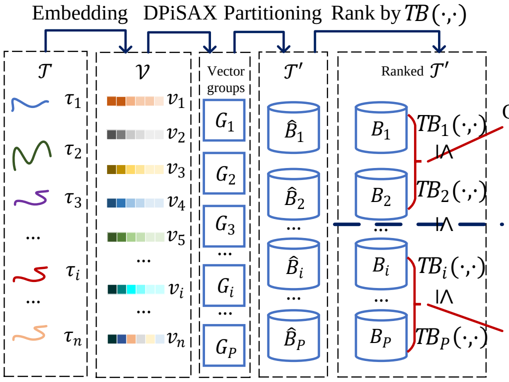


Figure 4: The Semantic Aware Partitioning

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.

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

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

Then, we unify the weight $\mathcal{W}$ of the above distance functions.

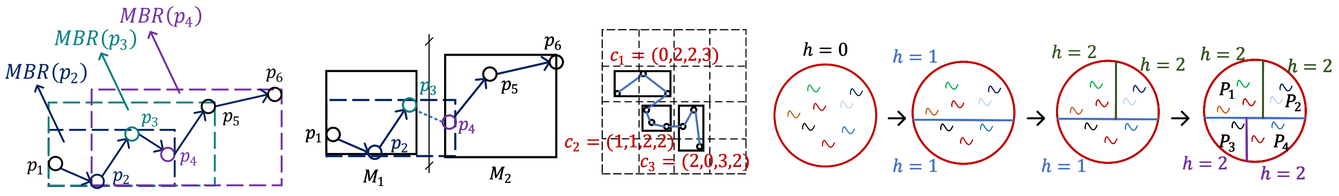
$$\mathcal{W}(d(\cdot, \cdot)) = \begin{cases} d(\cdot, \cdot) & \text{if } \mathcal{D} \text{ is } DTW \\ \begin{cases} 1 & \text{if } d(\cdot, \cdot) \leq \varepsilon \\ 0 & otherwise \end{cases} & \text{, if } \mathcal{D} \text{ is } LCSS \\ \begin{cases} 1 & \text{if } d(\cdot, \cdot) \geq \varepsilon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

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

Lemma 5.3. *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 5.2, 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 5.1, $\min_{v_i \in \tau.SS} \min_{v_j \in \tau_q.SS} d(iSAX(v_i), iSAX(v_i)) \leq \min_{v_i \in \tau.SS} \min_{v_j \in \tau_q.SS} d(v_i, v_j)$. 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 5.3, 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



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

Figure 5: The Spatial Aware Partitioning

$i \le j$. Here, $1 \le i \le P, 1 \le j \le P$, $B_i \in \mathcal{T}'$, 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 $(1 \le j \le i)$, the data batches ranked behind B_i can be pruned.

5.2 Spatial Aware Partitioning

As shown in Fig. 3, we split each data batch into data partitions leveraging the spatial features of trajectories. Fig. 5 shows the spatial aware partitioning, which involves four main steps: (a) envelope, (b) segmentation, (c) encoding, and (d) partitioning.

5.2.1 *Envelope.*

Definition 5.4. **(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*

$$x_i^l = \min\{x_j | i - \delta \le j \le i + \delta\}, \quad y_i^l = \min\{y_j | i - \delta \le j \le i + \delta\}$$
$$x_i^u = \max\{x_j | i - \delta \le j \le i + \delta\}, \quad y_i^u = \max\{y_j | i - \delta \le j \le i + \delta\} \tag{10}$$

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

Definition 5.5. **(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*

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

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

Definition 5.6. **(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 $m \le n$, we define their inter-distance as*

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

Lemma 5.7. *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 $d(p_i, p_j) \ge d(MBR(p_i), MBR(p_j))$. Assume that the length of GS_a and GS_b are m and n, respectively, where $m \le n$, $\mathcal{D}(GS_a, GS_b) = \sum_{c=1}^{C} \mathcal{W}^c \ge \sum_{i=1}^{m} \mathcal{W}\ (d(MBR(p_i^a), MBR(p_i^b)))$ + $\sum_{i=m+1}^{\min(n,m+\delta)} \mathcal{W}\ (d(MBR(p_m^a), MBR(\ p_i^b)))$. Then $\mathcal{D}(GS_a, GS_b) \ge MSD(\ MS_a, MS_b)$. The above proof still holds when $n \le m$. EDR and LCSS measures share a similar proof and thus are omitted. □

Based on Lemma 5.7, 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.

5.2.2 *Segmentation.*

Definition 5.8. **(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*

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

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

5.2.3 *Encoding.*

Definition 5.9. **(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 $\mathsf{MS} = \langle c_1, ..., c_i, ..., c_{\frac{n}{\mathcal{L}}}\rangle$, 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 5.10. **(Encoded Inter MBR-sequence Distance)** *Given two encoded MBR-sequences* $\mathsf{MS}_a = \langle c_1^a, ..., c_{sa}^a \rangle$ *and* $\mathsf{MS}_b = \langle c_1^b, ..., c_{sb}^b \rangle$*, assume* $sa \le sb$*, the distance between* MS_a *and* MS_b *is*

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

Lemma 5.11. *Given two MBR sequences* MS_a *and* MS_b *of two geo-sequences, as well as their segmented and encoded MBR sequences* MS_a *and* MS_b*. When* MS_a *and* MS_b *have the same fixed length* $\mathcal{L}$ *of MBR segment, then we have* $MSD(MS_a, MS_b) \mid = \mathcal{L} \cdot EMSD(\mathsf{MS}_a, \mathsf{MS}_b)$*.*

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 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 5.9, 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 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 $MSD(\ GS_a,$ $GS_b) \ge \mathcal{L} \cdot EMSD(\mathsf{MS}_a, \mathsf{MS}_b)$. EDR and LCSS share a similar proof process and are omitted. □

5.2.4 Partitioning.

Definition 5.12. **(Partition Code of an MBR)** *Given an encoded and segmented MBR sequence* MS*, for each MBR* $c_i = (c_i^l, c_i^d, c_i^r, c_i^u) \in \mathsf{MS}$*, 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 $\mathsf{MS} = \langle c_1, ..., c_i, ..., c_{\frac{n}{\mathcal{L}}} \rangle$, 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.

5.2.5 *Data partition pruning in Search.*

We first define the distance between a trajectory and a data partition.

Definition 5.13. **(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* $\mathsf{MS} = \langle c_1, ..., c_{\frac{n}{\mathcal{L}}} \rangle$*, where* $pc_i = (pc_i^l, pc_i^d, pc_i^r, pc_i^u)$*. Assume that 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 *as*

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

Lemma 5.14. *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* $\mathcal{D}(\tau_q.GS, \tau.GS) \mid = \mathcal{L} \cdot TP(\tau_q.GS, P)$*.*

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 MS_q and MS denote the encoded MBR-sequence of $\tau_q.GS$ and $\tau.GS$, respectively. According to Definitions 5.9 and 5.12, $\forall i \in [1, d]$, assume $c_i = \mathsf{MS}.c_i = MBR(c_i^l, c_i^d, c_i^r, c_i^u)$ and $c_i^q = \mathsf{MS}_q.c_i$, 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, $d(c_i^q,\ MBR(2^{\omega-1}pc_i^l,\ 2^{\omega-1}pc_i^d,\ 2^{\omega}pc_i^r,\ 2^{\omega}pc_i^u)) \le d(c_i^q,$ $c_i)$. According to Definition 5.13, 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 $d(c_i^q, c_i) \ge d(c_i^q, pc_i)$. Assume the lengths of $\tau_q.\mathsf{MS}$ and $\tau.\mathsf{MS}$ 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 $EMSD(\mathsf{MS}_q, \mathsf{MS}) \mid = TP(\tau_q.GS, P)$. When $\min(sq, s) = s$, the same result are obtained. According to Lemmas 5.7 and 5.11, we have $\mathcal{D}(\tau_q.GS, \tau.GS) \mid = MSD(\tau_q.GS, \tau.GS) \mid = \mathcal{L}\ EMSD(\mathsf{MS}_q, \mathsf{MS})$ $\mid = \mathcal{L}\ TP(\tau_q.GS, P)$, that is $\mathcal{D}(\tau_q.GS, \tau.GS) \mid = \mathcal{L}\,TP(\tau_q.GS, P)$. □

6 LOCAL INDEX AND QUERY PROCESS

With the above two-phase distributed partitioning, the input trajectories are distributed into data partitions. Based on that, given a query trajectory, it will be sent to relevant partitions for intra-partition queries. To accelerate the query process in each data partition, we further build the local ST-HNSW index for pruning.

6.1 Local ST-HNSW Index

6.1.1 *HNSW Index.*

Hierarchical Navigable Small World [30] 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 [29], FLANN [15], VP-tree [4], and FALCONN [1]. This is the first proposal to deploy HNSW into semantic trajectory similarity search.

6.1.2 *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.

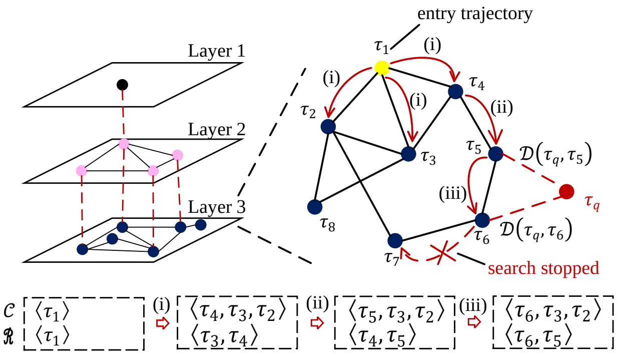

entry trajectory
Layer 1
Layer 2
Layer 3
τ_1
(i)
(i)
(i)
(ii)
(iii)
τ_2
τ_3
τ_4
τ_5
τ_6
τ_7
τ_8
$\mathcal{D}(\tau_q, \tau_5)$
$\mathcal{D}(\tau_q, \tau_6)$
τ_q
search stopped
$\mathcal{C}$ $\langle \tau_1 \rangle$ (i) $\langle \tau_4, \tau_3, \tau_2 \rangle$ (ii) $\langle \tau_5, \tau_3, \tau_2 \rangle$ (iii) $\langle \tau_6, \tau_3, \tau_2 \rangle$
$\mathfrak{R}$ $\langle \tau_1 \rangle$ $\langle \tau_3, \tau_4 \rangle$ $\langle \tau_4, \tau_5 \rangle$ $\langle \tau_6, \tau_5 \rangle$

Figure 6: The Search Process in ST-HNSW

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) [29], with the bottom layer being a complete NSW that contains all trajectories.

6.1.3 ST-HNSW Search. Here, 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 trajectory τ_q, the query stops, and $\{\tau_5, \tau_6\}$ is returned. The query process concludes.

6.2 Query Process

6.2.1 Two-Phase based Search. Given a query trajectory τ_q and a trajectory dataset $\mathcal{T}$, based on the semantic aware partitioning (see Sec. 4.1), 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 (see Sec. 4.2), 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 6.1. **(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*

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

Lemma 6.2. *Given a query trajectory τ_q, a trajectory distance function $\mathcal{D}$ (i.e., DTW, LCSS, or EDR), 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 5.3 and 5.14. □

Based on Lemma 6.2, 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).

6.2.2 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 6.3. *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 $\tau_n \notin \mathcal{R}$. □

Since the edge weights are only used for graph construction and triangle inequality pruning during the search, we use the bound distance BD (Definition 5.1) as the edge weights, which process has linear time complexity.

Lemma 6.4. *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 6.2, $\mathcal{D}(\tau_c, \tau_n)\ | = BD(\tau_c, \tau_n)$. 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 $\tau_n \notin \mathcal{R}$. □

Algorithm 1: Semantic Trajectory Similarity Search

```
Input: a set of queries $Q$, a distance function $\mathcal{D}$, the parameter $k$,
       the precision guarantee $\epsilon$, 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      $R_\tau[\tau_q.ID] \leftarrow$ PriorityQueue($k$)
4      bsfDist $\leftarrow \infty$
5      for each batch $B \in \mathcal{T}''$ do
6          for each partition $P \in B$ do
7              if $(1+\epsilon) \cdot bsfDist \mid = BD(\tau_q, P)$ then
8                  // Definition 6.1 and Lemma 6.2
9                  LocalRes $\leftarrow$ Optimized ST-HNSW Search
10                 // Section 6.2.2
11                 for $\tau \in LocalRes$ do
12                     $R_\tau[\tau_q.ID]$.add($\tau$)
13                 bsfDist $\leftarrow \mathcal{D}(\tau_q, R_\tau[\tau]$.tail())
14         if $\alpha \cdot TB(\tau_q, B) \mid = (1+\epsilon) \cdot bsfDist$ then
15             // Lemma 5.3
16             break
```

In the weighted ST-HNSW, a fixed entry point may increase search steps because the distance between the entry point and its neighbors is far, requiring multiple hops to reach the target. To this end, we propose a simple grid index to find a suitable entry point in advance. Specifically, we divide the space into grids, where each grid serves as an index item containing 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 effectively reduces the number of search steps and improves ST-HNSW searching efficiency.

Finally, we give the distributed semantic trajectory similarity search in Algorithm 1. It takes a set Q of query trajectories, a distance function $\mathcal{D}$, the parameter k of kNN, the precision guarantee ϵ, and a trajectory dataset $\mathcal{T}'' = Map(Int, List(List()))$ as input. Lines 2–4 initialize a result queue for each query, where each queue maintains the current nearest distance values. Initially, the nearest distance value is set to infinite. Note that, each queue has a predetermined length of k. Then, lines 5–7 calculate the bound distance BD between a query trajectory and a data partition. If the bound distance does not violate Lemma 6.2, we dispatch this query into this partition and perform optimized ST-HNSW search (lines 8–13). Simultaneously, if the TB distance between the query trajectory to the current data batch violates Lemma 5.3, we directly prune all data partitions from this data batch (lines 14–16).

7 EXPERIMENT

We use below research questions (RQs) to guide experiments:

RQ1. How does the proposed semantic modeling measure perform compared to existing semantic modeling methods?

RQ2. How do the proposed methods perform compared to existing semantic trajectory similarity search baselines in effectiveness, efficiency, scalability, resource occupancy, and workload?

RQ3. How useful of the proposed pruning technology?

Table 3: 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 4: 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 of semantic-spatial partitions	0.5, **1**, 1.5, 2
Search result size k	5, 10, 20, **50**
# trajectories O_r (%)	25, 50, 75, **100**
# parallel subtasks $N = P \times Q$ in Spark	48, 72, 96, **120**

RQ4. How robust is the framework for hyperparameters?

RQ5. How does this framework support other similarity queries?

7.1 Setup

Datasets. Since different moving objects may exhibit different traveling characteristics, we use different datasets to examine whether our framework is robust to different motions. We use two real datasets, T-Drive [58, 59] and GeoLife [68–70], as well as a synthetic dataset, Brinkhoff [5]. 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.

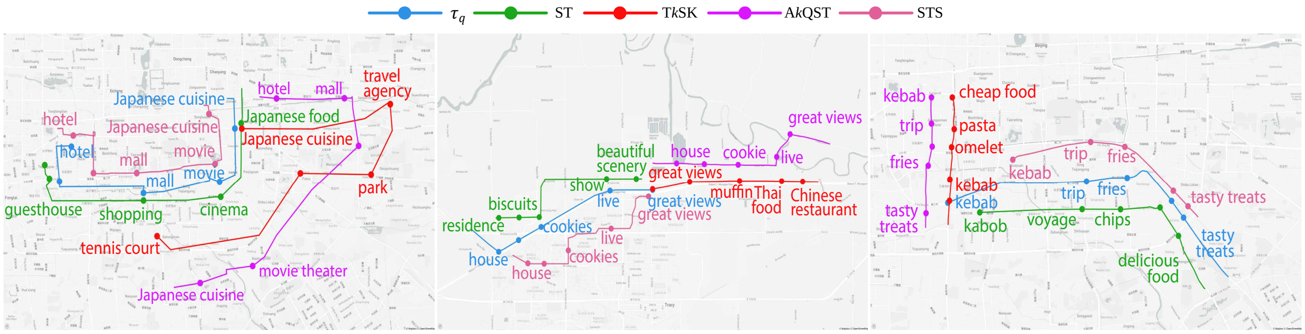
Following previous works [11, 65–67], we augment raw trajectory datasets with semantics. For T-Drive and GeoLife, we employ AMap[2] APIs to retrieve keywords for each GPS point and associate them as attributes to their semantic regions. Then, we crawled Weibo[3] check-in data. By using the coordinates of the check-in data, we associated them with the corresponding coordinates in T-Drive and GeoLife, i.e., assigning semantic activities. For Brinkhoff, we select a set of keywords from Amap and a collection of real-life tweets [39] to associate them with 40% of Brinkhoff data. Table 3 summarizes the statistics of the datasets used.

Parameters. Table 4 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.

Methods compared. As we propose a new semantic modeling method, based on which, we build the distributed query framework.

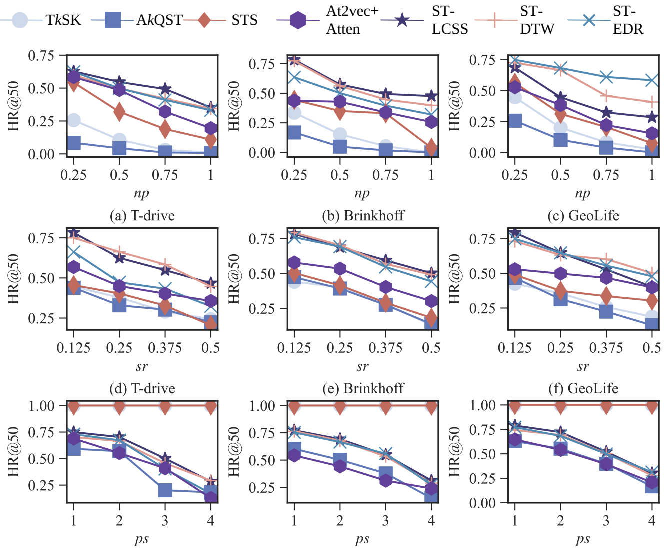
[2] https://lbs.amap.com/

[3] https://www.weibo.com/



(a) T-Drive (b) Brinkhoff (c) GeoLife

Figure 7: Case Studies: Expressiveness Capability of Different Semantic-aware Trajectory Distance Measures



(a) T-drive (b) Brinkhoff (c) GeoLife

(d) T-drive (e) Brinkhoff (f) GeoLife

(g) T-drive (h) Brinkhoff (i) GeoLife

Figure 8: Trajectory Measure Robustness Evaluation

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) T*k*ST [67] that combines global keyword set similarity and spatial similarly, ii) A*k*QST [65] that combines individual keyword similarity and spatial similarity, iii) STS [11] that combines GPS points with keyword annotations, and iv) At2vec+Atten [26] 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), i.e., Algorithm 1. 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 [11] that uses invert indexes for pruning; iii) a distributed implementation of At2vec+Atten [26], and iv) LSTS-Join [38] 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. Each cluster node runs Ubuntu 14.04.3 LTS and Spark 3.1.1. For potential studies, all codes, datasets are released at https://github.com/ZJU-DAILY/DSTSS.

7.2 Semantic Modeling Study (RQ1)

To verify the effectiveness of different semantic modeling methods, we select three dense areas from three 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 in T-Drive and GeoLife, while returns the same as STS in Brinkhoff. Overall, we have the following observations: (i) T*k*SK returns a semantic-similar while spatial faraway trajectory, as T*k*SK mainly considers the global semantic similarities. (ii) Although A*k*QST 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 all three datasets.

Following previous studies [19, 42], we use three noising parameters, i.e., noising-probability *np*, noising-sampling *sr*, and noising-shifting *sp*, to evaluate the robustness of spatial-semantic trajectory measure (i.e., Eq. 1) Specifically, we compare the query results returned from the raw dataset and those returned from the dataset after perturbation processing. The noising process via three parameters can be found in [19] and thus omitted. 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, this is because we use vector representations while the other measures use raw keyword forms. Finally, for the point shifting perturbation, T*k*SK and STS are not affected, while our measure is more robust than other affected measures, indicating that our spatial-semantic measure is order-sensitive, which is applicable in real life.

7.3 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, 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 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 two-phase data 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 proposed local optimization techniques on ST-HNSW. Overall, our methods trade effectiveness for efficiency, i.e., an acceptable effectiveness drop for 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, DisPSTSJ 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.

7.4 Parameter Sensitivity Study (RQ4)

We also explore the effects of the parameters, i.e., spatial-semantic weight α, segment length $\mathcal{L}$, grid length δ_g, and #data batchs/#data partitions P/Q on the performance.

Effect of α. Figs. 12(a)–12(c) report the HR@50 of DSTSS and DSTSS w/o LO by varying α from 0.25 to 1.00. 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. Therefore, only when α takes an appropriate weight between 0 and 1, HR@50 can reach 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.

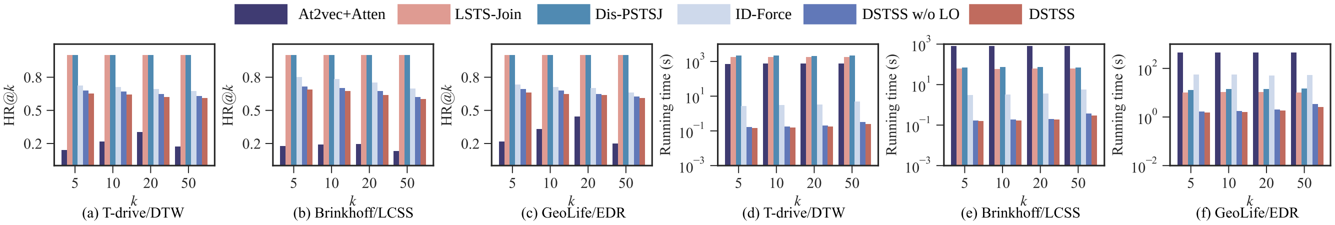
Effect of P/Q**.** Figs. 12(j)–12(l) plot the results by varying P/Q. When one of P or Q is very small while the other is large, both running time and index time will increase. This is because a partitioner with a smaller number of partitions has reduced pruning capabilities and the load becomes more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

7.5 Threshold-based Similarity Search (RQ5)

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

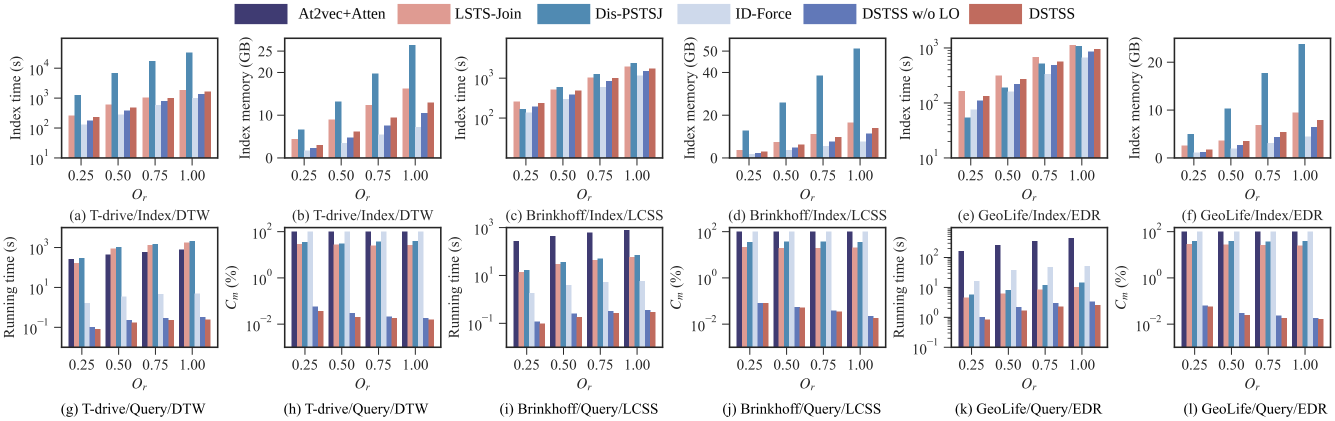
Definition 7.1. **(Threshold-based Similarity Search, TSS)** *Given a set of trajectories* $\mathcal{T}$*, a query trajectory* τ_q*, a distance function* $\mathcal{D}(\cdot,\cdot)$*, and a distance threshold* θ*, TSS returns a set of trajectories* $\mathcal{R} \subset \mathcal{T}$*, where* $\forall \tau \in \mathcal{R}, \theta| = \mathcal{D}(\tau, \tau_q)$.

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 $\mathcal{T}'' = Map(Int, List(List()))$ as input. Lines 2–3 initialize a result list for each query. Then, lines 4–7 calculate the bound



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

Figure 9: Effectiveness and Efficiency Evaluation vs. k



(a) T-drive/Index/DTW (b) T-drive/Index/DTW (c) Brinkhoff/Index/LCSS (d) Brinkhoff/Index/LCSS (e) GeoLife/Index/EDR (f) GeoLife/Index/EDR

(g) T-drive/Query/DTW (h) T-drive/Query/DTW (i) Brinkhoff/Query/LCSS (j) Brinkhoff/Query/LCSS (k) GeoLife/Query/EDR (l) GeoLife/Query/EDR

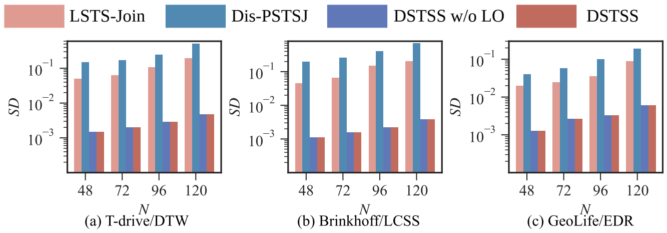
Figure 10: Scalability Evaluation vs. Data sizes O_r

Algorithm 2: 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  $R_\tau(Int, List())$ ← $Null$ // a map to store the results for each query
2  for each query $\tau_q \in Q$ do
3      $R_\tau[\tau_q.ID]$ ← 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 6.1 and Lemma 6.2
8                  LocalRes ← 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 5.3
13             break
```

distance BD between a query trajectory and a data partition. If the bound distance does not violate Lemma 6.2, 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 5.3, 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.



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

Figure 11: Workload Evaluation

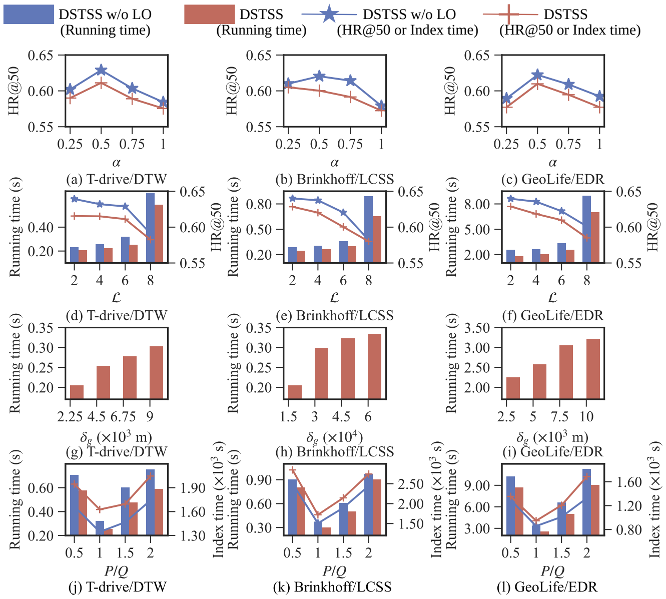
However, as θ increases, the running time grows rapidly because it requires evaluating and returning a large number of trajectories. This leads to a deterioration in the pruning effect of all methods, while our method, although experiencing a performance decline, still maintains great superiority over the baselines.

8 CONCLUSIONS

In this paper, we propose a distributed framework for semantic trajectory similarity search. Specifically, we propose a new semantic representation method to enable robust semantic modeling. Then, we develop a computation-aware two-phase partitioning architecture, so that on a query trajectory, most of the data partitions can be efficiently pruned. In each partition, we develop several optimization techniques based on ST-HNSW index for further speed up. Experiments using real and synthetic datasets show the superiority of our methods. In the future, it is interesting to include more types of semantics like geo-photos and videos or the temporal perspective into semantic trajectory analyses.

REFERENCES

[1] Alexandr Andoni and Ilya P. Razenshteyn. 2015. Optimal Data-Dependent Hashing for Approximate Near Neighbors. In *Proceedings of the Forty-Seventh*



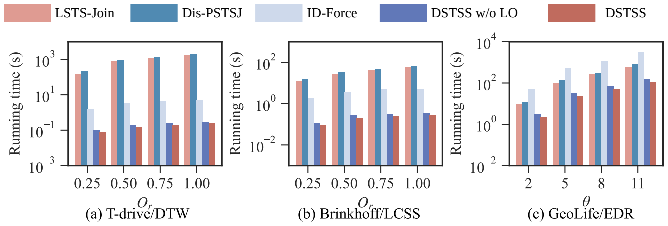
(a) T-drive/DTW (b) Brinkhoff/LCSS (c) GeoLife/EDR

(d) T-drive/DTW (e) Brinkhoff/LCSS (f) GeoLife/EDR

(g) T-drive/DTW (h) Brinkhoff/LCSS (i) GeoLife/EDR

(j) T-drive/DTW (k) Brinkhoff/LCSS (l) GeoLife/EDR

Figure 12: Parameter Sensitivity Study



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

Figure 13: Threshold-based Similarity Search Evaluation

Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, Rocco A. Servedio and Ronitt Rubinfeld (Eds.). ACM, 793–801. https://doi.org/10.1145/2746539.2746553

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomás Mikolov. 2017. Enriching Word Vectors with Subword Information. *Trans. Assoc. Comput. Linguistics* 5 (2017), 135–146. https://doi.org/10.1162/tacl_a_00051

[3] Thapana Boonchoo, Xiang Ao, and Qing He. 2019. Multi-Aspect Embedding for Attribute-Aware Trajectories. *Symmetry* 11, 9 (2019), 1149. https://doi.org/10.3390/sym11091149

[4] Leonid Boytsov and Bilegsaikhan Naidan. 2013. Learning to Prune in Metric and Non-Metric Spaces. In *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*, Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (Eds.). 1574–1582. https://proceedings.neurips.cc/paper/2013/hash/fc8001f834f6a5f0561080d134d53d29-Abstract.html

[5] Thomas Brinkhoff. 2002. A Framework for Generating Network-Based Moving Objects. *GeoInformatica* 6, 2 (2002), 153–180. https://doi.org/10.1023/A:1015231126594

[6] Hancheng Cao, Fengli Xu, Jagan Sankaranarayanan, Yong Li, and Hanan Samet. 2020. Habit2vec: Trajectory Semantic Embedding for Living Pattern Recognition in Population. *IEEE Trans. Mob. Comput.* 19, 5 (2020), 1096–1108. https://doi.org/10.1109/TMC.2019.2902403

[7] V. Prasad Chakka, Adam Everspaugh, and Jignesh M. Patel. 2003. 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. http://www-db.cs.wisc.edu/cidr/cidr2003/program/p15.pdf

[8] Manos Chatzakis, Panagiota Fatourou, Eleftherios Kosmas, Themis Palpanas, and Botao Peng. 2023. Odyssey: A Journey in the Land of Distributed Data Series Similarity Search. *Proc. VLDB Endow.* 16 (2023), 1140–1153. https://api.semanticscholar.org/CorpusID:256274619

[9] Lu Chen, Yunjun Gao, Ziquan Fang, Xiaoye Miao, Christian S. Jensen, and Chenjuan Guo. 2019. Real-time Distributed Co-Movement Pattern Detection on Streaming Trajectories. *Proc. VLDB Endow.* 12, 10 (2019), 1208–1220. https://doi.org/10.14778/3339490.3339502

[10] Lei Chen, M. Tamer Özsu, and Vincent Oria. 2005. Robust and Fast Similarity Search for Moving Object Trajectories. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005*, Fatma Özcan (Ed.). ACM, 491–502. https://doi.org/10.1145/1066157.1066213

[11] Lisi Chen, Shuo Shang, Christian S. Jensen, Bin Yao, and Panos Kalnis. 2020. Parallel Semantic Trajectory Similarity Join. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*. IEEE, 997–1008. https://doi.org/10.1109/ICDE48307.2020.00091

[12] Gao Cong, Hua Lu, Beng Chin Ooi, Dongxiang Zhang, and Meihui Zhang. 2012. Efficient Spatial Keyword Search in Trajectory Databases. *CoRR* abs/1205.2880 (2012). arXiv:1205.2880 http://arxiv.org/abs/1205.2880

[13] Jian Dai, Bin Yang, Chenjuan Guo, and Zhiming Ding. 2015. Personalized route recommendation using big trajectory data. In *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, Johannes Gehrke, Wolfgang Lehner, Kyuseok Shim, Sang Kyun Cha, and Guy M. Lohman (Eds.). IEEE Computer Society, 543–554. https://doi.org/10.1109/ICDE.2015.7113313

[14] Jiafeng Ding, Junhua Fang, Zonglei Zhang, Pengpeng Zhao, Jiajie Xu, and Lei Zhao. 2019. 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*. 1398–1405. https://doi.org/10.1109/HPCC/SmartCity/DSS.2019.00194

[15] Yixiang Fang, Reynold Cheng, Wenbin Tang, Silviu Maniu, and Xuan S. Yang. 2016. 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*. IEEE Computer Society, 1528–1529. https://doi.org/10.1109/ICDE.2016.7498408

[16] Edward Fredkin. 1960. Trie memory. *Commun. ACM* 3 (1960), 490–499.

[17] Chongming Gao, Zhong Zhang, Chen Huang, Hongzhi Yin, Qinli Yang, and Junming Shao. 2020. Semantic trajectory representation and retrieval via hierarchical embedding. *Inf. Sci.* 538 (2020), 176–192. https://doi.org/10.1016/j.ins.2020.05.107

[18] Antonin Guttman. 1984. R-Trees: A Dynamic Index Structure for Spatial Searching. In *SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, USA, June 18-21, 1984*, Beatrice Yormark (Ed.). ACM Press, 47–57. https://doi.org/10.1145/602259.602266

[19] Danlei Hu, Lu Chen, Hanxi Fang, Ziquan Fang, Tianyi Li, and Yunjun Gao. 2023. Spatio-Temporal Trajectory Similarity Measures: A Comprehensive Survey and Quantitative Study. *CoRR* abs/2303.05012 (2023). https://doi.org/10.48550/arXiv.2303.05012

[20] Hamza Issa and Maria Luisa Damiani. 2016. Efficient Access to Temporally Overlaying Spatial and Textual Trajectories. In *IEEE 17th International Conference on Mobile Data Management, MDM 2016, Porto, Portugal, June 13-16, 2016*, Chi-Yin Chow, Prem Prakash Jayaraman, and Wei Wu (Eds.). IEEE Computer Society, 262–271. https://doi.org/10.1109/MDM.2016.47

[21] Shunsuke Kanda, Koh Takeuchi, Keisuke Fujii, and Yasuo Tabei. 2020. 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*. 518–529. https://doi.org/10.1145/3397536.3422210

[22] Eamonn J. Keogh. 2002. 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*. Morgan Kaufmann, 406–417. https://doi.org/10.1016/B978-155860869-6/50043-3

[23] Eleazar Leal, Le Gruenwald, Jianting Zhang, and Simin You. 2015. 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*. 461–469. https://doi.org/10.1109/BigData.2015.7363787

[24] Ruiyuan Li, Huajun He, Rubin Wang, Sijie Ruan, Tianfu He, Jie Bao, Junbo Zhang, Liang Hong, and Yu Zheng. 2023. TrajMesa: A Distributed NoSQL-Based Trajectory Data Management System. *IEEE Trans. Knowl. Data Eng.* 35, 1 (2023), 1013–1027. https://doi.org/10.1109/TKDE.2021.3079880

[25] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S. Jensen, and Wei Wei. 2018. Deep Representation Learning for Trajectory Similarity Computation. In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*. 617–628. https://doi.org/10.1109/ICDE.2018.00062

[26] An Liu, Yifan Zhang, Xiangliang Zhang, Guanfeng Liu, Yanan Zhang, Zhixu Li, Lei Zhao, Qing Li, and Xiaofang Zhou. 2022. Representation Learning With Multi-Level Attention for Activity Trajectory Similarity Computation. *IEEE Trans. Knowl. Data Eng.* 34, 5 (2022), 2387–2400. https://doi.org/10.1109/TKDE.2020.3010022

[27] Kuien Liu, Bin Yang, Shuo Shang, Yaguang Li, and Zhiming Ding. 2013. 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*. IEEE Computer Society, 335–337. https://doi.org/10.1109/MDM.2013.49

[28] Zihan Luo, Lei Li, Mengxuan Zhang, Wen Hua, Yehong Xu, and Xiaofang Zhou. 2022. Diversified Top-k Route Planning in Road Network. *Proc. VLDB Endow.* 15, 11 (2022), 3199–3212. https://www.vldb.org/pvldb/vol15/p3199-luo.pdf

[29] Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. 2014. Approximate nearest neighbor algorithm based on navigable small world graphs. *Inf. Syst.* 45 (2014), 61–68. https://doi.org/10.1016/j.is.2013.10.006

[30] Yury A. Malkov and Dmitry A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (2020), 824–836. https://doi.org/10.1109/TPAMI.2018.2889473

[31] Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, and Theo Vassilakis. 2010. Dremel: Interactive Analysis of Web-Scale Datasets. *Proc. VLDB Endow.* 3, 1 (2010), 330–339. https://doi.org/10.14778/1920841.1920886

[32] Mario A. Nascimento and Jefferson R. O. Silva. 1998. Towards historical R-trees. In *Proceedings of the 1998 ACM symposium on Applied Computing, SAC'98, Atlanta, GA, USA, February 27 - March 1, 1998*, K. M. George and Gary B. Lamont (Eds.). ACM, 235–240. https://doi.org/10.1145/330560.330692

[33] Themis Palpanas. 2017. The Parallel and Distributed Future of Data Series Mining. *2017 International Conference on High Performance Computing & Simulation (HPCS)* (2017), 916–920. https://api.semanticscholar.org/CorpusID:21597889

[34] Christine Parent, Stefano Spaccapietra, Chiara Renso, Gennady L. Andrienko, Natalia V. Andrienko, Vania Bogorny, Maria Luisa Damiani, Aris Gkoulalas-Divanis, José Antônio Fernandes de Macêdo, Nikos Pelekis, Yannis Theodoridis, and Zhixian Yan. 2013. Semantic trajectories modeling and analysis. *ACM Comput. Surv.* 45, 4 (2013), 42:1–42:32. https://doi.org/10.1145/2501654.2501656

[35] Dieter Pfoser, Christian S. Jensen, and Yannis Theodoridis. 2000. Novel Approaches to the Indexing of Moving Object Trajectories. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, Amr El Abbadi, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal, Nabil Kamel, Gunter Schlageter, and Kyu-Young Whang (Eds.). Morgan Kaufmann, 395–406. http://www.vldb.org/conf/2000/P395.pdf

[36] Jiwei Qin, Liangli Ma, and Qing Liu. 2019. DFTHR: A Distributed Framework for Trajectory Similarity Query Based on HBase and Redis. *Inf.* 10, 2 (2019), 77. https://doi.org/10.3390/info10020077

[37] João B. Rocha-Junior, Orestis Gkorgkas, Simon Jonassen, and Kjetil Nørvåg. 2011. Efficient Processing of Top-k Spatial Keyword Queries. In *Advances in Spatial and Temporal Databases - 12th International Symposium, SSTD 2011, Minneapolis, MN, USA, August 24-26, 2011, Proceedings (Lecture Notes in Computer Science)*, Dieter Pfoser, Yufei Tao, Kyriakos Mouratidis, Mario A. Nascimento, Mohamed F. Mokbel, Shashi Shekhar, and Yan Huang (Eds.), Vol. 6849. Springer, 205–222. https://doi.org/10.1007/978-3-642-22922-0_13

[38] ruijie tian, Huawei Zhai, Jiajun Li, weishi zhang, fei wang, and Huan Liu. 2022. A Distributed Framework for Large-Scale Semantic Trajectory Similarity Join. (8 2022). https://doi.org/10.36227/techrxiv.20473794.v1

[39] Tapan Sahni, Chinmay Chandak, Naveen Reddy Chedeti, and Manish Singh. 2017. 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*. IEEE, 548–553. https://doi.org/10.1109/COMSNETS.2017.7945451

[40] Shuo Shang, Ruogu Ding, Bo Yuan, Kexin Xie, Kai Zheng, and Panos Kalnis. 2012. User oriented trajectory search for trip recommendation. In *15th International Conference on Extending Database Technology, EDBT '12, Berlin, Germany, March 27-30, 2012, Proceedings*, Elke A. Rundensteiner, Volker Markl, Ioana Manolescu, Sihem Amer-Yahia, Felix Naumann, and Ismail Ari (Eds.). ACM, 156–167. https://doi.org/10.1145/2247596.2247616

[41] Zeyuan Shang, Guoliang Li, and Zhifeng Bao. 2018. DITA: Distributed In-Memory Trajectory Analytics. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein (Eds.). ACM, 725–740. https://doi.org/10.1145/3183713.3183743

[42] Han Su, Shuncheng Liu, Bolong Zheng, Xiaofang Zhou, and Kai Zheng. 2020. A survey of trajectory distance measures and performance evaluation. *VLDB J.* 29, 1 (2020), 3–32. https://doi.org/10.1007/s00778-019-00574-9

[43] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger (Eds.). 3104–3112. https://proceedings.neurips.cc/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html

[44] Yufei Tao and Dimitris Papadias. 2001. MV3R-Tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries. In *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*, Peter M. G. Apers, Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, Kotagiri Ramamohanarao, and Richard T. Snodgrass (Eds.). Morgan Kaufmann, 431–440. http://www.vldb.org/conf/2001/P431.pdf

[45] Fabio Valdés and Ralf Hartmut Güting. 2019. A framework for efficient multi-attribute movement data analysis. *VLDB J.* 28, 4 (2019), 427–449. https://doi.org/10.1007/s00778-018-0525-6

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[47] Michail Vlachos, Dimitrios Gunopulos, and George Kollios. 2002. Discovering Similar Multidimensional Trajectories. In *Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, February 26 - March 1, 2002*, Rakesh Agrawal and Klaus R. Dittrich (Eds.). IEEE Computer Society, 673–684. https://doi.org/10.1109/ICDE.2002.994784

[48] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, and Gao Cong. 2022. A Survey on Trajectory Data Management, Analytics, and Learning. *ACM Comput. Surv.* 54, 2 (2022), 39:1–39:36. https://doi.org/10.1145/3440207

[49] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, Timos Sellis, and Xiaolin Qin. 2019. Fast Large-Scale Trajectory Clustering. *Proc. VLDB Endow.* 13, 1 (2019), 29–42. https://doi.org/10.14778/3357377.3357380

[50] Dong Xie, Feifei Li, and Jeff M. Phillips. 2017. Distributed Trajectory Similarity Search. *Proc. VLDB Endow.* 10, 11 (2017), 1478–1489. https://doi.org/10.14778/3137628.3137655

[51] Djamel Edine Yagoubi, Reza Akbarinia, Florent Masseglia, and Themis Palpanas. 2017. DPiSAX: Massively Distributed Partitioned iSAX. *2017 IEEE International Conference on Data Mining (ICDM)* (2017), 1135–1140. https://api.semanticscholar.org/CorpusID:1234878

[52] Djamel Edine Yagoubi, Reza Akbarinia, Florent Masseglia, and Themis Palpanas. 2020. Massively Distributed Time Series Indexing and Querying. *IEEE Trans. Knowl. Data Eng.* 32, 1 (2020), 108–120. https://doi.org/10.1109/TKDE.2018.2880215

[53] Chengcheng Yang, Lisi Chen, Hao Wang, and Shuo Shang. 2021. 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*. AAAI Press, 689–696. https://ojs.aaai.org/index.php/AAAI/article/view/16149

[54] Peilun Yang, Hanchen Wang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. 2021. T3S: Effective Representation Learning for Trajectory Similarity Computation. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*. 2183–2188. https://doi.org/10.1109/ICDE51399.2021.00221

[55] Di Yao, Haonan Hu, Lun Du, Gao Cong, Shi Han, and Jingping Bi. 2022. 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*. 2275–2285. https://doi.org/10.1145/3534678.3539358

[56] Byoung-Kee Yi, H. V. Jagadish, and Christos Faloutsos. 1998. Efficient Retrieval of Similar Time Sequences Under Time Warping. In *Proceedings of the Fourteenth International Conference on Data Engineering, Orlando, Florida, USA, February 23-27, 1998*, Susan Darling Urban and Elisa Bertino (Eds.). IEEE Computer Society, 201–208. https://doi.org/10.1109/ICDE.1998.655778

[57] Haitao Yuan and Guoliang Li. 2019. 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*. 1262–1273. https://doi.org/10.1109/ICDE.2019.00115

[58] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. 2011. 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*. 316–324. https://doi.org/10.1145/2020408.2020462

[59] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. 2010. 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*. 99–108. https://doi.org/10.1145/1869790.1869807

[60] Matei A. Zaharia, Reynold Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph E. Gonzalez, Scott Shenker, and Ion Stoica. 2016. Apache Spark. *Commun. ACM* 59 (2016), 56 – 65.

[61] Hanyuan Zhang, Xinyu Zhang, Qize Jiang, Baihua Zheng, Zhenbang Sun, Weiwei Sun, and Changhu Wang. 2020. Trajectory Similarity Learning with Auxiliary Supervision and Optimal Matching. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*. 3209–3215. https://doi.org/10.24963/ijcai.2020/444

[62] Yifan Zhang, An Liu, Guanfeng Liu, Zhixu Li, and Qing Li. 2019. Deep Representation Learning of Activity Trajectory Similarity Computation. In *2019 IEEE International Conference on Web Services, ICWS 2019, Milan, Italy, July 8-13, 2019*, Elisa Bertino, Carl K. Chang, Peter Chen, Ernesto Damiani, Michael Goul, and Katsunori Oyama (Eds.). IEEE, 312–319. https://doi.org/10.1109/ICWS.2019.00059

[63] Zhigang Zhang, Xiaodong Qi, Yilin Wang, Cheqing Jin, Jiali Mao, and Aoying Zhou. 2019. Distributed top-k similarity query on big trajectory streams. *Frontiers Comput. Sci.* 13, 3 (2019), 647–664. https://doi.org/10.1007/s11704-018-7234-6

[64] Bolong Zheng, Lianggui Weng, Xi Zhao, Kai Zeng, Xiaofang Zhou, and Christian S. Jensen. 2021. 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*. IEEE, 708–719. https://doi.org/10.1109/ICDE51399.2021.00067

[65] Bolong Zheng, Nicholas Jing Yuan, Kai Zheng, Xing Xie, Shazia Wasim Sadiq, and Xiaofang Zhou. 2015. Approximate keyword search in semantic trajectory database. In *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, Johannes Gehrke, Wolfgang Lehner, Kyuseok Shim, Sang Kyun Cha, and Guy M. Lohman (Eds.). IEEE Computer Society, 975–986. https://doi.org/10.1109/ICDE.2015.7113349

[66] Kai Zheng, Shuo Shang, Nicholas Jing Yuan, and Yi Yang. 2013. Towards efficient search for activity trajectories. In *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*, Christian S. Jensen, Christopher M. Jermaine, and Xiaofang Zhou (Eds.). IEEE Computer Society, 230–241. https://doi.org/10.1109/ICDE.2013.6544828

[67] Kai Zheng, Bolong Zheng, Jiajie Xu, Guanfeng Liu, An Liu, and Zhixu Li. 2017. Popularity-aware spatial keyword search on activity trajectories. *World Wide Web* 20, 4 (2017), 749–773. https://doi.org/10.1007/s11280-016-0414-0

[68] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. 2008. Understanding mobility based on GPS data. In *UbiComp 2008: Ubiquitous Computing, 10th International Conference, UbiComp 2008, Seoul, Korea, September 21-24, 2008, Proceedings*. 312–321. https://doi.org/10.1145/1409635.1409677

[69] Yu Zheng, Xing Xie, and Wei-Ying Ma. 2010. GeoLife: A Collaborative Social Networking Service among User, Location and Trajectory. *IEEE Data Eng. Bull.* 33, 2 (2010), 32–39. http://sites.computer.org/debull/A10june/geolife.pdf

[70] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. 2009. 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*. 791–800. https://doi.org/10.1145/1526709.1526816

[71] Panfeng Zhou, Donghui Zhang, Betty Salzberg, Gene Cooperman, and George Kollios. 2005. Close pair queries in moving object databases. In *13th ACM International Workshop on Geographic Information Systems, ACM-GIS 2005, November 4-5, 2005, Bremen, Germany, Proceedings*, Cyrus Shahabi and Omar Boucelma (Eds.). ACM, 2–11. https://doi.org/10.1145/1097064.1097067

[72] Chenghao Zhu, Jiajie Xu, Chengfei Liu, Pengpeng Zhao, An Liu, and Lei Zhao. 2015. Efficient Trip Planning for Maximizing User Satisfaction. In *Database Systems for Advanced Applications - 20th International Conference, DASFAA 2015, Hanoi, Vietnam, April 20-23, 2015, Proceedings, Part I (Lecture Notes in Computer Science)*, Matthias Renz, Cyrus Shahabi, Xiaofang Zhou, and Muhammad Aamir Cheema (Eds.), Vol. 9049. Springer, 260–276.