Semantic Trajectory Compression via Multi-resolution Synchronization-based Clustering

Chongming Gao[a], Yi Zhao[a], Ruizhi Wu[a], Qinli Yang[a], Junming Shao[a,*]

[a] *Data Mining Lab, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China*

Abstract

With the pervasive use of location-aware devices and rapid development of location sensing technology, trajectory data has been generated in diverse fields. How to store and manage these large amounts of data is a non-trivial task, and thus trajectory compression has gained increasing attention in recent years. As traditional compression algorithms often treat trajectories as sequences of lines in geometric space, the global statistics and the semantics embedded in trajectories are not well considered. Inspired by the powerful concept of synchronization, in this paper we introduce a new semantic trajectory compression approach, called CascadeSync, to yield multi-resolution trajectory abstractions with semantic enrichment. The basic idea is to introduce a multi-resolution synchronization-based clustering model to produce semantic regions-of-interest (ROIs) in a hierarchical way. Specifically, by imposing constraints on points with semantic information in the interaction model, all neighboring points with similar semantics will group together automatically. Afterwards, each trajectory is compressed as a sequence of semantic ROIs and is further represented as a hierarchical ROI network. Beyond, we further extend our model on the data stream setting. The extensive experiments on synthetic data and four

*Corresponding author

Email addresses: chongming.gao@gmail.com (Chongming Gao), yizhao.joey@outlook.com (Yi Zhao), ruizhiwuuestc@gmail.com (Ruizhi Wu), qinli.yang@uestc.edu.cn (Qinli Yang), junmshao@uestc.edu.cn (Junming Shao)

URL: dm.uestc.edu.cn (Junming Shao)

real-world data sets have demonstrated the effectiveness and efficiency of our proposed model.

Keywords: Trajectory Compression, Semantic Enrichment, Clustering

1. Introduction

In recent years, more and more trajectory data has been generated in diverse fields. Massive amount of these data often contains a variety of patterns, and plays an important role in a wide range of applications, such as location recommendation, on-board diagnostics and movement behavior analysis. However, how to store and manage these large amounts of data is a non-trivial task. To deal with this problem, trajectory compression has attracted a lot of attention in recent years. To date, many trajectory compression algorithms have been proposed, which can be broadly divided into two categories: traditional compression in free space ([1, 2, 3, 4, 5])and compression with semantic constraints ([6, 7, 8, 9, 10, 11]). The philosophy of the first class is to use fewer lines or paths to replace the original lines or paths on every trajectory, so as to reduce the information redundancy. Although those methods often have a good compression ratio, the real-world context often cannot be well-preserved. For instance, in route-based recommendation systems, the recommended route should be constrained by actual roads, instead of simple unconstrained lines. Meanwhile, the compressed points (or lines) should cover the important point-of-interest (POI) of cities, instead of merely taking account of the points extracted from raw trajectories.

To tackle this issue, a myriad of semantic trajectory compression algorithms have been proposed in recent years. The basic idea is to compress trajectory with some semantic information, which is usually either a network extracted from urban geographic infrastructure elements such as street, point-of-interest (POI), or activities beyond pure geometry, such as walking, non-walking segment ([8]) and *stay point* ([11]). However, the performances of these methods are highly dependent of the amount of accessible priors, i.e., if the roadmap

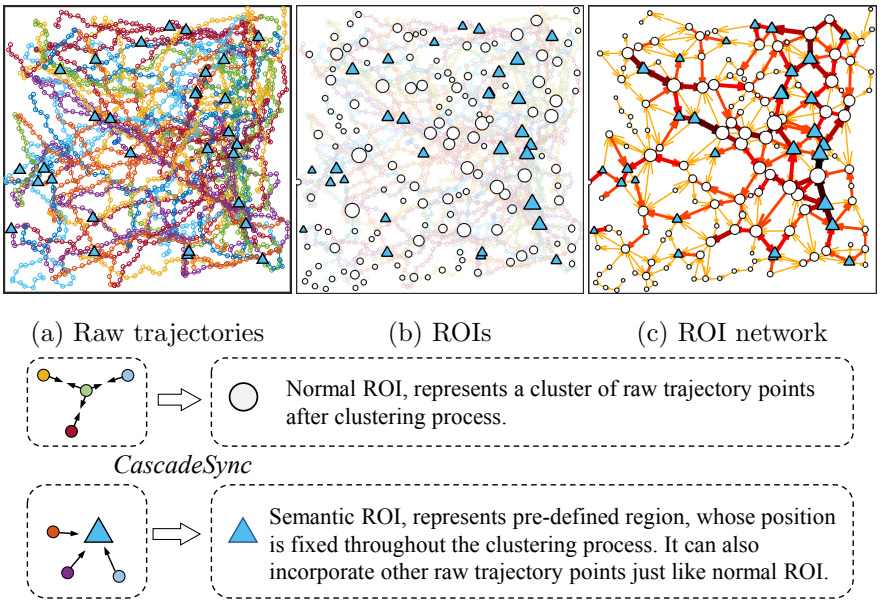


Figure 1: Illustration of semantic trajectory compression via *CascadeSync*. (a) The 80 trajectories with raw GPS points, and the 30 blue triangles indicate the GPS points with semantic information. (b) The resulting ROIs via contrained synchronization-based clustering. Here each white circle indicates one cluster (i.e., normal ROI), and each blue triangle means one semantic ROI. (c) Trajectory data is finally represented as a ROI network. (d) The normal ROI and semantic ROI.

or POI network is sparse or has a lot of missing values, the representation of compressed trajectories would be largely influenced.

In this paper, we introduce a multi-resolution synchronization-based clustering model, CascadeSync, to provide a new perspective to compress trajectory, and enrich compressed trajectories with semantic information. Moreover, we introduce a simple yet effective way for trajectory compression and retrieval in a stream setting. To evaluate the performance of proposed method, we conduct experiments on both synthetic data and four real-world data sets, and compare its performance to several baselines. We will see that our model has several desirable properties, but let us first illustrate its basic idea.

1.1. Basic Idea

Unlike traditional methods to extract key points in each trajectory independently or assign GPS points to a given semantic network, CascadeSync aims to find important regions that allows representing all raw trajectory points (i.e., global structure information). These regions, denoted as regions of interest (ROIs), are formed by aggregating surrounding data points based on a synchronization-based clustering model. Due to the powerful concept of synchronization, these GPS points in trajectory data can be clustered from fined-grained to coarse-grained levels. With the derived ROIs, the trajectory data can be represented as a set of multi-resolution ROI-based networks. Beyond, if some domain knowledge (e.g., the semantic information for some GPS points) is available, it can be well integrated into the synchronization-based clustering process. For example, we assume that there are a bunch of points with semantic information, such as landmark building, road intersections or places where big events occurred. The positions of those important points or regions will be fixed during the synchronization-based clustering process, and they attract neighboring points or regions to group together to form semantic clusters, which are referred to as semantic ROIs. In this way, both global structure information of GPS points and semantic information are well integrated in our compression model.

For illustration, Fig. 1 shows a toy example on a synthetic data set with 80 trajectories. For all these trajectories, suppose that there are 30 points with semantic information, which are indicated as blue triangles in Fig. 1(a). For compression, during the clustering process, these points with semantic information are fixed, and all points (including the fixed points) are mutually interacted, and finally similar points will tend to group (i.e., synchronize) together. Since the points with semantic information are fixed, after the synchronzation process, they will attract its neighboring GPS points to group together and thus form semantic regions-of-interest (ROIs). For other GPS points, if the neighboring points have no fixed points with semantic information, they are naturally grouped together to form ROIs (without semantic information) (Fig. 1 (b)).

Fig. 1 (d) further illustrates the two resulting ROIs: normal ROI and semantic ROI. Afterwards, with these ROIs, each trajectory can be represented as a sequence of ROIs, and the whole trajectory data is further compressed as a ROI network (Fig. 1 (c)). With the ROI network, the global trajectory moving pattern can be well-preserved. In addition, due to the desirable properties of synchronization-based clustering, these ROIs, can be further compressed into a higher level and yield a more compact trajectory representation upon the user's applications. Building upon the multi-resolution ROI network representation, the global statistics of trajectory data can be extracted and thus facilitate the downstream trajectory mining tasks.

1.2. Contributions

Building upon our multi-resolution ROI representation, the main contributions of this work are summarized as follows.

- **An intuitive Semantic Trajectory Compression Model.** We propose a semantic trajectory compression model by considering both global trajectory structure information and available semantic information. This method, provides a new perspective to compress trajectories in free space and with semantic requirements.

- **Multi-resolution Synchronization-based Representation.** Inspired by the powerful concept of synchronization, CascadeSync allows offering a promising way to compress trajectory data. More importantly, it supports a multi-resolution compression and the semantic information can be naturally integrated.

- **Online Trajectory Compression.** An online model is further proposed to support efficient compression and retrieval, where two operations: *Merge* and *Split* are defined on ROI networks.

- **High Performance.** The empirical experiments show that our proposed method has strong representative capability and is thousands of times

faster than the baseline algorithms. Besides, the desirable semantic information is also well-preserved and a global profile of urban traffic is established. The spatiotemporal retrieval shows great prospects for more applications.

The rest of paper is organized as follows. Section 2 discusses related work. Section 3 gives some notations and the problem statement. Section 4 elaborates CascadeSync algorithm. Section 5 extends the proposed compression algorithm to stream data. Section 6 presents our extensive experiments. Finally, we conclude our work in Section 7.

2. Related Work

During the past decade, many trajectory compression algorithms have been proposed. In the following, we will give some major works in the existing literature. Beyond, the works of synchronization-based clustering are briefly given.

2.1. Traditional Trajectory Compression

As a reaction to increasing volumes of trajectory data, a lot of trajectory compressiontechnologies have been developed, Sun et al. [12] give a comprehensive overview of trajectory compression algorithms. The early work treats trajectories as sequences of straight lines, so the compression is cast to a line simplification problem in pure geometric space. Douglas-Peucker algorithm is a classical model, which recursively selects the point whose perpendicular distance is greater than a given error bound, until all reserved points meet the condition [1]. Lee et al. [13] utilize the concept of minimal description length (MDL) to pursue the compressed trajectory, which aims to find an optimal trade-off between conciseness and accuracy. The Spatial QUalIty Simplification Heuristic (SQUISH) method based on the priority queue data structure [14], which prioritizes the most important points in a trajectory stream. It uses local optimization to select the best subset of points and permanently removes redundant or insignificant points from the original GPS trajectory. Afterwards, Muckell

et al. [3] present a new version of SQUISH, called SQUISH-E (Spatial QUalIty Simplification Heuristic-Extended), which has the flexibility of tuning compression with respect to compression ratio and error. Some work [15, 16] replaces the metric from spatial error to Synchronized Euclidean Distance (SED) to consider the temporal information. In addition, some methods are also proposed to compress trajectory in an online setting. For instance, Dead Reckoning algorithm estimates the successor point on the go through the current point and its velocity [2]. Liu et al. [4] present the Bounded Quadrant System (BQS) to select points by calculating distances of new point to a special line maintained online. Lin et al. [5] propose a one-pass strategy that process each point in a trajectory only once to pursue an error bound. However, those online algorithms only output a reduced trajectory data set, which still not fill the gap between the disordered raw data and the time-awared query for locations and trajectories. More importantly, these methods are not applicable when introducing constraints such as roadmap or POI network in real world. Those algorithms can be served as downsampling methods for portable mobile devices to reduce the transmitting cost. However, they are not suitable for establishing a holistic urban movement profile.

2.2. Semantic Trajectory Compression

There is a branch of approaches put forward to compress trajectories with semantic information. Schmid et al. [6] introduce a novel representation to replace raw trajectory data with the nodes and edges in a network enriched with urban semantic information. Following the same idea, Liu et al. [7] further propose velocity-based and beacon-based trajectory symbolization, to represent all trajectories using fewer reference points at the same time. Some methods also consider the behavior patterns contained in trajectories. For example, Chen et al. [8] partition a trajectory into walking and non-walking segments to maintain both the skeleton and semantic meanings of trajectories. Moreover, there are some works arguing that the direction of moving object is important to preserve [17, 18]. Another notable work is proposed by Song et al. [9], which

presents a framework *PRESS* to split trajectories in spatial representation and temporal representation, then a hybrid spatial compression (HSC) algorithm and an error-bounded temporal compression (BTC) algorithm are proposed, respectively. Besides, There are a myriad of works emphasizing when mining the trajectory of vehicles generated in urban network, every point in trajectories should be projected onto road, a.k.a. Map-matching [19, 20, 21, 22]. For more details, please refer to the excellent surveys [23, 24]. However, for most existing approaches, they assume that the semantic information is completely available as priors. Once the semantic priors are missing or sparse, for example, the roadmap is incomplete, then the performance will be greatly affected.

2.3. Synchronization-based Clustering

Our work is also highly related to synchronization-based data mining [25]. Synchronization phenomena is prevalent in physical, biological, chemical, and social systems. It comes from the example of typical synchronous flashing of swarms of fireflies in South Asia forests [26]. In the beginning, some fireflies start emitting flashes of light incoherently, but after some time all the fireflies are flashing with the common rhythm with mutual influence. Currently, many synchronization-based models have been proposed in diverse fields [27, 28, 25, 29, 30, 31, 32, 33, 34, 35]. The extensive Kuramoto model [36], [37] is one of the most successful approaches to exploring synchronization phenomena. Seliger et al. [27] discuss mechanisms of learning and plasticity in networks of phase oscillators through a generalized Kuramoto model. Arenas et al. [28] apply the Kuramoto model for network analysis, and study the relationship between topological scales and dynamic time scales in complex networks. This analysis provides a useful connection between synchronization dynamics, network topology and spectral graph analysis. From bioinformatics, Kim et.al. [38] propose a strategy to find groups of genes by analyzing the cell cycle specific gene expression with a modified Kuramoto model. Similarly, the co-clusters with correlated genes and conditions are yielded based on two-sided interaction model [34]. Recently, many synchronization-based clustering algorithms [25],

Table 1: Notations.

Notation	Description
$\mathcal{T}$	Trajectory dataset, each $T_i \in \mathcal{T}$ is a trajectory, defined in (1).
$\mathcal{P}$	GPS point set, each $P_i \in \mathcal{P}$ is a GPS point.
$\mathcal{ROI}$	ROI set, each $ROI_i \in \mathcal{ROI}$ is a node on ROI network.
$\mathcal{E}$	Edge set, each $e_i \in \mathcal{E}$ is an edge of ROI network.
ζ	Representative error bound, defined in (2).
ϵ	The radius of interaction range, defined in (3).

[33], [39], [31], [29] have been proposed by reformulating the Kuramoto model. For example, Shao et al. [25] propose a new synchronization-based clustering algorithm by introducing local synchronization and minimum description length principle. In contrast to other algorithms, synchronization-based clustering algorithms have many benefits which allow identifying high-quality clusterings and are robust to noisy objects or outliers.

Motivated by previous studies, in this paper, we view trajectory compression from a dynamic perspective, and extract multi-resolution trajectory data abstractions based on synchronization principle. To the best of our knowledge, it is the first time to apply the concept of synchronization for data compression.

3. Preliminary

In this section, we will introduce the semantic trajectory compression problem. Before that, we first give some notations used in the following sections, which is listed in Table 1.

3.1. Trajectory and Trajectory Compression

The most prevalent data format of trajectories, usually collected by GPS devices, are the temporal sequence of latitude/longitude coordinates as follows.

$$T = \{\langle s_1, s_2, \cdots, s_n \rangle \, | s_i = (P_i, t_i)\}, \tag{1}$$

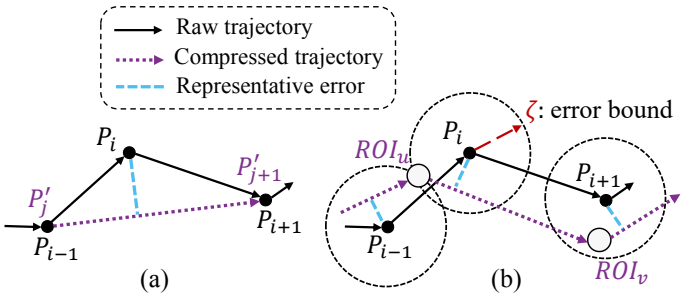


Figure 2: Illustration of the representative error and error bound. (a) Traditional representation, where the solid line sequence $\langle P_{i-1}, P_i, P_{i+1}\rangle$ is the original trajectory, and the dash line sequence $\langle P_{i-1}, P_{i+1}\rangle$ is the compressed trajectory. (b) Representation using ROI, where sequence $\langle ROI_u, ROI_v\rangle$ is the compressed trajectory. There is at least one ROI existing in ζ-radius circle of each raw point, so that equation (2) can be satisfied).

where the $P_i = (x_i, y_i)$ is a (*latitude, longitude*) pair representing a GPS coordinate, representing a sample point on the map. t_i is the time stamp of a point P_i, and n is the length of the trajectory T.

The traditional trajectory compression problem is to use fewer points to replace original points of each trajectory while most trajectory information needs to be preserved. Given a trajectory T, the compressed trajectory is denoted as $T' = \{\langle P'_1, P'_2, \cdots, P'_m\rangle; , m \leq n)\}$. In order to make a compromise between compression ratio and representative error, we introduce the representative error bound to guide the compression.

Definition 3.1 (Representative Error Bound)**.** Given a trajectory T and a compression algorithm $\mathcal{A}$ that produces the corresponding compressed trajectory T', the algorithm $\mathcal{A}$ is bounded by error bound ζ, if for each $P_i \in T$, there exists a point $P'_j \in T'$ with:

$$distance\left(P_i, \mathcal{L}(P'_j, P'_{j+1})\right) \leq \zeta, \tag{2}$$

where $\mathcal{L}(P'_j, P'_{j+1})$ is the straight line that passes the two points (P'_j, P'_{j+1}).

The distance between the point P_i and $\mathcal{L}$ is shown as light blue dash line in Fig. 2(a). Note that most trajectory compression algorithms are bounded by ζ.

3.2. Semantic Region of Interest

Precisely modeling location identities in trajectories data is of significant importance in location-based service. In many applications, locations represented by GPS coordinates are not adequate to support effective mining tasks. To extract patterns from human mobility, the locations representing high level activities are required to describe the profiles of moving objects. After enriched with domain knowledge, a trajectory can be transformed to sequence such as residence, bus station and school. Therefore, the queries referring to the semantic meanings such as "return objects that stop at the building for half an hour and pass the bus station around eight o'clock" are naturally supported.

Up to now, there are a myriad of algorithms devoted to automatically detecting and inferring the significant places for pattern learning [40, 41], recommendation [42, 43, 44] and route classification [45]. However, the basic idea is to simply split the map into multiple grids or group the raw GPS points to clusters, which fail to associate the real-world semantics into the detected places. As mentioned in [6], the semantics should be explicitly defined as a priori. In their work, important urban infrastructures such as hotels, touristic places and major intersections are defined as nodes and all nodes expand a network. We refer to those nodes as semantic regions of interests (ROIs). By clustering and map-matching techniques, all raw GPS points are projected to those semantic ROIs.

However, it is noteworthy that the capability of representation and performance of compression hinge on the quality of predetermined semantic ROIs. When semantic ROIs are insufficient to cover each corner of city, i.e., the prior is sparse, a profound representative error would be introduced into the compressed results. That is the motivation of our work, where we aim to introduce a new perspective to compress trajectory with limited semantic information. Specifically, we will propose a model to identify important ROI at appropriate position, which will be introduced in next section.

3.3. Semantic Trajectory Compression with ROIs

As aforementioned, trajectory compression should make use of real-world semantic information, such as roadmap and POI network, to yield more meaningful compression results. Without loss of generality, these semantic information can be expressed by the regions-of-interest (ROIs), which are a set of fixed regions on the map with specific semantics. The task of semantic trajectory compression is find an algorithm $\mathcal{A}$, which is error bounded by ζ. For each raw trajectory $T \in \mathcal{T}$, apply $\mathcal{A}$ to generate a sequence of ROIs to represent T as $T' = \{\langle ROI_1, ROI_2, \cdots, ROI_m \rangle, m \leq n)\}$.

In order to satisfy the error bound condition defined in (2), the representation radius, i.e., the distance between point P_i and corresponding ROI, should be bounded in ζ. In other words, for each point there should be at least one ROI in its ζ-radius circle. Fig. 2(b) illustrates a trajectory segment $\langle P_i - 1, P_i, P_i + 1 \rangle$ and its compressed version $\langle ROI_u, ROI_v \rangle$.

4. CascadeSync: A Multi-resolution Clustering Model for Trajectory Compression

In this section, we present CascadeSync, a new semantic trajectory compression framework built upon multi-resolution constrained synchronization-based clustering.

4.1. Region-of-Interest (ROI) Detection via Synchronization-based Clustering

As GPS-based trajectory data are represented by a set of points, it is not a good way to mine trajectory data on these points directly, for different trajectories often have varying lengths and different sampling rates. One intuitive way is to group all these points into many regions with semantics. However, traditional clustering methods are not suitable to this task. For instance, *k-means* style clustering algorithms need to explicitly select the cluster number k, and the resulting clusters are not evenly distributed, hence the representative error can not be bounded in ζ. Unlike traditional clustering algorithms,

synchronization-based clustering approaches view each data object as a phase oscillator, and simulate the dynamical behaviors of the objects over time. By the interaction with similar objects, the phase of an object gradually aligns with its adjacent objects, resulting in a non-linear object movement driven by the local cluster structure. Finally, the objects in a cluster are synchronized together and have the same phase. Therefore, synchronization-based clusters can identify clusters driven by the local data structure, and more importantly, the global data structure can be well-preserved with synchronized objects.

Typically, a synchronization-based clustering algorithm needs three definitions to simulate a dynamic clustering process: First, a parameter ϵ specifying the interaction range among objects, second, the interaction model for clustering, and finally, a stopping criterion to terminate dynamic clustering. Our approach follows and extends the synchronization-based clustering underlying the algorithm SYNC, presented and discussed in full detail in [46].

Definition 4.1 (ϵ-Range Neighborhood)**.** Given a GPS data set $\mathcal{P} \subset \Re^n$, the ϵ-range neighborhood of a GPS point $p \in \mathcal{P}$, denoted as $N_\epsilon(p)$, is defined as:

$$N_\epsilon(p) = \{q \big| dist(p,q) \leq \epsilon\}, \tag{3}$$

where $dist(p,q)$ is a metric distance function. Euclidean distance is used in this study.

Definition 4.2 (Interaction Model)**.** Let p be a GPS point on the map. With an ϵ-range neighborhood interaction, the dynamics of the value of the point p is defined as:

$$p(t+1) = p(t) + \frac{1}{|N_\epsilon(p)|} \cdot \sum_{q \in N_\epsilon(p)} \sin(q(t) - p(t)), \tag{4}$$

where $sin(x)$ is the coupling function, applying to every dimension of vector x. $p(t+1)$ is the renewal position of $p(t)$ during the dynamic clustering, $t \in \{0, \ldots, T\}$ denotes the iteration step. Note all dimensions are normalized to $[0, \pi/2]$.

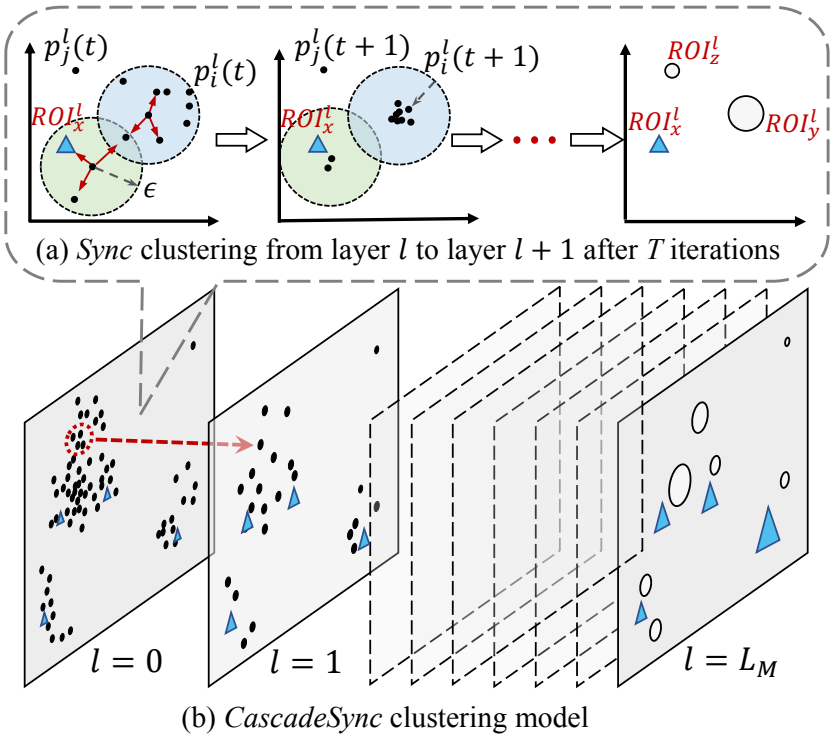


Figure 3: Illustration of CascadeSync algorithm. The blue triangles are semantic ROIs, which stay fixed throughout the interaction. The dots on layer $l = 0$ are raw GPS points, and the circles on layer $l \geq 1$ are ROIs generated by points on the last layer.

Definition 4.3 (Cluster Order Parameter)**.** The cluster order parameter r is used to terminate the dynamic clustering by investigating the degree of local synchronization, which is defined as:

$$r(t) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{|N_\epsilon(p(t))|} \sum_{q \in N_\epsilon(p)} \mathbf{e}^{-||q(t)-p(t)||} \tag{5}$$

The dynamic clustering terminates when $r(t)$ converges, which indicates local phase coherence. At this moment, all cluster points have the same location.

For synchronization-based dynamic clustering, each point is viewed as a phase oscillator and has its own phase (feature vector) at the beginning. As time evolves, each point interacts with its ϵ-range neighborhood according to the interaction model (Eq. (4)). As illustrated in Fig. 3(a), each point in-

teracts with its neighborhood points, and finally all locally similar points are synchronized together and form clusters. For examples, point p_i^l is influenced by its neighbors, and finally forms ROI_y^l together with other points in the same region. Meanwhile, since potential outlier points do not interact with other points during the dynamic clustering, they maintain their original values and thus are easily identified. These outliers are also treated as ROIs since they are important to describe some distinctive trajectories. For instance, p_j^l has no near point to interact with, and it is finally represented as ROI_z^l.

The salient feature of synchronization-based clustering is its dynamic property. During the process of interactions, the value of each dimension of a given point changes in a non-linear way driven by the local data structure, and finally the feature vectors of points in a cluster will become the same. More importantly, the derived synchronized ROIs, can be viewed as new points to form a new data set. The new data set, well preserves the original data structure. Therefore, the powerful concept of synchronization supports a natural hierarchical clustering.

4.2. Multi-resolution Network Modeling with Semantic Enrichment

It is important to note that with derived ROIs, the new data set can be further compressed with synchronization-based clustering at a higher level. Therefore, for our trajectory data, we extend Sync to multi-resolution data representation, which is called CascadeSync. The basic idea is quite intuitive. For the first step, we cluster all trajectory data points with a small interaction range ϵ, which usually results in numerous small-size clusters. Since points in a same cluster have synchronized together, we can use the synchronized point to characterize the whole cluster objects. Therefore, a new data set including all these synchronized points can be generated, and cluster again with a much larger interaction range ϵ. However, for a new data set, since each point in the new data set characterizes different number of points in the previous layer of

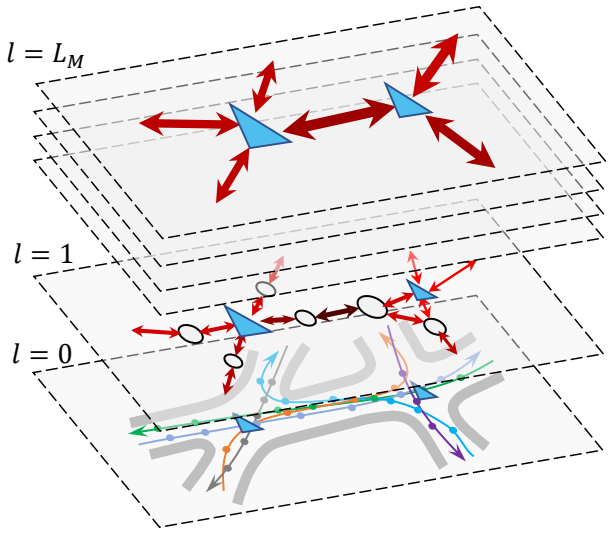


Figure 4: Illustration of hierarchical ROI network, where the size of ROI is proportional to the point weight. The width and color shade of edges are proportional to the number of trajectories passing through the two ending ROIs.

data, the interaction model should consider the weight of each point, which is:

$$p^l(t+1) = p^l(t) + \frac{1}{\sum_{q^l \in N_{\epsilon^l}(p^l)} w_{q^l}} \cdot \sum_{q^l \in N_{\epsilon^l}(p^l)} w_{q^l} \sin(q^l(t) - p^l(t)), \tag{6}$$

where the weight w_{q^l} is the number of points that are represented by the prototype q^l on the layer $l-1$. And ϵ^l is the interaction range of layer l, which is manually set by the user according to the application scenario. In the study, we initiate ϵ to be $0.005 \times (L+W)/2$, and add ϵ by $0.005 \times (L+W)/2$ with the number of layer l increasing, where L and W are the length and width of the map.

For illustration, Fig. 3(b) gives a toy example. In fact, CascadeSync not only supports a hierarchical data representation, the speedup of synchronization process is another desirable property. The reason is with small interaction range, CascadeSync is easier to converge. Although more clusters are generated, they are viewed as new objects, and can be further clustered efficiently.

Semantic Enrichment. In previous sections, we focus on the synchronization-

based trajectory compression. However, as mentioned in the introduction section, the semantic information is an important property, which needs to be considered for trajectory compression. In real-world scenarios, the semantic information is usually embodied in a collection of fixed points or regions on map, such as the intersections and turning points on roadmap, which are drawn as light blue triangles in Fig. 3.

There is an intuitive way to plug those predefined semantic ROIs in our CascadeSync model. Specifically, here we make the positions of those ROIs to be fixed during the interaction process. Since each point will interact with its neighboring points, and thus all surrounding GPS points will move to these fixed points to form semantic ROIs. If there are no fixed points for some GPS points, near GPS points will synchronize together to form normal ROIs. After all raw GPS points are represented by normal ROIs or semantic ROIs, each trajectory is represented as a new path with these resulting ROIs. To further explore the statistical information of trajectories contained in ROIs, we further define the hierarchical ROI network.

Definition 4.4 (Hierarchical ROI Network)**.** Given a trajectory data set, the ROI network is represented as a multi-layer graph $G(\mathcal{V}, \mathcal{E})$ with every layer being $G^l(\mathcal{V}^l, \mathcal{E}^l)$, where the node set $\mathcal{V}^l$ contains all ROIs on the layer l. $\mathcal{E}^l$ is edge set of layer l. And there is an edge $e^l_{uv} \in \mathcal{E}^l$ connecting nodes (ROI^l_u, ROI^l_v) if there exists any GPS coordinate pair (P_i, P_j) as a trajectory segment, with P_i represented by ROI^l_u and P_j represented by ROI^l_v on layer l.

We illustrate a hierarchical ROI network in Fig. 4, where the trajectory data is visualized from fine-grained to rough-grained. The ROI network provides a compact representation to characterize a trajectory. Each trajectory could be expressed as a temporal sequence of ROIs on a given ROI network. In addition, from each ROI, all trajectories that passed this region are recorded. Therefore, a wide range of statistical information, e.g., visiting time distribution, staying time distribution and passing direction, can be summarized for the ROI. For instance, we illustrate a toy example in Fig. 5, in which the raw data and

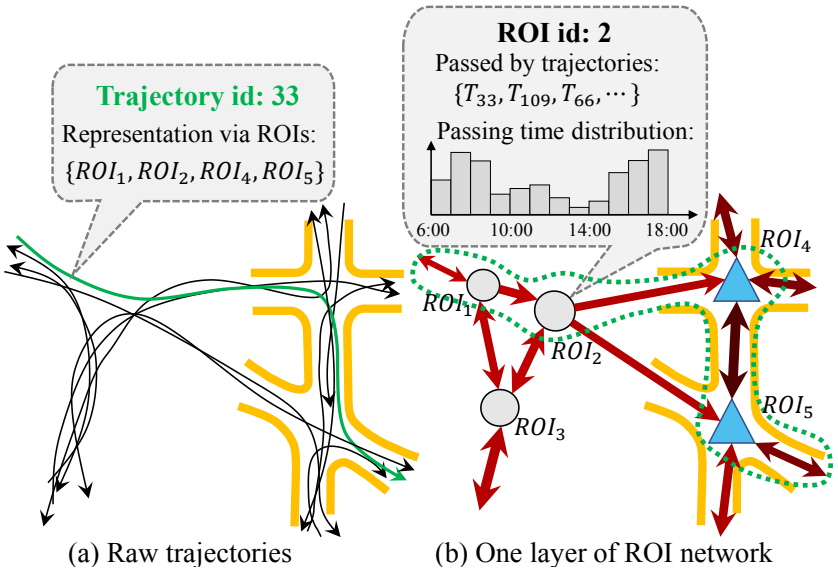


Figure 5: ROI network in an incomplete roadmap. Where a specific trajectory T_{33} is selected and compressed to four ROIs: $\langle ROI_1, ROI_2, ROI_4, ROI_5 \rangle$. The green dash line encloses the four ROIs on ROI network. And the basic statistics of ROI_2 is illustrated.

one layer of ROI network is shown. In Fig. 5(a), a specific trajectory T_{33} is selected, from one layer of ROI network shown in Fig. 5(b), T_{33} is compressed and represented by four ROIs $\langle ROI_1, ROI_2, ROI_4, ROI_5 \rangle$. Besides, the basic statistical information of ROI_2 is shown. The traffic patterns of this region can be evaluated from the passing time distribution. Furthermore, some downstream mining tasks, such as trajectory retrieval and frequent pattern mining, could work on such representation directly.

4.3. Time Complexity Analysis

The time complexity of CascadeSync is $\mathcal{O}(L \times T \times N_l \log N_l)$, T is the time steps in each round and L is number of layers in CascadeSync. Usually, L is small with $L \leq 20$ in practice. N_l is the number of points, which reduced exponentially over the layer l. Finally, the pseudocode of our approach is illustrated in Algorithm 1.

Algorithm 1: Semantic Trajectory Compression via CascasdeSync

Input : Trajectory dataset $\mathcal{T}$; Semantic ROI set $\mathcal{ROI}_s$

Output : ROI network;

Parameter: Initial value ϵ_0; Incremental change $\triangle\epsilon$; Maximal layer $\mathcal{L}_M$

$\mathcal{P}_0 = \mathcal{T} \bigcup \mathcal{ROI}_s$; ▷ The union of raw GPS points set and semantic ROI set.

$\mathcal{P}_0 = \texttt{Norm}\ (\mathcal{P}_0)$; ▷ Normalized each dimension to $[0, \pi/2]$.

$layer = 0; \epsilon = \epsilon_0$;

while $layer \leq \mathcal{L}_M$ **do**

  $\mathcal{P}_{layer+1} = \texttt{Sync}(\mathcal{P}_{layer}, \mathcal{ROI}_s, \epsilon)$; ▷ Fig. 3

  $layer = layer + 1; \epsilon = \epsilon + \triangle\epsilon$;

end

Regions-of-interest set: $\mathcal{ROI} = \mathcal{P}_{layer}$;

ROI network edge set $\mathcal{E} = \texttt{ROINetworkConstructor}\ (\mathcal{ROI}, \mathcal{T})$; ▷ Fig. 4,5

Function $\texttt{Sync}(\mathcal{P}, \mathcal{ROI}_s, \epsilon)$**:**

  $t = 0$;

  while *TRUE* **do**

    foreach *point* $p_i(t) \in \mathcal{P}$ AND $p_i(t) \notin \mathcal{ROI}_s$ **do**

      Search its ϵ-range neighbors $N_\epsilon(p_i(t))$ with Eq.(3);

      foreach *neighbors* $q(t) \in N_\epsilon(p_i(t))$ **do**

        Compute $p_i(t+1)$ with Eq.(4);

      end

    end

    Compute order parameter $r(t)$ with Eq.(5);

    if $r(t)$ converges **then**

      return $\{p_1(t), \cdots, p_N(t)\}$;

    end

    $t = t + 1$;

  end

end

5. Trajectory Stream Compression and Retrieval

Since trajectory data are constantly generated and collected in overwhelming speed in the real world, it is unrealistic to represent and store all trajectories in one ROI network. Here, we extend our model on the stream setting, which aims to compress and store the trajectories over time with a simple strategy.

5.1. Trajectory Stream Compression

Trajectories are generated by human and vehicles without a moment's pause. The mounting data would be stored and represented in many chunks. A natural

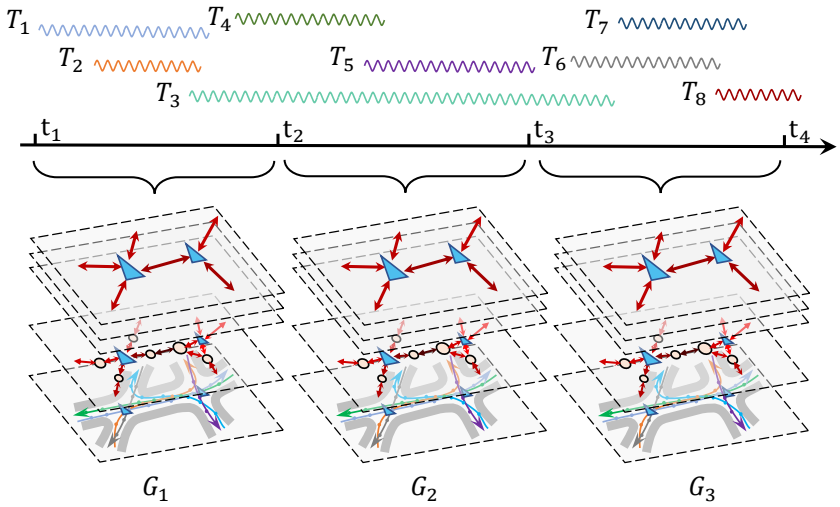


Figure 6: ROI networks maintenance in trajectory stream.

idea is to partition data by time, e.g., a week or a month per chunk. Fig. 6 illustrates the ROI networks generated by CascadeSync in three period. The partitioning strategy is simple: Firstly we select appropriate window size for time partition, which is depended on the real life scenario. For each window, we apply CascadeSync to all GPS points that are generated in the period that this window represents. For example, GPS points of trajectory T_1, T_2 and part of T_3 and T_4 are collected and fed in CascadeSync to construct a hierarchical ROI network G_1. The rest GPS points of T_3 and T_4 are generated in the next period and thus represented by G_2 and G_3.

To keep the integrity of the trajectory and support trajectory visualization, separated ROI networks should be capable of reassembling appropriately. In this respect, we define two operations, *Merge* and *Split*, on the ROI networks to support the mixture and split of data.

ROI Networks Aggregation. When the time range for a given query spans more than one time window, we need to merge all corresponding ROI networks together. Here we develop an operation on ROI networks, called *Merge*, which aims at connecting two ROI networks into a single ROI network. Fig. 7 il-

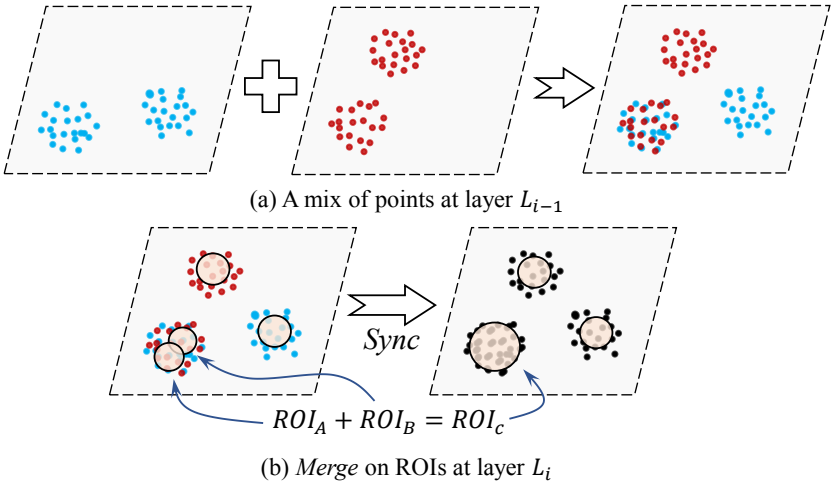
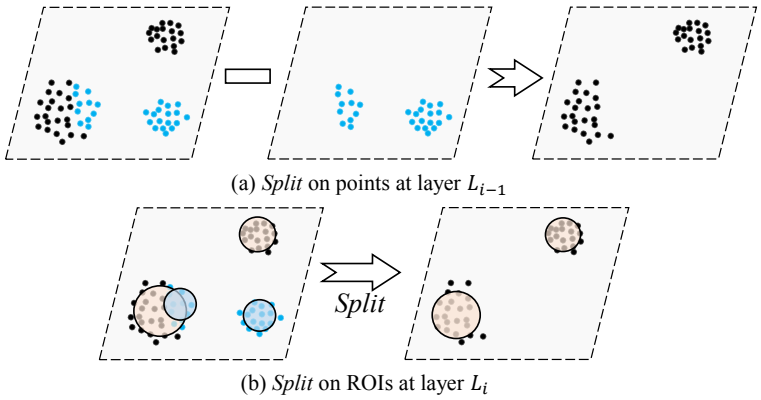


Figure 7: ROI aggregation on two hierarchical ROI networks.

lustrates how it works. For each layer L_i, the initial points, i.e., original GPS points or ROIs generated in layer L_{i-1}, from two sources are mixed together, which is shown in Fig. 7(a). The ROIs distributed in a small region (with radius of ϵ) should be merged to one ROI, as they are comprised of points in the same region. For example, as the red points and blue points mixed at layer L_{i-1}, the representative region ROI_A and ROI_B should merge to yield ROI_C at layer L_i. Naturally, the operation can be implemented by applying synchronization-based clustering on the layer, as shown in Fig. 7(b). After applying *Merge* operation on all layers, the two ROI networks would merge to one ROI network, and this procedure can be repeated to merge multiple ROI networks.

There is an inevitable error introduced in the merging process, that is the regret term in online learning which measures the difference between the loss of online model and the offline model. We will analyze it in the experimental section.

ROI Network Decomposition. Similar to *Merge* operation, a ROI network should be capable of breaking up into several ROI networks, when the application requires a fine-grained temporal range. So we introduce another operation:



(a) *Split* on points at layer L_{i-1}

(b) *Split* on ROIs at layer L_i

Figure 8: ROI decomposition on a hierarchical ROI network.

Split on ROI network. The strategy of splitting a ROI network is relatively simple: For layer L_i, some points (light blue points in 8(a)) need to be removed from data on existing ROI network. Just remove them directly and update the indexing of network structure so that the generated ROI will no longer contain those points (8(b)). At the first sight, the removal of points should result in the adjustment of the positions of ROIs, but we choose to keep the original ROI fixed. This is because any adjustment of positions would yield new representative error, besides, the amount of removal points is relatively small comparing to original data points. And experiments shows the operation *Split* can keep the representative error stable.

5.2. Trajectory Stream Retrieval

In order to conduct the complex analysis on trajectory data, the basic requirement for database is to support trajectory retrieval effectively. According to Deng et al. [47], there are three kinds of relationship in trajectory queries: (1) Trajectories and points. For example, find all trajectories within 500m of a gas station between 9:00pm-9:30pm. (2) Trajectories and regions, e.g., find the region which is passed by certain trajectories between 9:00pm-9:30pm. (3) Trajectories and trajectories, e.g., travelers who may take a similar path in the coming 30mins.

Now we demonstrate our model naturally supports the first two types of query mentioned above, i.e., given at least one point or region, the trajectories near to the point(s)/region(s) in specific time could be easily retrieved. Formally, we define the spatiotemporal query on trajectory stream as follows.

Definition 5.1 (Spatiotemporal Query)**.** For each time, a spatiotemporal query $\mathcal{Q} = \{(P_1, \triangle t_1), (P_2, \triangle t_2), \cdots, (P_n, \triangle t_n)\}$, which comprises a set of distinct points and their time ranges, is to ask for the set of trajectories $\mathcal{T}_q$ that pass or surround all the points in query $\mathcal{Q}$ during the required periods. Where $P_i = (x_i, y_i)$ is a two-dimensional data point or the center of a given region, and $\triangle t_i = (t_i^s, t_i^e)$ denotes the starting time t_i^s and ending time t_i^e for trajectory passing nearby the point P_i.

Assume a set of ROI networks $\mathcal{G} = \{G_1, G_2, \cdots, G_T\}$ have been generated for all continuously time windows. The spatiotemporal query can be readily done by applying the following three steps:

Step 1: For each point $P_i \in \mathcal{Q}$, identify the ROI network set $\mathcal{G}_i$ that contains the time range $\triangle t_i = (t_i^s, t_i^e)$ for the query.

Step 2: For each network $G \in \mathcal{G}_i$ generated in Step 1, extract all trajectories that pass through the point P_i, which is denoted as $\mathcal{T}_k^i$. Therefore, take the union operation to get all trajectories on all networks as $\mathcal{T}^i = \texttt{Union}(\mathcal{T}_1^i, \cdots, \mathcal{T}_K^i)$.

Step 3: To obtain the final results under all query conditions in $\mathcal{Q}$, perform the interaction operation on the retrieval results from Step 2, i.e., $\mathcal{T}_q = \texttt{Intersect}(\mathcal{T}^1, \mathcal{T}^2, \cdots, \mathcal{T}^n)$.

The time complexity for the query is $\mathcal{O}(nT)$, where n is the number of points/regions in one query, and T is the number of trajectory. Using Hash technique, the process can be accelerated further. The pseudocode is shown in Algorithm 2.

Algorithm 2: Spatiotemporal Query in Trajectory Stream

Input : Derived ROI networks in all time periods $\mathcal{G} = \{G_1, G_2, \cdots, G_T\}$;
Query $\mathcal{Q} = \{(P_1, \triangle t_1), (P_2, \triangle t_2), \cdots, (P_n, \triangle t_n)\}$;

Output : Trajectory set $\mathcal{T}_q$;

foreach *point* $P_i \in \mathcal{Q}$ **do**
  Identify the ROI network set $\mathcal{G}_i$;
  foreach *ROI network* $G_k \in \mathcal{G}_i$ **do**
    Get the trajectory set $\mathcal{T}_k^i$;
  end
  $\mathcal{T}^i = \texttt{Union}(\mathcal{T}_1^i, \cdots, \mathcal{T}_K^i)$;
end
$\mathcal{T}_q = \texttt{Intersect}(\mathcal{T}^1, \mathcal{T}^2, \cdots, \mathcal{T}^n)$;

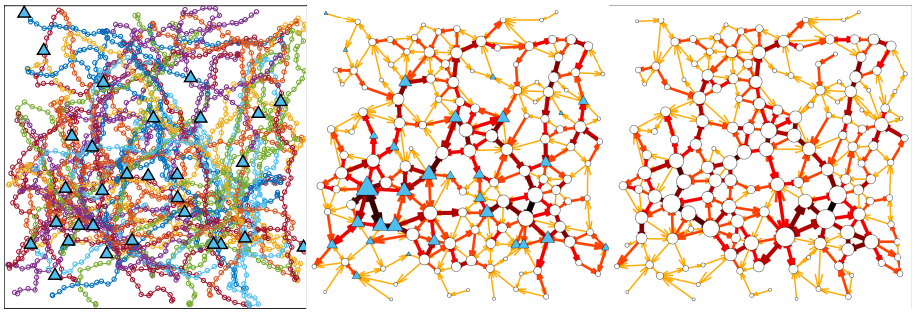
6. Experiment

In this section we perform experiments to evaluate the performance of our algorithm CascadeSync. To reveal the robust property of CascadeSync in different semantic conditions, we design experiments on data sets with and without predefined semantic ROIs, respectively. We also conduct experiments to prove the desirable properties of CascadeSync and also compare the performance of our algorithm with some baseline algorithms. Beyond, the experiments of spatiotemporal query are also given to show the effectiveness of our model in the trajectory retrieval task.

6.1. Experimental Setup

6.1.1. Synthetic Data

To prove the concepts, we generate random trajectories in a rectangle of 100×100 meters, given the random starting point and ending point, using a probabilistic path planning algorithm [48], which is implemented in Matlab 2016b as `robotics.PRM` class. To define the semantics in the trajectories, we randomly select 30 points as semantic ROIs. Fig. 9(a) shows the generated data



(a) Synthetic data (b) With Semantics (c) Without Semantics

Figure 9: Trajectory compression on 100 synthetic trajectories. (a) The resulting one-layer ROI network ($\epsilon = 5m$) with 30 semantic points. (a) The resulting one-layer ROI network ($\epsilon = 5m$) without 30 semantic points. Here the size of ROI is proportional to the point weight. The width and color shade of edges are proportional to the number of trajectories passing through the two ending ROIs.

set with 100 random trajectories, where the light blue triangles are selected as semantic ROIs.

6.1.2. Real-world Data

We evaluate our proposed method on four trajectory data sets: Geolife[1], T-drive[2], Atlantic Hurricanes[3] and Migration of Argentine Barn Swallows[4]. Geolife is a trajectory dataset collected from human daily life by Zheng et al. [49, 50, 51], which reveals human mobility. T-drive contains trajectories generated by urban taxi, which is collected by Yuan et al. [52, 53]. Both Geolife and T-Drive are trajectory data set in Beijing, the substantial distinction consists in the sampling rate. About 91 percent of trajectories from Geolife are logged in a dense representation, e.g., every 1-5 seconds or every 5-10 meters per point, while trajectories from T-Drive are sampled in a very low frequency, like 2-5 minutes per point. Atlantic Hurricanes, collected by NHC (National Hurricane Center), records the track of Atlantic hurricane in 1851-2016. And Argentine

[1] https://www.microsoft.com/en-us/research/publication/geolife-gps-trajectory-dataset-user-guide/

[2] https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/

[3] http://www.nhc.noaa.gov/data/hurdat/

[4] https://www.datarepository.movebank.org/handle/10255/move.655

Table 2: Statistics of four well-known real-world data sets.

Dataset	#Point	#Trajectory	[Longitude, Latitude Range]	[Width × Length] (m)
Geolife	24,876,978	18,670	$[116.194, 116.553, 39.751, 40.033]$	$[31024 \times 31368]$
T-Drive	6,969,481	8,768	$[116.194, 116.553, 39.751, 40.033]$	$[31024 \times 31368]$
Hurricane	49,682	1,830	$[-113.708, 44.713, 5.221, 74.725]$	$[7.728 \times 10.860] \times 10^6$
Migration	4,544	9	$[-69.75, -33.80, -39.61, 16.80]$	$[6.272 \times 3.915] \times 10^6$

Barn Swallows records the tracks of migrating birds in South America. All data sets contain the sampling GPS points described by latitude, longitude and time stamp. We summarize the statistics of the four data sets in Table 2.

6.1.3. Semantic Enrichment

To generate the groundtruth of semantic information, we downloaded the urban roadmap of Beijing from OpenStreetMap[5]. The roadmap is expressed by nodes and edges. We randomly select some semantic ROIs as the intersections, e.g., the nodes with degree greater than two. Besides, to avoid the intersections to dominate all resulting ROIs, we randomly sample 1000 intersections as fixed semantic ROIs, and apply them in CascadeSync on Geolife and T-Drive. Here we do not introduce semantic ROI on Hurricane and Migration data, since it is meaningless to define the semantic point or region on the overland area.

6.1.4. Evaluation Metrics

Generally, traditional trajectory compression algorithms are evaluated by the compression ratio. It is not a suitable metric for semantic trajectory compression algorithms, since there is a trade-off between compression ratio and semantic information preservation. Considering our proposed model CascadeSync can compress trajectory data with and without semantic information, we compare our model to two traditional compression algorithms with respect to compression

[5] `https://www.openstreetmap.org`

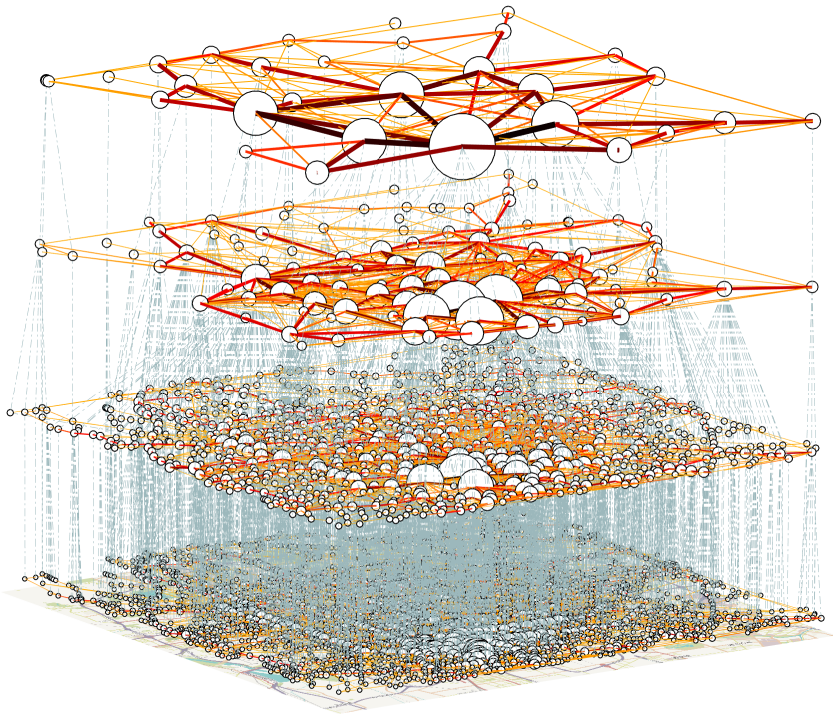


Figure 10: Hierarchical ROI network generated from raw trajectories on Geolife data. Where the epsilon ϵ is set as $(200m, 500m, 1000m, 3000m)$ from bottom layer to top layer.

ratio. The compression ratio is defined as follows.

$$\text{Compression ratio} = 1 - \sum_{i} \frac{|T_i'|}{|T_i|} \times 100\% \tag{7}$$

where each $T_i \in \mathcal{T}'$ is one trajectory of raw trajectory set $\mathcal{T}'$, and T_i' is its compressed representation. $|T_i|$ denotes the number of points or regions on trajectory T_i. Besides, we also compare the runtime of all algorithms on four real-world data sets.

6.1.5. Comparison Models

Performances of semantic trajectory compression methods are not convenient to compare directly, because they are proposed to enrich semantic information to the new representations of trajectory data. To demonstrate the effectiveness and efficiency of CascadeSync, we use five well-known trajectory compression

algorithms as baselines.

Douglas-Peucker [1]: is the most notable line simplification algorithm, which uses a greedy strategy to examine all points between the first and last point until the maximum spatial deviation is within the error bound ζ. The worst-case runtime is $\mathcal{O}(n^2)$.

Douglas-Peucker-SED [15, 16]: is a transformation of original Douglas-Peucker algorithm. It takes a full consideration of spatiotemporal characteristics by replacing perpendicular distance with Synchronized Euclidean Distance (SED), which measures the distance between two points at identical time stamps. The worst-case runtime complexity is also $\mathcal{O}(n^2)$.

Dead Reckoning [2, 14]: is an online compression model, which estimates every successor position through the current position and velocity. By computing the deviation between the estimated position and the true position, the point can be left off from the compressed trajectory if the deviation is less than the error bound. The time complexity of Dead Reckoning is $\mathcal{O}(n)$.

Squish [14, 3]: uses a priority queue where the priority of each point is defined as an estimate of the error that the removal of that point would introduce. SQUISH compresses each trajectory by removing points of the lowest priority from the priority queue until it achieves the target compression ratio or the error bound. The time complexity of Squish is $\mathcal{O}(\log n)$.

Traclus-MDL [13]: is a compression technique that can only achieve constant compression ratio. It uses minimum description length (MDL) [54] to jointly model the complexity and representative capacity of compressed trajectories, so as to derive the optimal characteristic points as compression results. The time complexity of this model is $\mathcal{O}(n)$.

With these baseline algorithms, we can compare the compression ratio, given the error bound. Also, the compression time of all models can be listed for efficiency comparison. Furthermore, the effect of semantic ROIs in CascadeSync is shown.

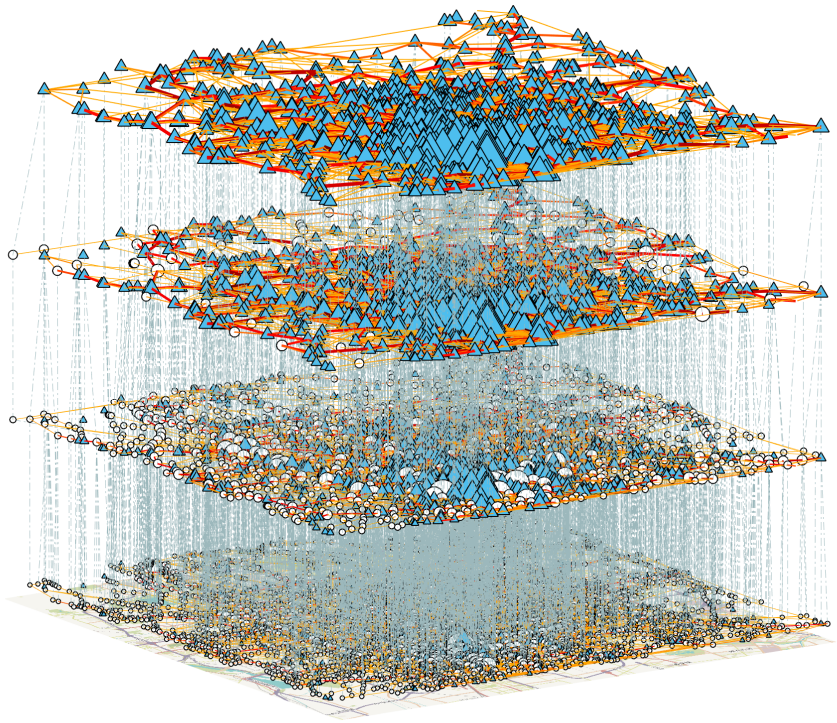
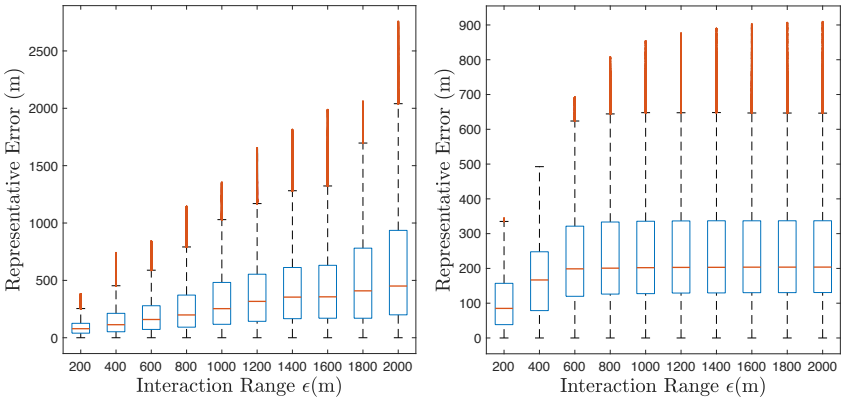


Figure 11: Hierarchical ROI network generated from raw trajectories with additional 1000 road intersections (which are represented by blue triangles) on Geolife data. Where the epsilon ϵ is set as $(200m, 500m, 1000m, 3000m)$ from bottom layer to top layer.

6.2. Proof of Concepts

6.2.1. Visualization of ROI network

For illustration, we first visualize the ROI networks generated on different data sets. For the sake of conciseness, we only visualize the multi-layer ROI network of Geolife data in Fig. 10 and Fig. 11, which are the results with and without using 1000 fixed intersections in CascadeSync, respectively. Besides, one layer on hierarchical ROI network of all data sets are illustrated in Fig. 9 and Fig. 15. From those figures, we can see our model is effective for visualization, which may provide great prospects for many real-life scenarios.



(a) Without semantic ROIs (b) With semantic ROIs

Figure 12: Relationship between representative error and interaction range. The horizontal red line in box is the median value, the upper boundary of blue box is the third quartile points, the upper end of black dash line indicates the maximal value, and the red line beyond the maximal value are considered as outlier points.

6.2.2. Representative Error Analysis

Unlike traditional methods that compress each trajectory independently. Our synchronization-based clustering model, CascadeSync compresses trajectory globally, and the representative error can be bound by ζ with high probability. Here we start with the experiment to explore how well the trajectories can be represented by ROIs, as well as the relationship between error bound ζ and interaction range ϵ.

Without loss of generality, we conduct this experiment on Geolife data set. By applying CascadeSync model, 24,876,978 GPS points are represented as ROIs from fine-grained to coarse-grained resolution. For illustration, the interaction range ϵ is selected ten times evenly from 200m to 2000m, which is around 6/1000 to 60/1000 of the length of the map. The average representative error of all points can be calculated at different levels by increasing interaction range of each layer in CascadeSync. The results are illustrated as box plot in Fig. 12. Where the horizontal red line in box is the median value, the upper boundary of blue box is the third quartile points, the upper end of black dash line indicates

the maximal value, and the red line beyond the maximal value are considered as outlier points.

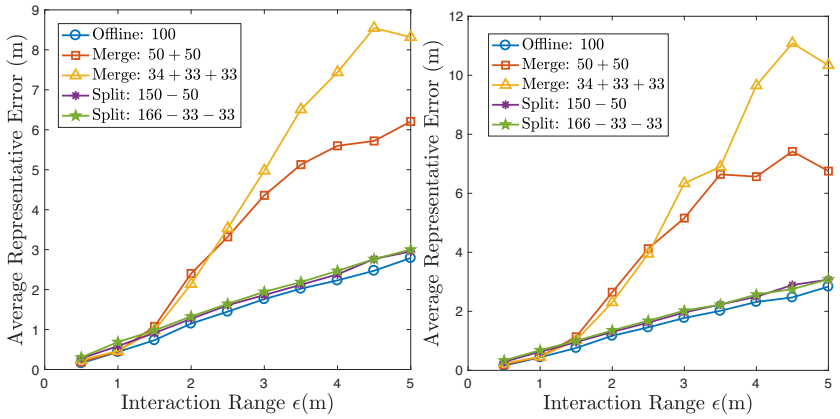
CascadeSync without Semantic ROIs. In the experiment without predefined semantic ROIs, where the result is plotted in Fig. 12(a), we can observe that the maximum is proportional to interaction range ϵ, and is approximately equal to ϵ. Actually, there are 97.64% points whose representative errors are smaller than ϵ. In other word, if we set the error bound $\zeta = \epsilon$, the result of CascadeSync without semantic ROIs would be bounded by ζ with probability greater than 0.9764.

CascadeSync with Semantic ROIs. By introducing intersections as fixed semantic ROIs, the result is plotted in Fig. 12(b). Different from the previous one, the representative error is increasing with interaction range ϵ until ϵ reachs 800 meters, then it remains as constant with increase of ϵ. In this experiment, if we set error bound $\zeta = \epsilon$, the result would be bounded by ζ, with probability greater than 0.9971.

Until now, we have shown that our method CascadeSync is bounded by error bound $\zeta = \epsilon$, with very high probabilities. Moreover, by introducing predefined semantic ROIs, the representative error could be further reduced. The reason is quite intuitive: more semantic information will lead to more dense representative ROIs. However, the compression ratio will also increase, we will show this in the next experiment.

6.2.3. ROI Network Online Merge/Split Analysis

As the amount of trajectory grows, it is impractical to represent all trajectory in a single ROI network. We have developed a window-based method to breaking the trajectory data depending on data recency. However, if there is a request to present or visualize data with a given specific period, which the existing window size fails to satisfy, the operation *Merge/Split* would be applied. The online operations on ROI networks reduce the runtime which is needed by constructing a ROI network from scratch, although the additional error will be introduced.



(a) With Semantic ROIs (b) Without Semantic ROIs

Figure 13: Average representative error generated by offline/online compression.

Now we conduct experiments to evaluate the error introduced by the online *Merge/Split* operation(s).

Using the synthetic data generation method, 200 random trajectories and 30 fixed semantic ROI are generated. 100 trajectories from the 200 trajectories are randomly selected as bases. We design five schemes to generate a ROI network by applying CascadeSync model : 1) ROI network is generated by the 100 bases in one attempt (the offline version); 2) merge two ROI networks which are generated by first half and second half of the bases individually (i.e., 100 trajectories using (50 + 50) merge operation); 3) merge three ROI networks that are generated from three parts of bases (i.e., 100 trajectories using (34 + 33 + 33) merge operation); 4) split a ROI network that is generated from 50 non-base trajectories from a ROI network and make the rest ROI network contains exactly the 100 base trajectories (i.e., 100 trajectories using (150 - 50) split operation); 5) split two ROI networks and make the result ROI network contains exactly the 100 base trajectories (i.e., 100 trajectories using (166 - 33 - 33) split operations).

Fig. 13 illustrate the representative errors with the five schemes. From the figure, we can observe that the operation *Split* always remains the representative error as the same level as the offline version, which means *Split* would not

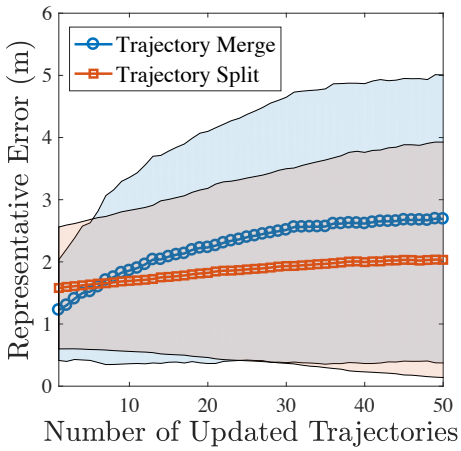
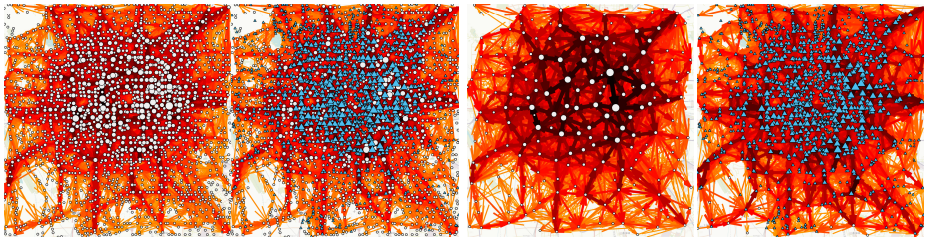


Figure 14: The relationship between representative error and number of merged/split trajectories on one layer of ROI network with interaction range $\epsilon = 2.5$m. The mean values and area bounded by one standard deviation are shown.

introduce additional error and thus reliable for various applications. However, the operation *Merge* will introduce additional error as the interaction range ϵ exceeds 1.5m (15/1000 of the length of map), and it grows with the increase of ϵ. Another observation is the more times the *Merge* conducted, the larger the additional error introduced since more trajectory is summarized in a higer level. Therefore, the *Merge* operation is only suitable for the applications which are not sensitive to the representative error such as visualization.

Furthermore, to get a fine-grained view of how *Merge* and *Split* affect the representative error. We merge the ROI network with 50 trajectories by adding one single trajectory per time, until all 100 base trajectories are added in an existing ROI network. Similarly, we split the ROI network with 150 trajectories by removing one single trajectory per time, till the existing ROI network contains exactly the 100 base trajectories. The mean value and area bounded by one standard deviation of error curve on layer with interaction range $\epsilon = 2.5$m are drawn in Fig. 14. From the result, we can see that the *Split* operation can keep the representative error on a low level, and *Merge* operation also will not introduce the large error as in Fig. 13. This is because the trajectories are

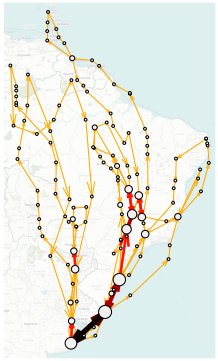
(a) Geolife: $\epsilon = 200$ (m) (b) Geolife: $\epsilon = 1000$ (m)



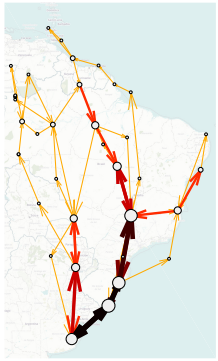
(c) T-drive: $\epsilon = 200$ (m) (d) T-drive: $\epsilon = 1000$ (m)

(e) $\epsilon = 180$ (km)

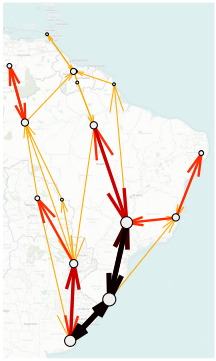
(f) $\epsilon = 540$ (km)



(g) $\epsilon = 10$ (km)



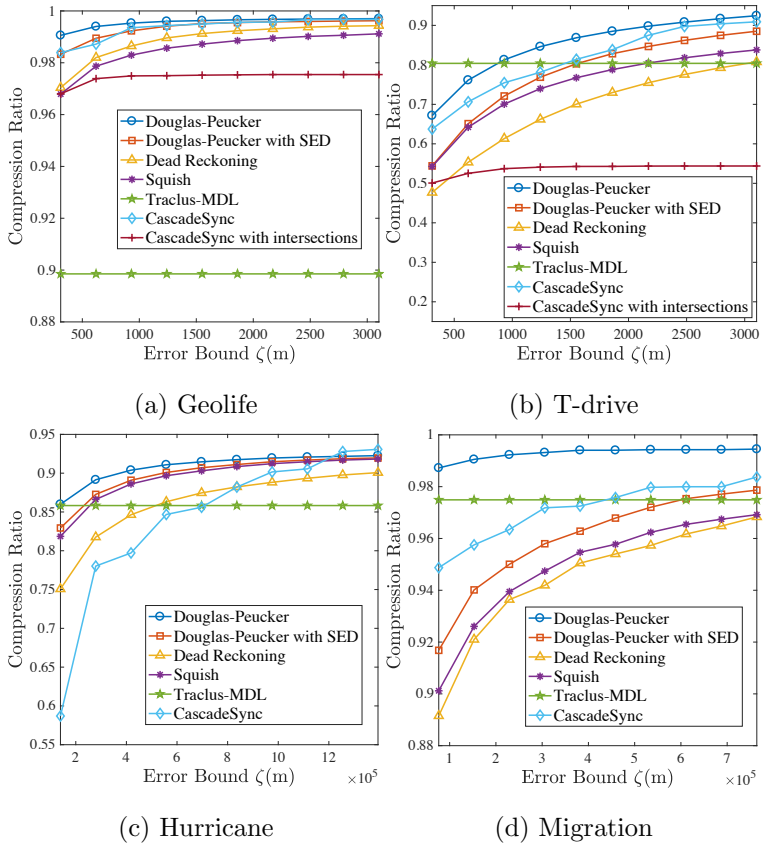
(h) $\epsilon = 30$ (km)



(i) $\epsilon = 50$ (km)

Figure 15: The ROI networks on different data sets with different error bounds. In each subfigure, the left-hand side ROI network is generated without semantic ROIs, and the right-hand side ROI network is generated with 1000 random selected intersections. The size of ROI is proportional to the node weight (i.e., the number of raw GPS coordinates each ROI represented). The edge width and color shade are proportional to the transportation flow of two ending ROIs.

gradually added into existing ROI network, so it can maintain the correct representation. This suggests the model can be extended to compress and represent trajectory data on the fly.



(a) Geolife (b) T-drive

(c) Hurricane (d) Migration

Figure 16: Compression ratio with varying representative error bound.

6.3. Evaluation of Compression

In this section, we compare the compression ratio with baseline algorithms to show the excellent representative capability of CascadeSync. In order to compare the performance directly, different algorithms should be bounded with the same error bound ζ. We set $\zeta = \epsilon$ in CascadeSync, since we have illustrated that the results are convincing with high probabilities (greater than 97%).

As mentioned above, CascadeSync is employed twice on Geolife and T-Drive data, where the difference is whether the predefined intersections (i.e., semantic information) is used or not. By varying the error bound ζ in a reasonable range, we calculate the compression ratios of four data sets and plot the

results in Fig. 16. Here, part of results are visualized in Fig. 15.

Note the scales of compression ratio are different on these four data sets. For example, Geolife and T-drive both are trajectories in Beijing, however, the sampling rate of Geolife is hundreds of times larger than that of T-drive. So there is much more redundancy contained in Geolife, which results in a very high compression ratio. The ROI networks in Fig. 15(a)-(d) also show the differences. The ROI network of T-drive is more complicated than Geolife, which indicates there are many gaps on trajectories.

Nonetheless, the compression ratios of four models are comparable in each data set. Douglas-Peucker is superior to the other models, and our CascadeSync in free space inclines to exceed other models with the increase of error bound. Note that Traclu-MDL presents a flat line, since the model does not take the error bound as a parameter, thus cannot control the compression ratio. Meanwhile, CascadeSync with semantic information are not good at compression ratio, which is due to the semantic ROIs that not allowed to move or reduce on map. This phenomenon can also be revealed from ROI networks in Fig. 15(a-d): by increasing the error bound ζ, there are few and few normal ROIs exist, the road intersections gradually become the dominant ROIs on the map.

The experiment has shown the excellent compression ratio and representative capability of CascadeSync. Now we show the most salient feature of CascadeSync by comparing the runtime with other algorithms. The runtimes of six algorithms on four real-world data are reported. Each algorithm runs ten times on each data set, the average runtime is calculated as listed in Table 3. Note that the runtime in Table 3 and the compression ratio in Fig. 16 are measured simultaneously in the same experimental setting. CascadeSync is thousands times faster than other five methods. This is because the traditional models compress trajectory one by one, thus the runtime is proportional to the points in data set. Our model works globally, so the number of points would reduce exponentially so that the compression process would be accelerated.

Table 3: Runtime records on the real-world data sets. The result is measured in second (s).

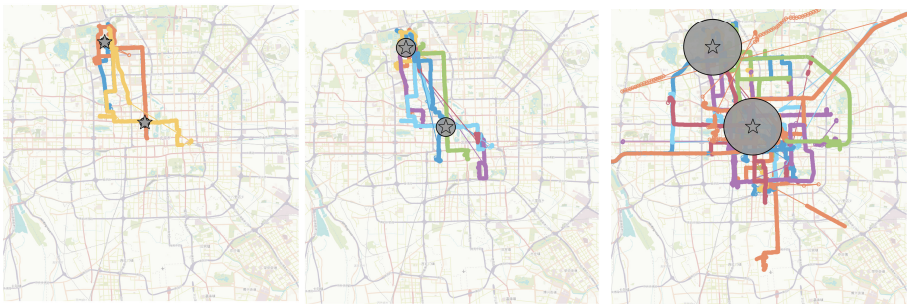
	DP	**DP-SED**	**DR**	**Squish**	**Traclus-MDL**	**CascadeSync**
Geolife	6100.42	36621.08	14525.53	14712.10	6357.91	**15.38**
Tdrive	2288.57	35422.91	5101.41	5289.05	2344.51	**7.51**
Hurricane	16.46	49.80	30.98	33.54	13.78	**0.42**
Migration	1.79	12.09	4.40	4.44	2.28	**0.10**

Table 4: Spatiotemporal queries on Geolife data.

Spatiotemporal queries			**Number of trajectories**		
No.	Location	Time range	$\epsilon = 500m$	$\epsilon = 1000m$	$\epsilon = 3000m$
1	Peking University, Jinrong Street.	2008-11-28,2009-01-07	3	8	51
2	Wangfujing, Beijing West Railway Station.	2009-02-14,2009-03-20	4	18	44
3	Tiananmen Square, Beijing Railway Station, National Stadium.	2009-01-27,2009-02-25	0	2	9
4	Zhongguancun, Guangximen, Jianguomen.	2008-06-18,2008-06-22	1	2	12
5	Tiantan Park, Forbidden City, Beihai Park.	2008-04-02,2009-11-23	15	139	578

6.4. Spatiotemporal Trajectory Query

The introduced ROI network is a compact representation of trajectory data. The global information and special semantic information can be summarized, which can facilitate the downstream applications. In this section, we take the spatiotemporal retrieval task as an example. Given a set of landmarks or regions in the city as well as the periods, retrieve the trajectories passed all those regions during corresponding periods. Since our multi-resolution ROI networks has recorded the passing trajectories with visiting time, trajectories can be easily



(a) $\epsilon = 500m$, #Traj.=3 (b) $\epsilon = 1000m$, #Traj.=8 (c) $\epsilon = 3000m$, #Traj.=51

Figure 17: Retrieval result of spatiotemporal trajectory query with different searching ranges. Two locations are selected, which are Peking University and Jinrong Street. The time range from 2008-11-28 to 2009-01-07. The radius of circle indicates the range ϵ.

retrieved by applying Algorithm 2 on the existing ROI networks.

For simplicity, we only illustrate the results of queries on Geolife. Five exemplified queries are performed. Since most trajectories are located around Microsoft Research Asia (MSRA) and campuses of some universities, the data is sparse with respect to the whole Beijing city. For illustration, here we only consider three ranges (500m, 1000m and 3000m), and give the first retrieval result from 2008-11-28 to 2009-01-07 in Fig. 17. There are only 3 trajectories pass the campus of Peking University and Jinrong Street, when the ϵ-range is set as $500m$. The number of results grows, naturally, over the increasing ϵ-range.

The experiment shows that by given specific points and corresponding time ranges, the query results are listed and visualized with different resolution settings, which could facilitate a wide range of applications.

7. Conclusion

In this paper, we propose a synchronization-based semantic trajectory compression algorithm CascadeSync, by representing a given trajectory data set to a multi-resolution ROI network. Building upon the concept of synchronization, CascadeSync allows yielding multi-resolution trajectory abstractions (hierarchical ROI network) with and without available semantic information. More

importantly, the abstracted trajectory representation well preserves the global trajectory information, and the semantic can be well integrated. For trajectory stream, we develop a simple window-based method to represent trajectories on multiple ROI networks according to data recency. The experiments we performed on synthetic data and four real-world data sets have demonstrated its effectiveness and efficiency, and show its superiority to other five baseline methods.

References

[1] D. H. Douglas, T. K. Peucker, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, Cartographica: The International Journal for Geographic Information and Geovisualization 10 (2) (1973) 112–122.

[2] G. Trajcevski, H. Cao, P. Scheuermanny, O. Wolfsonz, D. Vaccaro, On-line data reduction and the quality of history in moving objects databases, in: Proceedings of the 5th ACM International Workshop on Data Engineering for Wireless and Mobile Sccess, ACM, 2006, pp. 19–26.

[3] J. Muckell, P. W. Olsen, J.-H. Hwang, C. T. Lawson, S. Ravi, Compression of trajectory data: a comprehensive evaluation and new approach, GeoInformatica 18 (3) (2014) 435–460.

[4] J. Liu, K. Zhao, P. Sommer, S. Shang, B. Kusy, R. Jurdak, Bounded quadrant system: Error-bounded trajectory compression on the go, in: Data Engineering (ICDE), 2015 IEEE 31st International Conference on, IEEE, 2015, pp. 987–998.

[5] X. Lin, S. Ma, H. Zhang, T. Wo, J. Huai, One-pass error bounded trajectory simplification, Proceedings of the VLDB Endowment 10 (7) (2017) 841–852.

[6] F. Schmid, K.-F. Richter, P. Laube, Semantic trajectory compression, Advances in Spatial and Temporal Databases (2009) 411–416.

[7] K. Liu, Y. Li, J. Dai, S. Shang, K. Zheng, Compressing large scale urban trajectory data, in: Proceedings of the Fourth International Workshop on Cloud Data and Platforms, ACM, 2014, p. 3.

[8] Y. Chen, K. Jiang, Y. Zheng, C. Li, N. Yu, Trajectory simplification method for location-based social networking services, in: Proceedings of the 2009 International Workshop on Location Based Social Networks, ACM, 2009, pp. 33–40.

[9] R. Song, W. Sun, B. Zheng, Y. Zheng, Press: A novel framework of trajectory compression in road networks, Proceedings of the VLDB Endowment 7 (9) (2014) 661–672.

[10] R. Gotsman, Y. Kanza, Compact representation of gps trajectories over vectorial road networks, in: International Symposium on Spatial and Temporal Databases, Springer, 2013, pp. 241–258.

[11] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, W.-Y. Ma, Mining user similarity based on location history, in: Proceedings of the 16th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, 2008, p. 34.

[12] P. Sun, S. Xia, G. Yuan, D. Li, An overview of moving object trajectory compression algorithms, Mathematical Problems in Engineering 2016.

[13] J.-G. Lee, J. Han, K.-Y. Whang, Trajectory clustering: a partition-and-group framework, in: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, 2007, pp. 593–604.

[14] J. Muckell, J.-H. Hwang, V. Patil, C. T. Lawson, F. Ping, S. Ravi, Squish: an online approach for gps trajectory compression, in: Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications, ACM, 2011, p. 13.

[15] N. Meratnia, A. Rolf, Spatiotemporal compression techniques for moving point objects, in: International Conference on Extending Database Technology, Springer, 2004, pp. 765–782.

[16] M. Potamias, K. Patroumpas, T. Sellis, Sampling trajectory streams with spatiotemporal criteria, in: 18th International Conference on Scientific and Statistical Database Management., IEEE, 2006, pp. 275–284.

[17] C. Long, R. C.-W. Wong, H. Jagadish, Direction-preserving trajectory simplification, Proceedings of the VLDB Endowment 6 (10) (2013) 949–960.

[18] C. Long, R. C.-W. Wong, H. Jagadish, Trajectory simplification: on minimizing the direction-based error, Proceedings of the VLDB Endowment 8 (1) (2014) 49–60.

[19] A. Civilis, C. S. Jensen, S. Pakalnis, Techniques for efficient road-network-based tracking of moving objects, IEEE Transactions on Knowledge and Data Engineering 17 (5) (2005) 698–712.

[20] R. Gotsman, Y. Kanza, A dilution-matching-encoding compaction of trajectories over road networks, GeoInformatica 19 (2) (2015) 331–364.

[21] I. S. Popa, K. Zeitouni, V. Oria, A. Kharrat, Spatio-temporal compression of trajectories in road networks, GeoInformatica 19 (1) (2015) 117–145.

[22] Y. Dong, D. Pi, Novel privacy-preserving algorithm based on frequent path for trajectory data publishing, Knowledge-Based Systems.

[23] Y. Zheng, Trajectory data mining: an overview, ACM Transactions on Intelligent Systems and Technology 6 (3) (2015) 29.

[24] J. D. Mazimpaka, S. Timpf, Trajectory data mining: A review of methods and applications, Journal of Spatial Information Science 2016 (13) (2016) 61–99.

[25] J. Shao, X. He, C. Böhm, Q. Yang, C. Plant, Synchronization-inspired partitioning and hierarchical clustering, IEEE Transactions on Knowledge and Data Engineering 25 (4) (2013) 893–905.

[26] J. A. Acebrón, L. L. Bonilla, C. J. P. Vicente, F. Ritort, R. Spigler, The kuramoto model: A simple paradigm for synchronization phenomena, Reviews of modern physics 77 (1) (2005) 137.

[27] P. Seliger, S. C. Young, L. S. Tsimring, Plasticity and learning in a network of coupled phase oscillators, Physical Review E 65 (4) (2002) 041906.

[28] A. Arenas, A. Diaz-Guilera, C. J. Pérez-Vicente, Synchronization reveals topological scales in complex networks, Physical review letters 96 (11) (2006) 114102.

[29] J. Shao, Synchronization on data mining: a universal concept for knowledge discovery, LAP LAMBERT Academic Publishing, Saarbrücken.

[30] J. Shao, Z. Han, Q. Yang, T. Zhou, Community detection based on distance dynamics, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp. 1075–1084.

[31] W. Ying, F.-L. Chung, S. Wang, Scaling up synchronization-inspired partitioning clustering, IEEE Transactions on Knowledge and Data Engineering 26 (8) (2014) 2045–2057.

[32] J. Shao, Q. Yang, H.-V. Dang, B. Schmidt, S. Kramer, Scalable clustering by iterative partitioning and point attractor representation, ACM Transactions on Knowledge Discovery from Data 11 (1) (2016) 5.

[33] J. Shao, F. Huang, Q. Yang, G. Luo, Robust prototype-based learning on data streams, IEEE Transactions on Knowledge and Data Engineering.

[34] J. Shao, C. Gao, W. Zeng, J. Song, Q. Yang, Synchronization-inspired co-clustering and its application to gene expression data, in: 2017 IEEE 11th International Conference on Data Mining (ICDM), IEEE, 2017.

[35] J. Shao, X. Wang, Q. Yang, C. Plant, C. Böhm, Synchronization-based scalable subspace clustering of high-dimensional data, Knowledge and Information Systems 52 (1) (2017) 83–111.

[36] Y. Kuramoto, Self-entrainment of a population of coupled non-linear oscillators, in: International symposium on mathematical problems in theoretical physics, 1975, pp. 420–422.

[37] Y. Kuramoto, Chemical oscillations, waves, and turbulence, Vol. 19, Springer Science & Business Media, 2012.

[38] C. S. Kim, C. S. Bae, H. J. Tcha, A phase synchronization clustering algorithm for identifying interesting groups of genes from cell cycle expression data, BMC bioinformatics 9 (1) (2008) 56.

[39] J. Shao, C. Böhm, Q. Yang, C. Plant, Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part III, Springer Berlin Heidelberg, 2010, Ch. Synchronization Based Outlier Detection.

[40] L. Liao, D. Fox, H. Kautz, Extracting places and activities from gps traces using hierarchical conditional random fields, The International Journal of Robotics Research 26 (1) (2007) 119–134.

[41] L. Wang, K. Hu, T. Ku, X. Yan, Mining frequent trajectory pattern based on vague space partition, Knowledge-based systems 50 (2013) 100–111.

[42] C.-Y. Tsai, B.-H. Lai, A location-item-time sequential pattern mining algorithm for route recommendation, Knowledge-Based Systems 73 (2015) 97–110.

[43] Y. Zheng, L. Zhang, Z. Ma, X. Xie, W.-Y. Ma, Recommending friends and locations based on individual location history, ACM Transactions on the Web 5 (1) (2011) 5.

[44] Y. Si, F. Zhang, W. Liu, Ctf-ara: An adaptive method for poi recommendation based on check-in and temporal features, Knowledge-Based Systems 128 (2017) 59–70.

[45] M. Lv, L. Chen, Y. Shen, G. Chen, Measuring cell-id trajectory similarity for mobile phone route classification, Knowledge-Based Systems 89 (2015) 181–191.

[46] C. Böhm, C. Plant, J. Shao, Q. Yang, Clustering by synchronization, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010, pp. 583–592.

[47] K. Deng, K. Xie, K. Zheng, X. Zhou, Trajectory indexing and retrieval, in: Computing with spatial trajectories, Springer, 2011, pp. 35–60.

[48] L. E. Kavraki, P. Svestka, J.-C. Latombe, M. H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, in: IEEE Transactions on Robotics and Automation, 1996.

[49] Y. Zheng, Q. Li, Y. Chen, X. Xie, W.-Y. Ma, Understanding mobility based on gps data, in: Proceedings of the 10th International Conference on Ubiquitous Computing, 2008, pp. 312–321.

[50] Y. Zheng, L. Zhang, X. Xie, W.-Y. Ma, Mining interesting locations and travel sequences from gps trajectories, in: Proceedings of the 18th International Conference on World Wide Web, 2009, pp. 791–800.

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

[52] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, Y. Huang, T-drive: driving directions based on taxi trajectories, in: Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems, 2010, pp. 99–108.

[53] J. Yuan, Y. Zheng, X. Xie, G. Sun, Driving with knowledge from the physical world, in: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, 2011, pp. 316–324.

[54] P. D. Grünwald, I. J. Myung, M. A. Pitt, Advances in minimum description length: Theory and applications, MIT press, 2005.