Compression of Uncertain Trajectories in Road Networks

Tianyi Li† Ruikai Huang⋆ Lu Chen† Christian S. Jensen† Torben Bach Pedersen†

†Department of Computer Science, Aalborg University, Denmark
⋆College of Computer Science, Zhejiang University, Hangzhou, China
{tianyi, luchen, csj, tbp}@cs.aau.dk ngzuekai@zju.edu.cn

ABSTRACT

Massive volumes of uncertain trajectory data are being generated by GPS devices. Due to the limitations of GPS data, these trajectories are generally uncertain. This state of affairs renders it is attractive to be able to compress uncertain trajectories and to be able to query the trajectories efficiently without the need for (full) decompression. Unlike existing studies that target accurate trajectories, we propose a framework that accommodates uncertain trajectories in road networks. To address the large cardinality of instances of a single uncertain trajectory, we exploit the similarity between uncertain trajectory instances and provide a referential representation. First, we propose a reference selection algorithm based on the notion of Fine-grained Jaccard Distance to efficiently select trajectory instances as references. Then we provide referential representations of the different types of information contained in trajectories to achieve high compression ratios. In particular, a new compression scheme for temporal information is presented to take into account variations in sample intervals. Finally, we propose an index and develop filtering techniques to support efficient queries over compressed uncertain trajectories. Extensive experiments with real-life datasets offer insight into the properties of the framework and suggest that it is capable of outperforming the existing state-of-the-art method in terms of both compression ratio and efficiency.

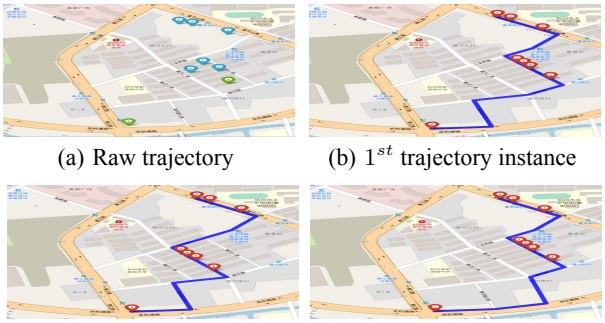
PVLDB Reference Format:
Tianyi Li, Ruikai Huang, Lu Chen, Christian S. Jensen, and Torben Bach Pedersen. Compression of Uncertain Trajectories in Road Networks. *PVLDB*, 13(7): 1050-1063, 2020.
DOI: https://doi.org/10.14778/3384345.3384353

Keywords

Compression, Uncertain Trajectory, Road Network, Query

1. INTRODUCTION

GPS sensors in smart-phones, in-vehicle navigation systems, and wearable devices more generally are generating massive volumes of raw trajectories, increasing the cost of data storage and transmission [20, 21, 42]. This has led to the proposal of a variety of trajectory compression methods that fall into two general categories: raw trajectory compression and road network-embedded compression. The former category of methods aim to compress trajectories without considering an underlying road network [1, 3, 4, 6, 8, 9, 11, 13, 16, 22, 23, 24, 25, 27, 29, 30, 40, 41, 44]. Methods in the latter category first project a trajectory onto a road network using a map-matching algorithm and then exploit the road network to reduce the storage by transforming the mapped trajectories into compact formats [5, 13, 17, 18, 30, 31]. As map-matching can help improve the quality of trajectories and as in-network trajectories are useful, we study road network-embedded compression.



(a) Raw trajectory (b) 1^{st} trajectory instance
(c) 2^{nd} trajectory instance (d) 3^{rd} trajectory instance

Figure 1: A real-life example of a raw trajectory and a corresponding network-constrained uncertain trajectory

The uncertainty of trajectories is caused by two characteristics, a low sampling rate and inaccurate GPS positions [38, 45]. A low sampling rate can make multiple routes between two map-matched GPS positions possible, while position inaccuracy means that a raw GPS position may be map-matched to multiple road-network positions. Here, a real-life example is shown in Fig. 1a for a trajectory consisting of 8 raw GPS points recorded by a taxi in Hangzhou, China. All these GPS points are off the road, and the sampling interval between two green points exceeds 4 minutes. To address this, probabilistic map-matching [2, 15] has been proposed to transform a raw trajectory into a set of network-constrained trajectory instances. Instead of mapping each position in a raw trajectory to a unique road-network location, probabilistic map-matching generally finds several potential road-network locations. Figs. 1b, 1c, and 1d are trajectory instances generated from Fig. 1a, which are similar to each other. To the best of our knowledge, no existing solutions aim to compress uncertain trajectories, which is the target of our study.

Two challenges must be addressed in order to effectively compress uncertain trajectories. *The first challenge is how to achieve a high compression ratio.* As shown in the example in Fig. 1, the instances of an uncertain trajectory are similar, which is also validated by statistics from three real-life datasets, to be covered in Section 4.2. Hence, we adopt a referential representation for uncer-

Proceedings of the VLDB Endowment, Vol. 13, No. 7
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3384345.3384353

tain trajectories that has proven effective for highly similar genome sequences [10, 34, 35]. *The second challenge is how to support efficient querying of compressed uncertain trajectories.* An existing study [40] provides an index for accurate compressed trajectories that considers neither the uncertainty nor is applicable to referentially represented trajectory instances. Consequently, we propose a novel indexing technique that eliminates both shortcomings and also propose associated query processing algorithms.

We integrate these contributions into a novel framework that compresses uncertain trajectories and supports efficient query processing without the need for (full) decompression. First, we separate the temporal, path, and distance information of uncertain trajectories, thus following the state-of-the-art TED model [40]. Here, we propose a representation of the temporal information to improve the compression ratio when sample intervals vary. Then, we represent trajectory instances referentially using a two-step process, i.e., selecting high-quality reference trajectory instances and transforming other trajectory instances accordingly. As part of this, we propose a Fine-grained Jaccard Distance function to measure the similarity between trajectories, and we propose a greedy algorithm to efficiently select high-quality references. Finally, we develop an index structure and algorithms that exploit effective filtering techniques and partial decompression to support typical probabilistic queries on compressed uncertain trajectories.

In summary, our main contributions are as follows:

- We propose a novel uncertain trajectory compression framework that supports efficient probabilistic query processing. To the best of our knowledge, this is the first proposal for compression of uncertain trajectories in road networks.
- We develop a representation that accommodates a novel encoding scheme for the temporal information of trajectories, and we use referential representation to compress uncertain trajectories in road networks.
- We design an effective indexing structure and filtering techniques to accelerate query processing, including the processing of *probabilistic where*, *when*, and *range queries*.
- We conduct extensive experimental evaluations that offer insight into the framework and demonstrate that it is able to outperform the state-of-the-art solution [40] by a factor of more than two in terms of the compression ratio and by more than one order of magnitude in terms of compression efficiency.

The rest of the paper is organized as follows. We present preliminaries in Section 2 and give an overview of the proposed framework in Section 3. Section 4 details the representation and compression schemes, and Section 5 covers the index structure and the query processing methodology. Section 6 reports the experimental results. Section 7 reviews related work, and Section 8 concludes and offers directions for future work.

2. PRELIMINARIES

We proceed to introduce probabilistic map-matching and the TED model. Table 1 summarizes frequently used notation.

2.1 Probabilistic Map-Matching

A raw trajectory is a series of time-stamped point locations $p_0, ..., p_{n-1}$ of a moving object of the form (x, y, t), where (x, y) is a point location in 2D Euclidean space, with x being a longitude and y being a latitude, and t is a timestamp. $Tp = \langle p_0, ..., p_6 \rangle$ in Fig. 2a is an example of a raw trajectory.

Table 1: Frequently used notation

Notation	Description
Tu	a set of uncertain trajectories
Tu^j	an uncertain trajectory in **Tu**
N^j	the number of instances in Tu^j
n_p^j	the number of pivots selected for Tu^j
Tu_w^j	an instance of the uncertain trajectory Tu^j
$SV(Tu_w^j)$	the start vertex of Tu_w^j
$D(Tu_w^j)$	the relative distance sequence of Tu_w^j
$Tu_w^j.p$	the probability of Tu_w^j
$E(Tu_w^j)$	the edge sequence of Tu_w^j
$T'(Tu_w^j)$	the time flag bit-string of Tu_w^j
$T(Tu^j)$	the time sequence of Tu^j
l_i	the i^{th} mapped location
Ref_i^j	the i^{th} reference in Tu^j
$Ref_i^j.Rrs$	the referential representation set of Ref_i^j
$Nref_{ik}^j$	the k^{th} non-reference in the set $Ref_i^j.Rrs$
$Com_\phi(Nref_{ik}^j, Ref_i^j)$	the referential representation of $Nref_{ik}^j$
$\phi_{ik}^j(Ma_h)$	the h^{th} factor in $Com_\phi(Nref_{ik}^j, Ref_i^j)$
$\hat{seq}$	the binary code of a sequence *seq*
ω_{seq}	the *flag array* of a sequence *seq*
γ_{seq}	the *original array* of a sequence *seq*

Definition 1. *A* road network *is modeled as a directed graph $G = (V, E)$, where V is a set of vertices $v\,(x, y)$ denoting intersections or end points, and E is a set of directed edges $e\,(v_i \rightarrow v_j)$. Here, (x, y) denotes the 2D location of a vertex.*

For simplicity, we use v and $(v_i \rightarrow v_j)$ to denote a vertex and a directed edge of a road network, respectively. **Map-matching** [14, 28] aligns a raw trajectory with the road network that constrains the movement of the corresponding object, and the result is a network-constrained accurate trajectory. This process transforms each original point location into a mapped location.

Definition 2. *A* mapped location *l is a network-constrained location in a road network G, represented as $\langle (v_s \rightarrow v_e), ndist, t \rangle$, where ndist is the network distance between v_s and l on $(v_s \rightarrow v_e)$ and t is a timestamp.*

For example in Fig. 2a, $l_6 = \langle (v_7 \rightarrow v_8), ndist, 5{:}27{:}25 \rangle$ is the mapped location corresponding to p_6; thus, the two have the same timestamp. We also denote a mapped location as $\langle (v_s \rightarrow v_e), ndist \rangle$ when the timestamp t is not considered.

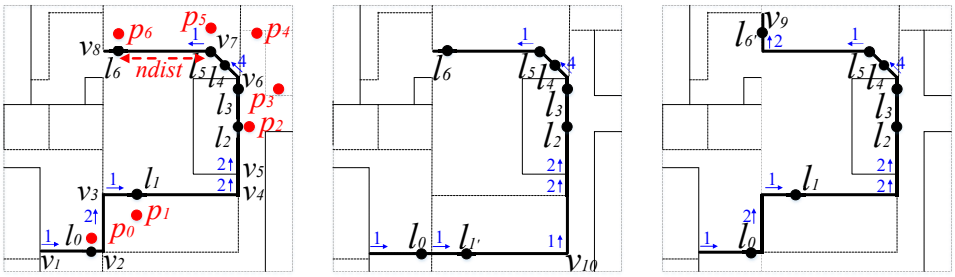
Definition 3. *A* network-constrained accurate trajectory *$Tr = \langle l_0, l_1, ..., l_{n-1} \rangle$ is a time-ordered sequence of mapped locations $l_0, l_1, ..., l_{n-1}$ of a moving object.*

For example in Fig. 2a, $Tr = \langle l_0, ..., l_6 \rangle$ is a network-constrained accurate trajectory.

Definition 4. *Given two vertices v_s and v_e in a road network G, a* path *is a sequence of connected edges $(v_i \rightarrow v_j)$, that starts from v_s and ends at v_e, i.e., $(v_s \rightarrow v_1), (v_1 \rightarrow v_2), \cdots, (v_{n-1} \rightarrow v_e)$.*

For example in Fig. 2a, the path of *Tp* after map-matching is $(v_1 \rightarrow v_2), (v_2 \rightarrow v_3), \cdots, (v_7 \rightarrow v_8)$. A network-constrained **accurate** trajectory encodes a **unique** path of a raw trajectory after map-matching. However, to take into account the uncertainty in raw trajectories (as shown in Fig. 1) when mapping them to the road network, **probabilistic map-matching** [2, 15] is proposed. Unlike a network-constrained accurate trajectory, a network-constrained **uncertain** trajectory encodes a set of **potential** paths, each of which is associated with a likelihood.

p_0=(120.0692, 30.28622, 5:03:25)
p_1=(120.0724, 30.28618, 5:07:25)
p_2=(120.0757, 30.28621, 5:11:26)
p_3=(120.0762, 30.28621, 5:15:26)
p_4=(120.0772, 30.28620, 5:19:25)
p_5=(120.0786, 30.28617, 5:23:25)
p_6=(120.0796, 30.28498, 5:27:25)



(a) Tu_1^1 (b) Tu_2^1 (c) Tu_3^1

Figure 2: Instances of the network-constrained uncertain trajectory Tu^1

Table 2: An example of TED representation

$E(Tr)$	⟨185190 → 1,2,1,2,2,0,4,1,0⟩
$D(Tr)$	⟨0.875,0.25,0.5,0.875,0.5,0,0.875⟩
$T'(Tr)$	⟨1,0,1,0,1,1,1,1,1⟩
$T(Tr)$	⟨(0, 5:03:25), (1, 5:07:25), (2, 5:11:26) (3, 5:15:26), (4, 5:19:25), (6, 5:27:25)⟩

Definition 5. *A network-constrained uncertain trajectory Tu^j contains a set of instances Tu_w^j $(1 \leq w \leq N^j)$ generated from a raw trajectory Tp. Each Tu_w^j is associated with a probability and is represented by a time-ordered sequence of mapped locations that is different from that of Tu_v^j $(1 \leq v \leq N^j \wedge v \neq w)$. All instances of Tu^j share the same temporal information for all mapped locations.*

Fig. 2 shows a network-constrained uncertain trajectory Tu^1 generated from the raw trajectory $Tp = \langle p_0, ..., p_6 \rangle$. Tu^1 contains three instances, i.e., Tu_1^1, Tu_2^1, and Tu_3^1, where $Tu_1^1 = \langle l_0, l_1, l_2, l_3, l_4, l_5, l_6 \rangle$ has probability 0.75, $Tu_2^1 = \langle l_0, l_{1'}, l_2, l_3, l_4, l_5, l_6 \rangle$ has probability 0.2, and $Tu_3^1 = \langle l_0, l_1, l_2, l_3, l_4, l_5, l_{6'} \rangle$ has probability 0.05. They share the temporal information ⟨5:03:25, 5:07:25, 5:11:26, 5:15:26, 5:19:25, 5:23:25, 5:27:25⟩, as they are all generated from *Tp*. In this example, the uncertain trajectory has three possible paths $(v_1 \rightarrow v_2), (v_2 \rightarrow v_3), (v_3 \rightarrow v_4), \cdots, (v_7 \rightarrow v_8)$ for Tu_1^1, $(v_1 \rightarrow v_2), (v_2 \rightarrow v_{10}), (v_{10} \rightarrow v_4), \cdots, (v_7 \rightarrow v_8)$ for Tu_2^1, and $(v_1 \rightarrow v_2), (v_2 \rightarrow v_3), (v_3 \rightarrow v_4), \cdots, (v_8 \rightarrow v_9)$ for Tu_3^1. In contrast, only Tu_1^1 that has the highest probability would be chosen as the network-constrained accurate trajectory *Tr*.

In the rest of the paper, we use accurate trajectory instead of network-constrained accurate trajectory, and we use uncertain trajectory instead of network-constrained uncertain trajectory when this does not cause any ambiguity.

2.2 TED Representation

TED represents an accurate trajectory *Tr* as an edge sequence $E(Tr)$, a time sequence $T(Tr)$, a time flag bit-string $T'(Tr)$, and a relative distance sequence $D(Tr)$. Fig. 2a shows an example of *Tr*, where an object moves from $(v_1 \rightarrow v_2)$ to $(v_7 \rightarrow v_8)$ in sequence, and l_i $(0 \leq i \leq 6)$ are the mapped locations. The TED representation of *Tr* is shown in Table 2.

Edge Sequence. $E(Tr)$ is represented by a start vertex v followed by a sequence of outgoing edge numbers.

Definition 6. *The outgoing edge number n_o (≥ 1) of an edge $(v_s \rightarrow v_e)$ means that $(v_s \rightarrow v_e)$ is the $n_o{}^{th}$ exit edge of v_s.*

In Fig. 2a, the edge sequence of *Tr* can be straightforwardly represented as $\langle (v_1 \rightarrow v_2), (v_2 \rightarrow v_3), ..., (v_7 \rightarrow v_8) \rangle$. Let ID of v_1 be 185190, and assume that $(v_1 \rightarrow v_2)$ is labeled as the first outgoing edge w.r.t. v_1. Then $(v_1 \rightarrow v_2)$ can be represented as 185190 → 1. Here, we keep the ID of v_1 to identify the start vertex of the path. Next, if $(v_2 \rightarrow v_3)$ is assigned as the second outgoing edge w.r.t. v_2, $(v_2 \rightarrow v_3)$ is encoded as 2. To capture that an edge $(v_s \rightarrow v_e)$ has r $(r > 1)$ GPS points located on it, TED includes $(r-1)$ 0s after $(v_s \rightarrow v_e)$'s outgoing edge number w.r.t. v_s. As shown in Table 2, the 0 after the 2 (2 is the outgoing edge number of $(v_5 \rightarrow v_6)$ w.r.t. v_5) in $E(Tr)$ indicates that $(v_5 \rightarrow v_6)$ has two mapped locations, i.e., l_2 and l_3.

Time Sequence. $T(Tr)$ is represented by omitting the consecutive timestamps with **unchanged** sample intervals. For example, $\langle t_i, t_{i+1}, t_{i+2} \rangle$ is encoded as $\langle (i, t_i), (i+2, t_{i+2}) \rangle$ if $t_{i+2} - t_{i+1} = t_{i+1} - t_i$. However, statistics from real-life datasets show that the actual sample intervals vary frequently over time. To be specific, the sample interval changes every 6.80, 2.32 and 1.97 sample intervals on average on three real-life datasets, namely the Denmark (DK), Chengdu (CD), and Hangzhou (HZ) datasets, respectively. This translates into redundant representations and subsequent poor compression ratios for TED. We propose a new compression scheme that tackles this problem (Section 4.1).

Time Flag Bit-String. $T'(Tr)$ is a time flag bit-string that maps timestamps in $T(Tr)$ to outgoing edge numbers in $E(Tr)$. As shown in Table 2, the mapping between the timestamps in $T(Tr)$ and outgoing edge numbers in $E(Tr)$ is not a one-to-one mapping. The reason is that an edge used by a trajectory may not contain any mapped location. For example, the fourth bit 0 of $T'(Tr)$ indicates that no GPS point is mapped to $(v_4 \rightarrow v_5)$; otherwise, it is set to 1.

Relative Distance Sequence. $D(Tr)$ is a sequence of relative distances of the mapped locations on their edges.

Definition 7. *Given a mapped location $l = \langle (v_s \rightarrow v_e), ndist \rangle$, the* relative distance *rd of l w.r.t. $(v_s \rightarrow v_e)$ is the ratio of ndist to the length of $(v_s \rightarrow v_e)$ (denoted as $|(v_s \rightarrow v_e)|$).*

For example in Fig. 2a, *rd* of $l_6 = \langle (v_7 \rightarrow v_8), ndist \rangle$ w.r.t. $(v_7 \rightarrow v_8)$ is $\frac{ndist}{|(v_7 \rightarrow v_8)|}$. In the rest of the paper, we use $\langle (v_s \rightarrow v_e), ndist \rangle$ and $\langle (v_s \rightarrow v_e), rd \rangle$ to represent a mapped location l interchangeably, as they are semantically equivalent.

2.3 Compression with TED

We illustrate TED's compression [40] of $E(Tr)$, $T(Tr)$, $T'(Tr)$, and $D(Tr)$.

Compression on $E(Tr)$. Each $E(Tr)$ has a start vertex followed by a sequence of outgoing edge numbers, as illustrated in Section 2.2. TED compresses $E(Tr)$ using the following three steps: i) encoding each outgoing edge number using $\lceil \log_2(o) \rceil$ bits, where o is the maximal number of outgoing edges for all vertices $v \in V$; ii) grouping trajectories by the length of the binary code of $E(Tr)$, and forming an $A \times B$ binary code matrix for each group, where A

is the number of trajectories and B is the length of $E(Tr)$; iii) applying multiple bases-based compression to each matrix, based on the observation that the highest bit of each code in the matrix has a high probability of being 0. Although the last two steps improve the compression ratio, we do not adopt them in our proposal due to two reasons. First, additional cost must be expended to group trajectories according to their lengths. Second, they require extra space for storing the auxiliary information and extra time for matrix operations during the multiple bases-based compression.

Compression on $T(Tr)$ and $T'(Tr)$. Each element in $T(Tr)$ is binary encoded while $T'(Tr)$ are bit-strings that are compressed using an existing bitmap compression algorithm [33].

Compression of $D(Tr)$. $D(Tr)$ is encoded by using a distance-preserving scheme. The binary code $C(rd)$ of a relative distance rd $(0 \leq rd < 1)$ is defined as $C(rd) = \sum_{i=0}^{I} C(rd_{x_i}) \frac{1}{2^i}$, where $C(rd_{x_i})$ is its i^{th} bit $(i \geq 0)$. Given an error bound η, I equals the smallest number of bits having $|C(rd) - rd| \leq \eta$. In addition, a PDDP-tree is proposed to further reduce the storage cost.

3. FRAMEWORK

We study the compression and subsequent querying of network-constrained uncertain trajectories (NCUTs). We take $\mathbf{Tu} = \{Tu^j | 1 \leq j \leq M\}$, where M is the number of uncertain trajectories, as the input to our framework. Each Tu^j contains a set of instances Tu_w^j $(1 \leq w \leq N^j)$ consisting of i) $U_e(Tu_w^j)$ that is the edge sequence of Tu_w^j; ii) $U_d(Tu_w^j)$ that is the relative distances of all mapped locations; iii) $U_{t'}(Tu_w^j)$ that is the time flag bit-string to associate timestamps with edges; and iv) $U(Tu_w^j).p$ that is the probability associated with Tu_w^j. All Tu_w^j $(1 \leq w \leq N^j)$ share the same time sequence $U_t(Tu^j)$. Fig. 3 depicts our framework for Uncertain Trajectory Compression and Querying (UTCQ) that encompasses three steps, where the red dashed box encloses the input and output to the steps, the blue dashed boxes capture the operations conducted during each step, and the orange dashed boxes capture techniques corresponding to each step.

The ***trajectory representor*** converts the NCUTs into a new format. This primarily involves three operations, i.e., improved TED representation, reference selection, and referential representation, as described below:

i) We separate the start vertices from the edge sequences $U_e(\mathbf{Tu}) = \{U_e(Tu^j) | Tu^j \in \mathbf{Tu}\}$ to obtain $SV(\mathbf{Tu})$ and $E(\mathbf{Tu})$, to achieve a more compact format. Moreover, we propose a new scheme to represent the time sequences $U_t(\mathbf{Tu}) = \{U_t(Tu^j) | Tu^j \in \mathbf{Tu}\}$ to achieve a high compression ratio. The details are presented in Section 4.1.

ii) The representation of path information $SV(Tu^j)$ and $E(Tu^j)$ of each uncertain trajectory Tu^j is given to the *reference selector* (Fig. 3①), that selects one or more high-quality reference instances for each uncertain trajectory. For each selected reference Ref_i^j, the *reference selector* finds a set of *non-references* (denoted as $Ref_i^j.Rrs$), i.e., other instances that can be represented by Ref_i^j. The reference selector sends the above results back to the improved TED representation step to identify the references as well as their non-references (Fig. 3②). The detailed algorithm is covered in Section 4.3.

iii) For each uncertain trajectory Tu^j, the non-reference instances of Tu^j are represented according to their references, by applying a referential representation method (Fig. 3③). Several formats are proposed to represent the non-references compactly. The details are provided in Section 4.2.

The ***trajectory compressor*** compresses the references and non-references into binary codes. Specifically, we adapt Exp-Golomb encoding [32] to compress $T(Tu^j)$, and we apply variable-length encoding to compress each non-reference based on its reference, in order to further reduce the storage cost. The detailed method is covered in Section 4.4.

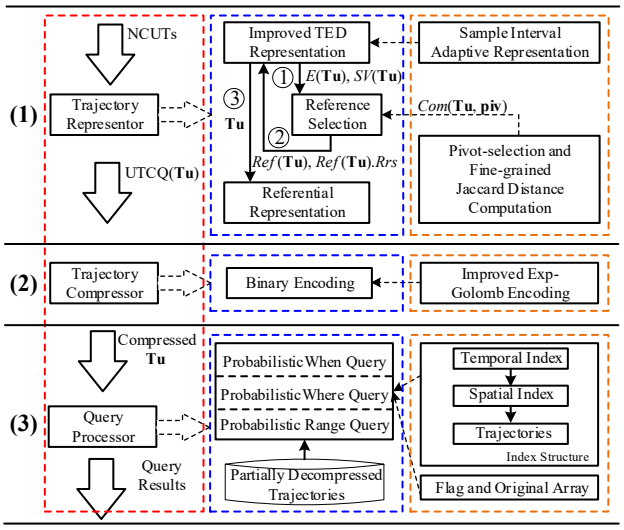


Figure 3: Framework

The ***query processor*** is equipped with efficient algorithms that only partially decompress compressed trajectories. Specifically, we construct and use two auxiliary data structures to facilitate decompression of necessary information of non-references. In addition, we build a spatio-temporal index during compression that can effectively reduce the search space without the need for full decompression. The details are presented in Section 5.

4. REPRESENTOR AND COMPRESSOR

We present the expression and encoding scheme designed for uncertain trajectories. We use the running example depicted in Fig. 2, which contains three instances (i.e., Tu_1^1, Tu_2^1, and Tu_3^1) of an uncertain trajectory Tu^1.

4.1 Improved TED Representation

We denote each uncertain trajectory instance Tu_w^j $(1 \leq w \leq N^j)$ of a particular NCUT Tu^j as a tuple $(SV(Tu_w^j), E(Tu_w^j), D(Tu_w^j), T'(Tu_w^j), Tu_w^j.p)$. Table 3 shows how the example in Fig. 2 is represented.

$SV(Tu_w^j)$. $SV(Tu_w^j)$ is the start vertex ID of the first edge traversed by Tu_w^j. In Table 3, $SV(Tu_1^1) = 185190$, which is the ID of v_1 in Fig. 2a.

$D(Tu_w^j)$ and $Tu_w^j.p$. $D(Tu_w^j)$ and $Tu_w^j.p$ are the relative distance sequence (cf. Definition 7) and probability generated via the probabilistic map-matching process. As Tu^j consists of N^j instances, we have $\sum_{w=1}^{N^j} Tu_w^j.p = 1$.

$E(Tu_w^j)$. $E(Tu_w^j)$ is the edge sequence of Tu_w^j (exclude the start vertex). We adopt the representation of edge sequence used by TED to represent it.

$T'(Tu_w^j)$. We represent the time flag bit-string $T'(Tu_w^j)$ by modifying the TED representation slightly. The first and last edges traversed by Tu_w^j must each have at least one GPS point mapped onto them, so the first and last bits of $T'(Tu_w^j)$ must be 1. As a result, we omit the first and last bits when representing $T'(Tu_w^j)$, to improve the compression ratio.

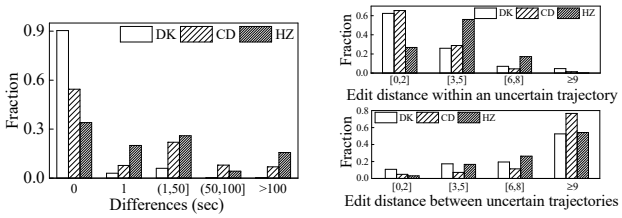
The sample interval is unstable in real-life applications, and TED has a problem when representing such trajectories. In order to tackle

Table 3: Example of improved TED representation of Tu^1 in Fig. 2

w	1	2	3
$SV(Tu_w^1)$	185190	185190	185190
$E(Tu_w^1)$	⟨1, 2, 1, 2, 2, 0, 4, 1, 0⟩	⟨1, 1, 1, 2, 2, 0, 4, 1, 0⟩	⟨1, 2, 1, 2, 2, 0, 4, 1, 2⟩
$D(Tu_w^1)$	⟨0.875, 0.25, 0.5, 0.875, 0.5, 0, 0.875⟩	⟨0.875, 0.25, 0.5, 0.875, 0.5, 0, 0.875⟩	⟨0.875, 0.25, 0.5, 0.875, 0.5, 0, 0.5⟩
$T'(Tu_w^1)$	⟨0, 1, 0, 1, 1, 1, 1⟩	⟨1, 0, 0, 1, 1, 1, 1⟩	⟨0, 1, 0, 1, 1, 1, 1⟩
$Tu_w^1.p$	0.75	0.2	0.05

Table 4: An referential representation example for Table 3

ϕ	$Com_\phi(Nref_{11}^1, Ref_1^1)$	$Com_\phi(Nref_{12}^1, Ref_1^1)$
SV	$\emptyset$	$\emptyset$
E	⟨(0, 1, 1), (2, 7)⟩	⟨(0, 8, 2)⟩
D	$\emptyset$	⟨(6, 0.5)⟩
T'	⟨(1, 2), (3, 4)⟩	$\emptyset$



(a) Sample intervals (b) Similarity

Figure 4: Statistics of real-life datasets

this problem, we develop a new representation scheme to represent $T(Tu^j)$, namely Sample Interval Adaptive Representation (SIAR) .

Sample Interval Adaptive Representation of $T(Tu^j)$. Fig. 4a counts the differences between the actual sample intervals and the default ones using three real-life datasets, i.e., DK, CD, and HZ. As can be observed, most of the actual sample intervals (93% in DK, 62% in CD, and 54% in HZ) are equal to or deviate only 1 second from the default one. Motivated by this, we only record the *difference* between the (actual) sample interval and the default one. Assume that the time sequence of Tu^j starts with t_0 and that T_s is the default sample interval. SIAR keeps t_0 as the first value in $T(Tu^j)$ and represents the following timestamps as $(t_{i+1} - t_i) - T_s$, where t_i is the i^{th} timestamp. Given the time sequence ⟨5:03:25, 5:07:25, 5:11:26, 5:15:26, 5:19:25, 5:23:25, 5:27:25⟩, an example of SIAR used in UTCQ is $T(Tu^1) = ⟨5:03:25, 0, 1, 0, -1, 0, 0⟩$, where the default sample interval is 240 sec. In contrast, TED represents the sequence as ⟨(0, 5:03:25), (1, 5:07:25), (2, 5:11:26), (3, 5:15:26), (4, 5:19:25), (6, 5:27:25)⟩, because most of its adjacent (actual) sample intervals are different. Hence, SIAR achieves a more compact representation when the sample interval varies frequently.

4.2 Referential Representation

The referential representation encodes the differences of an input sequence w.r.t. a reference sequence by exploiting the similarity between them (i.e., the more similar the sequences are, the higher the compression ratio). It is a **lossless** encoding [10, 34, 35]. Fig. 4b shows statistics on the similarity between trajectory instances in DK, CD, and HZ. Here, we use edit distance to measure the similarity of $E(\cdot)$ between two instances as in [37, 43]. As can be observed, the edit distance between most of the instances of a particular uncertain trajectory (88% in DK, 94% in CD, and 83% in HZ) is at most 5, while that between most of the instances from different uncertain trajectories (53% in DK, 77% in CD, and 54% in HZ) is no less than 9. Hence, we only apply the referential representation to the trajectory instances of an uncertain trajectory rather than to the instances of different uncertain trajectories, to guarantee high compression ratios. For each uncertain trajectory, we select one or more instances as references, i.e., reference trajectory instances (see Section 4.3 for the selection). Then, other instances can be represented according to their reference using a set of *factors* defined below.

Definition 8. *Given a non-reference $Nref_{ik}^j$ and its corresponding reference Ref_i^j, $Nref_{ik}^j$ can be expressed as a list of factors, i.e., $Com_\phi(Nref_{ik}^j, Ref_i^j) = ⟨\phi_{ik}^j(Ma_h) | 1 \le h \le H⟩$, where H is the number of factors, and a factor $\phi_{ik}^j(Ma_h)$ denotes a subsequence in $Nref_{ik}^j$.*

Since one reference can be used for representing multiple non-references, we use the *referential representation set $Ref_i^j.Rrs$* to denote the set of $Nref_{ik}^j$ ($1 \le k \le |Ref_i^j.Rrs|$) represented by Ref_i^j. Table 4 shows the referential representation of Table 3, where Tu_1^1 is selected as the reference Ref_1^1 and is used to represent Tu_2^1 and Tu_3^1 (also called $Nref_{11}^1$ and $Nref_{12}^1$). The detailed representation is explained below.

$E(Nref_{ik}^j)$. Several strategies exist for encoding a factor [10, 34, 35]. We adopt the (S, L, M) representation [35] to encode each factor of $Com_E(Nref_{ik}^j, Ref_i^j)$, as it has a high compression ratio when the to-be-compressed sequence and the reference are highly similar. Specifically, S is the *start position* of the subsequence in the reference, L is the *length* of the subsequence, and M is the first *mismatched element* following the subsequence. However, we rewrite the form of each factor in two cases:

A) When M does not exist (no mismatch), we record the factor as (S, L), omitting M. This does not introduce any ambiguity, since (S, L) only occurs at the end of the factor list.

B) When an outgoing edge number (denoted as n_o) in $E(Nref_{ik}^j)$ does not exist in $E(Ref_i^j)$, we denote the factor by assigning $S = |E(Ref_i^j)|$ and $M = n_o$, where $|E(Ref_i^j)|$ is the length of $E(Ref_i^j)$. The idea is to append n_o to the end of the reference, i.e., to consider n_o as its last value. We further omit L as it always equals 1. Hence, the factor in this case has the form (S, M). For example, given $E(Tu_4^1) = ⟨3, 2, 1, 2, 2⟩$, we have $E_{13}^1(Ma_1) = (9, 3)$, by referentially representing Tu_4^1 (denoted as $Nref_{13}^1$).

$T'(Nref_{ik}^j)$. The referential representation of $T'(Nref_{ik}^j)$ is similar to that of $E(Nref_{ik}^j)$. We represent each factor using the format (S, L) because M can be inferred easily from the reference. To be specific, if the bit immediately following the longest prefix in $T'(Ref_i^j)$ is 1, the first mis-matched bit in $T'(Nref_{ik}^j)$ is 0, i.e., $M = 0$; otherwise, $M = 1$. Specifically, we always keep the last factor in the form (S, L, M) when M exists in order to avoid ambiguity. Then, if $T'(Nref_{ik}^j)$ is exactly the same as $T'(Ref_i^j)$, $Com_{T'}(Nref_{ik}^j, Ref_i^j) = \emptyset$ (as shown in Table 4).

$D(Nref_{ik}^j)$. We observe that even though a raw positional point could be mapped to two different edges via the probabilistic map-matching, the mapped locations may have the same relative distance, as the GPS records shown in Fig. 1. Based on this observation, we encode each factor in $D(Nref_{ik}^j)$ using the format (pos, rd),

where *pos* is the position of the different value *rd*. Note that this strategy is not good for referentially representing $E(Nref_{ik}^j)$ and $T'(Nref_{ik}^j)$ because their lengths vary across the instances of a single uncertain trajectory Tu^j.

Finally, we omit $SV(Nref_{ik}^j)$. Because we do not choose a non-reference $Nref_{ik}^j$ that has a different start vertex than $Rref_i^j$. Also, we do not referentially compress $T(Nref_{ik}^j)$ or $Nref_{ik}^j.p$, because i) all instances for a single uncertain trajectory Tu^j share the same $T(Tu^j)$, and ii) the probability $Nref_{ik}^j.p$ has quite different values.

4.3 Reference Selection

Intuitively, the more similar a reference and a non-reference are, the higher the compression ratio. A naive reference selection strategy is to try every instance as the reference. However, its cost is too high. Inspired by an existing study [35], we use the similarity between the referential representations of trajectory instances to approximate their exact one. Here, we first select a set of *pivots*, and then represent each instance using these pivots. Thereafter, a *fine-grained Jaccard Distance function* is defined to estimate the similarity between two represented instances.

Pivot Selection. We select a set of pivots $\{piv_i | 1 \leq i \leq n_p^j\}$ from Tu^j. High-quality pivots are usually far away from each other and far away from other instances [7]. Hence, we i) randomly choose a trajectory instance and referentially represent all the remaining ones according to it; ii) select the one with the most factors as a pivot; iii) referentially represent all the trajectory instances using the most recently selected pivot; and iv) we repeat steps ii) and iii) until enough pivots are selected. Note that we only referentially represent $E(\cdot)$ of each trajectory instance by that of each pivot, as it is sufficient to distinguish the distances between instances.

Pivot Representation. We adopt the format (S, L) [10] to represent each factor in step iii). An example of (S, L) representation in Table 3 is $Com_E(Tu_1^1, piv_1) = \langle (0, 8), (5, 1) \rangle$, where $piv_1 = Tu_3^1$. If an outgoing edge number in $E(Tu_w^j)$ does not exist in $E(piv_i)$, we omit the factor but increase the number of factors by 1. After pivot selection and representation, we get the referential representations of each trajectory Tu^j w.r.t. a set of pivots, i.e., $Com_E(Tu_w^j, piv_i)$ $(1 \leq w \leq N^j, 1 \leq i \leq n_p^j)$. The time complexity of pivot selection and representation for a Tu^j is $O(N^j \cdot n_p^j \cdot avg(|E|) \cdot avg(|Com_E|)$, where n_p^j is the number of pivots selected for Tu^j, $avg(|E|)$ is the average length of $E(\cdot)$ of all instances of Tu^j, and $avg(|Com_E|)$ is the average length of all instances w.r.t. all pivots of Tu^j.

Fine-grained Jaccard Distance Function. Given two trajectory instances Tu_w^j and Tu_v^j, and a pivot piv_i, we use the similarity between $Com_E(Tu_w^j, piv_i)$ and $Com_E(Tu_v^j, piv_i)$ to estimate the similarity between $E(Tu_w^j)$ and $E(Tu_v^j)$. Previous work [35] uses the Jaccard Distance to measure the similarity. However, this distance is inaccurate in some cases. Given $piv_1 = Tu_3^1$ and $E(Tu_5^1) = \langle 1, 2, 1, 2, 2, 0, 4 \rangle$, we have $Com_E(Tu_1^1, piv_1) = \langle (0, 8), (5, 1) \rangle$ and $Com_E(Tu_5^1, piv_1) = \langle (0, 7) \rangle$. Thus, the Jaccard Distance between them is 1. However, $E(Tu_1^1)$ is actually very similar to $E(Tu_5^1)$. To obtain a more fine-grained distance notion, we propose a new distance metric, called *Fine-grained Jaccard Distance* (*FJD*), to calculate the distance from $E(Tu_w^j)$ to $E(Tu_v^j)$ against a pivot piv_i.

$$FJD(Tu_w^j \rightarrow Tu_v^j, piv_i)(w \neq v) = \frac{\sum_{h'=1}^{H'} sim(E_{iv}^j(Ma_{h'}), Com_E(Tu_w^j, piv_i))}{\max\{H, H'\}}, \quad (1)$$

where H and H' denote the number of factors in $Com_E(Tu_w^j, piv_i)$ and $Com_E(Tu_v^j, piv_i)$, respectively, while $E_{iv}^j(Ma_{h'})$ denotes the h'^{th} factor $(S_{h'}^{iv}, L_{h'}^{iv})$ in $Com_E(Tu_v^j, piv_i)$. In addition, we use $sim(E_{iv}^j(Ma_{h'}), Com_E(Tu_w^j, piv_i))$ to measure the similarity between $E_{iv}^j(Ma_{h'})$ and $Com_E(Tu_w^j, piv_i)$, calculated as follows,

$$sim(E_{iv}^j(Ma_{h'}), Com_E(Tu_w^j, piv_i)) = \frac{\max_{h=1}^{H}(E_{iw}^j(Ma_h) \cap E_{iv}^j(Ma_{h'}))}{\max\{L_{\max}^{iw}, L_{h'}^{iv}\}} \quad (2)$$

We define $E_{iw}^j(Ma_h) \cap E_{iv}^j(Ma_{h'})$ as $\max\{\min\{S_h^{iw} + L_h^{iw}, S_{h'}^{iv} + L_{h'}^{iv}\} - \max\{S_h^{iw}, S_{h'}^{iv}\}, 0\}$, and $L_{\max}^{iw} = \arg\max_{L_h^{iw}} E_{iw}^j(Ma_h) \cap E_{iv}^j(Ma_{h'})$. If $L_{\max}^{iw}$ is not unique, we choose the minimum value.

Example 1. (FJD computation) *Consider the example in Table 3. Given* $piv_1 = Tu_3^1$, *we have* $Com_E(Tu_1^1, piv_1) = \langle (0, 8), (5, 1) \rangle$ *and* $Com_E(Tu_2^1, piv_1) = \langle (0, 1), (0, 1), (2, 6), (5, 1) \rangle$. *Then we calculate* $\frac{E_{11}^1(Ma_1) \cap E_{12}^1(Ma_1)}{\max\{L_1^{11}, L_1^{12}\}} = \frac{1}{8}$, *in order to get* $sim(E_{12}^1(Ma_1), Com_E(Tu_1^1, piv_1))$. *Similarly, we are able to gain* $sim(E_{12}^1(Ma_2), Com_E(Tu_1^1, piv_1)) = \frac{1}{8}$, $sim(E_{12}^1(Ma_3), Com_E(Tu_1^1, piv_1)) = \frac{3}{4}$, *and* $sim(E_{12}^1(Ma_4), Com_E(Tu_1^1, piv_1)) = 1$. *Hence,* $FJD(Tu_1^1 \rightarrow Tu_2^1, piv_1) = (\frac{1}{8} + \frac{1}{8} + \frac{3}{4} + 1)/4 = \frac{1}{2}$.

Based on $FJD(Tu_w^j \rightarrow Tu_v^j, piv_i)$, we present our score function $SF(Tu_w^j, Tu_v^j)$ $(w \neq v)$ for representing Tu_v^j by Tu_w^j, which is calculated as $Tu_w^j.p \cdot \max_{i=1}^{n_p^j} FJD(Tu_w^j \rightarrow Tu_v^j, piv_i)$. Here, a trajectory instance with higher probability of occurrence is expected to get a higher chance to be a reference, in order to speed up decompression during querying. Therefore, we multiply $Tu_w^j.p$ with the maximum *FJD* value. $SF(Tu_w^j, Tu_w^j)$ $(1 \leq w \leq N^j)$ is set to 0, as we do not consider the case of representing a trajectory instance by itself. In addition, we calculate $SF(Tu_w^j, Tu_w^j)$ only when $SV(Tu_w^j) = SV(Tu_v^j)$, because two trajectory instances with different start vertices usually are not similar to each other. According to the score function *SF*, the optimal reference for Tu_v^j can be derived by the following formula:

$$Ref(Tu_v^j) = \arg\max_{Tu_w^j} SF(Tu_w^j, Tu_v^j) \quad (3)$$

There are two constraints in our setting: i) each non-reference only has one reference, to avoid redundancy; and ii) we only consider single-order compression. Our goal of reference selection is to maximize $\sum_{Tu_w^j, Tu_v^j \in Tu^j} SF(Tu_w^j, Tu_v^j)$ for an uncertain trajectory Tu^j under these two constraints. Unfortunately, we have to enumerate all the possible combinations to get the best selection choice, which is unfeasible, as the enumeration cost is $O(((N^j)^2)!)$ for Tu^j.

Therefore, we propose a greedy algorithm, shown in Algorithm 1, for selecting the references for each uncertain trajectory Tu^j $(1 \leq j \leq M)$. By applying *SF* to each pair of instances of an uncertain trajectory Tu^j, we can get a score matrix **SM**, where $SM[w][v] = SF(Tu_w^j, Tu_v^j)$ is the score of representing Tu_v^j by Tu_w^j. Algorithm 1 shows that we always choose the maximal element from **SM**, since it represents the current best reference. After each selection, we delete the elements in **SM** that do not satisfy the constraints (Line 7 and 9). The above-mentioned procedure is repeated until **SM** $= \emptyset$ or the current maximum of it is 0. In the latter case, the trajectory instances that have not been selected are formally added to the reference set of Tu^j for easier query processing but are not associated with a reference representation set (Lines 11–13). The efficiency of Algorithm 1 can be further improved by pre-sorting the elements in **SM**. The time complexity of reference selection for Tu^j is $O(N^j \cdot n_p^j \cdot avg(|E|) \cdot avg(|Com_E|) + N^{j^2} \cdot n_p^j \cdot avg(|Com_E|)^2 + N^{j^2} \cdot 2\log_2^{N^j})$, while the space complexity is $O(N^j \cdot n_p^j \cdot avg(|Com_E|) + (N^j)^2 \cdot n_p^j)$.

Algorithm 1: Reference Selection Algorithm

Input: **SM** of Tu^j
Output: the reference set $Ref(Tu^j)$ of Tu^j, and the referential representation set of each reference in $Ref(Tu^j)$ if it exists

initialize an empty reference set $Ref(Tu^j)$
while SM $\neq \emptyset$ **do**
  select the maximum score $SM[w][v]$ from **SM**
  if $SM[w][v] > 0$ **then**
    if $Tu^j_w \notin Ref(Tu^j)$ **then**
      add Tu^j_w to $Ref(Tu^j)$, and create $Tu^j_w.Rrs$
      remove $SM[v'][w]$ from **SM** $(1 \leq v' \leq N^j)$
    add Tu^j_v to $Tu^j_w.Rrs$
    remove $SM[w'][w]$, $SM[v][w'']$ from **SM** $(1 \leq w', w'' \leq N^j)$
  else
    for *each diagonal element* $SM[w][w]$ *in* **SM do**
      if $SM[w][w]$ *exists* **then**
        add Tu^j_w to $Ref(Tu^j)$
return $Ref(Tu^j)$ and the non-empty $Tu^j_w.Rrs$ for each $Tu^j_w \in Ref(Tu^j)$

Example 2. (Algorithm 1 overview) *Assuming that we only select Tu^1_3 as a pivot for Tu^1, we get an* **SM**. *Then we find the maximum in* **SM**, *i.e., $SF(Tu^1_1, Tu^1_2)$, based on which we get a reference Tu^1_1 and add Tu^1_2 to its Rrs. Afterwards, $SM[w'][2] \cup SM[2][w''] \cup SM[v'][1]$ $(1 \leq w', w'', v' \leq 3)$ are removed from* **SM** *according to the two constraints. This process is shown below,*

$$\mathbf{SM} = \begin{bmatrix} 0 & \frac{3}{8} & \frac{1}{3} \\ \frac{7}{80} & 0 & \frac{1}{30} \\ \frac{1}{40} & \frac{1}{80} & 0 \end{bmatrix} \rightarrow \begin{bmatrix} \cancel{0} & \cancel{\frac{3}{8}} & \frac{1}{3} \\ \cancel{\frac{7}{80}} & \cancel{0} & \cancel{\frac{1}{30}} \\ \cancel{\frac{1}{40}} & \cancel{\frac{1}{80}} & 0 \end{bmatrix}$$

Then we add Tu^1_3 to $Tu^1_1.Rrs$ due to $SM[1][3] > SM[3][3]$, and remove $SM[w'][3] \cup SM[3][w''] (1 \leq w', w'' \leq 3)$ from **SM**. *Finally, since* **SM** $= \emptyset$, *we return the reference Tu^1_1 with its Rrs $= \{Tu^1_2, Tu^1_3\}$ for Tu^1.*

4.4 Compression

Binary Encoding References. We follow TED [40] to compress $E(Ref)$. As discussed in Section 2.2, we omit the time-consuming procedures of TED. In this case, UTCQ still outperforms TED in terms of compressing ratio due to the referential representation and compression, while significantly improving the compression efficiency. Also, we adopt the PDDP-tree [40], which is the only lossy component in our framework, to encode $D(Ref)$ and $Ref.p$ that are floats. We use the error bounds η_D and η_p to constrain their compression accuracy. Both η_D and η_p are pre-set compression parameters, and the actual errors between the original and compressed data are constrained by these settings. $T'(Ref)$ is already represented as bit strings as discussed in Section 4.1 and does not need any further compression. Moreover, we propose an improved Exp-Golomb encoding to compress $T(Ref)$, which effectively addresses the sample interval fluctuation.

Improved Exp-Golomb Encoding. As different deviations between the actual sample interval and the default occur with different frequencies, encoding each value of $T(Tu^j)$ in a binary code with fixed length may waste space. Specifically, we find that small deviations are much more frequent than large ones. The statistics in Fig. 4a exemplify this. Thus, we adopt the well-known Exp-Golomb encoding [32] to compress $T(Tu^j)$. It encodes smaller values with shorter lengths and larger values with longer lengths. We set the parameter k, which controls the length of the first group, to 0. Given timestamps t_i and t_{i+1} of an uncertain trajectory Tu^j with default sample interval T_s, we let $\triangle t_i = (t_{i+1} - t_i) - T_s$ $(0 \leq i < |T(Tu^j)| - 1)$, where $|T(Tu^j)|$ is the length of $T(Tu^j)$. However, since $\triangle t_i$ may be negative, the Exp-Golomb encoding needs to be modified.

Assuming that the longest actual sample interval is T_l, we have $\triangle t_i \in (-T_s, T_l - T_s]$ $(0 \leq i < |T(Tu^j)| - 1)$. We divide $[0, \max\{T_s - 1, T_l - T_s\}]$ into n groups, where $n = \lceil \log_2(\max\{T_s - 1, T_l - T_s\} + 1) \rceil$, and the range of the $j^{th}(j \geq 0)$ group is $\left[-2^{j+1} + 2,\ -2^j + 1\right] \cup \left[2^j - 1, 2^{j+1} - 2\right]$. This way, all the possible deviations between the actual sample interval and the default one can be covered as $[-\max\{T_s - 1, T_l - T_s\}, \max\{T_s - 1, T_l - T_s\}] \subseteq [-2^n + 2, 2^n - 2]$. The offset of $\triangle t_i$ in the j^{th} group is given by $|\triangle t_i| - (2^j - 1)$. Moreover, we add one 1 bit immediately before the offset if $\triangle t_i$ is a negative digit; otherwise, 0 is added. Following the example of SIAR in Section 4.1, $\langle$5:03:25, 0, 1, 0, -1, 0, 0$\rangle$ is encoded as $\langle$00100011100011101, 0, 1000, 0, 1010, 0, 0$\rangle$ by the improved Exp-Golomb encoding. Hence the compression ratio of $T(Tu^1)$ by our method is $\frac{32\times7}{12+17} = 7.72$, while the counterpart by TED is $\frac{32\times7}{(17+12)\times6} = 1.29$, where we assume each trajectory contains at most 2^{12} timestamps and t_i $(0 \leq i < |T(Tu^j)|)$ is encoded using 17 bits.

Binary Encoding Non-references. Let $|E(Ref^j_i)|$ be the length of $E(Ref^j_i)$ and o be the maximum number of outgoing edges for any vertex $v \in V$. Then S takes $\lceil \log_2 |E(Ref^j_i)| + 1 \rceil$ bits, L takes $\lceil \log_2 |E(Ref^j_i)| \rceil$ bits, and M takes $\lceil \log_2 o \rceil$ bits when performing binary encoding of a factor (S, L, M) in $E(Nref^j_{ik})$. Next, S and L in the factor of $T'(Nref^j_{ik})$ are encoded in $\lceil \log_2 |T'(Ref^j_i)| \rceil$ bits, while M takes 1 bit. Further, *pos* in each factor of $D(Nref^j_{ik})$ occupies $\lceil \log_2 |D(Ref^j_{ik})| \rceil$ bits, and *rd* is encoded by a PDDP-tree [40]. It can be seen that, by using referential representation, binary codes of different non-references may have different lengths depending on their similarities to the corresponding references, further saving space. We denote the binary code of a sequence *seq* as $\hat{seq}$ in the rest of the paper, e.g., the binary code of $E(\cdot)$ is denoted as $\hat{E}(\cdot)$.

Overall, the space complexity of compressing Tu^j is $O(N^j \cdot n^j_p \cdot avg(|Com_E|) + (N^j)^2 \cdot n^j_p + size_{in}(Tu^j) + size_{out}(Tu^j))$, where $size_{in}(Tu^j)$ and $size_{out}(Tu^j)$ are the input size and the compressed size of Tu^j, respectively.

5. QUERY PROCESSOR

In this section, we describe how to compute queries directly on compressed uncertain trajectories. First of all, we introduce a strategy to extract the necessary information from compressed *time flag bit-strings* by means of partial decompression. Second, we design an index to achieve fast retrieval and partial decompression. In addition, several pruning techniques are proposed to more efficiently support probabilistic queries on compressed uncertain trajectories.

5.1 Time Flag Bit-string Decompression

We have represented the time flag bit-strings of non-references as a list of factors. Since time flag bit-strings associate $D(\cdot)$ and $T(\cdot)$ with $E(\cdot)$, we need to get the number of 1s before any position in order to support queries over compressed data [40]. Naively, we can first decompress factors of $\hat{Com}_{T'}(Nref^j_{ik}, Ref^j_i)$ to $T'(Nref^j_{ik})$, and then count the number, with the cost $O(|\hat{Com}_{T'}(Nref^j_{ik}, Ref^j_i)| + |T'(Nref^j_{ik})|)$. To accelerate this, we propose an effective method by constructing two assisting arrays, *flag array and original array*.

Flag Array and Original Array. The *flag array* of $T'(Ref^j_i)$ is denoted as $\omega_{T'(Ref^j_i)}$, and counts the number of 1s before the g^{th} (not including g) bit of $T'(Ref^j_i)$ $(0 < g \leq |T'(Ref^j_i)|)$. However,

$T'(Ref_i^j)$ omits the first and last bit of the original time flag bit-string during representation (in Section 4.1). Therefore, we define the *original array*, $\gamma_{T'(Ref_i^j)}$, which records the number of 1s until the g^{th} bit in the original time flag bit-string ($0 \le g < |T'(Ref_i^j)|$). Here, we simplify $\varphi_{T'(Ref_i^j)}$ and $\varphi_{T'(Nref_{ik}^j)}$ by representing them as $\varphi_{Ref_i^j}$ and $\varphi_{Nref_{ik}^j}$, where $\varphi \in \{\omega, \gamma\}$.

For a reference Ref_i^j, it is easy to get $\omega_{Ref_i^j}$ by linearly scanning $T'(Ref_i^j)$. To get $\gamma_{Nref_{ik}^j}$ for a non-reference, we propose a strategy by partially decompressing $Com_{T'}(Nref_{ik}^j, Ref_i^j)$. Given $\omega_{Ref_i^j}$ and g, we can get $\gamma_{Nref_{ik}^j}[g]$ by only decompressing at most one factor in $Com_{T'}(Nref_{ik}^j, Ref_i^j)$. More specifically, we first locate the factor in $Com_{T'}(Nref_{ik}^j, Ref_i^j)$ that the g^{th} bit of the original $T'(Nref_{ik}^j)$ falls into, as follows.

$$\max h \quad s.t.\ h + \sum_{l=1}^{h} L_l^{ik} \le g \wedge h < H, \tag{4}$$

where H is the number of factors in $Com_{T'}(Nref_{ik}^j, Ref_i^j)$ and L_l^{ik} is the length of the subsequence represented by $T'^j_{ik}(Ma_l)$. Formula 4 ensures that the g^{th} bit of the original $T'(Nref_{ik}^j)$ either falls into $T'^j_{ik}(Ma_{h+1})$ or exactly corresponds to the M that is omitted in $T'^j_{ik}(Ma_h)$. Thus we only need to decompress $T'^j_{ik}(Ma_{h+1})$ after calculating the number of 1s within the subsequence before the $(h+1)^{th}$ factor, as follows.

$$\begin{aligned} Z =& 1 + \sum_{l=1}^{h} \omega_{Ref_i^j}[S_l^{ik} + L_l^{ik}] - \omega_{Ref_i^j}[S_l^{ik}] \\ &+ \sim T'(Ref_i^j)[S_l^{ik} + L_l^{ik}], \end{aligned} \tag{5}$$

where $\sim (x)$ means NOT(x), i.e., the neglected mismatched elements of $T'^j_{ik}(Ma_l)$, and S_l^{ik} refers to the start position of the subsequence represented by $T'^j_{ik}(Ma_l)$. Let $g' = g - h - \sum_{l=1}^{h} L_l^{ik}$. Then $\gamma_{Nref_{ik}^j}[g]$ can be derived as follows.

$$\gamma_{Nref_{ik}^j}[g] = Z + \omega_{Ref_i^j}[S_{h+1}^{ik} + g'] - \omega_{Ref_i^j}[S_{h+1}^{ik}], \tag{6}$$

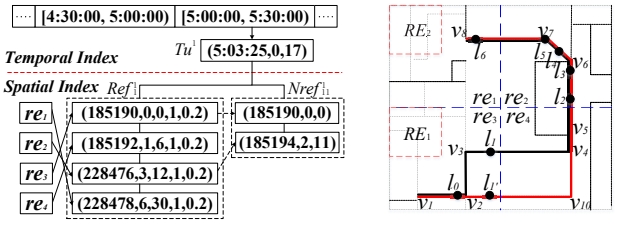
where $g \ge L_1^{ik} + 1 \wedge g < H + \sum_{l=1}^{H} L_l^{ik} (H \ne 1)$.

Hence, we extract the necessary information $\gamma_{Nref_{ik}^j}[g]$ with the cost $O(|\hat{Com}_{T'}(Nref_{ik}^j, Ref_i^j)| + \frac{|T'(Nref_{ik}^j)|}{H})$ if g is given, where $\frac{|T'(Nref_{ik}^j)|}{H}$ is the average length of each factor.

5.2 StIU Index

We propose an index, called Spatio-temporal Information based Uncertain Trajectory Index (StIU), to support efficient probabilistic queries by partial decompression. The partial decompression is lossless and decompresses only the information necessary for answering queries [40]. As shown in Fig. 5a, an StIU index is built on Tu^1, where Tu_1^1 (used as Ref_1^1) and Tu_2^1 (used as $Nref_{11}^1$) are instances of the uncertain trajectory Tu^1 depicted in Fig. 2. Let the IDs of v_1, v_2, v_3, v_4, v_5, and v_7 be 185190, 185191, 185192, 185194, 228476, and 228478, respectively. Assume that the mapping positions of the relative distances of l_1, l_2, and l_5 in $\hat{D}(Ref_1^1)$ are 6, 12, and 30, respectively, and that the maximum outgoing edge number of the road network in Fig. 5b is 7. The index contains two parts. The upper part indexes the temporal information of trajectories, while the lower part supports effective spatial search.

Temporal Index of StIU. We first partition a day into equal-length time intervals. Then, we associate each interval with the



(a) Index Structure (b) Tu_1^1 and Tu_2^1

Figure 5: StIU built on $\boldsymbol{Tu} = \{\boldsymbol{Tu}^1\}$ depicted in Fig. 2

uncertain trajectories, whose timestamps intersect with it. The information on an uncertain trajectory Tu^j corresponding to a time interval is stored in a tuple (*t.start, t.no, t.pos*), where *t.start* is the earliest timestamp of Tu^j falling into the time interval, *t.no* indicates that *t.start* is the $t.no^{th}$ timestamp in $T(Tu^j)$, and *t.pos* refers to the matching position of the $(t.no+1)^{th}$ timestamp in $\hat{T}(Tu^j)$.

Spatial Index of StIU. We further organize the trajectory instances in each time interval according to their spatial information. To be specific, we first partition the road network G using grid cells, each of which represents a region re_i. Then we create tuples, each linking a trajectory instance to a region it has passed. The tuples for a trajectory instance form a chronologically ordered list. For example, in Fig. 5a, the tuple associated with re_3 is followed by that associated with re_4 for Tu_1^1 (used as Ref_1^1). Before detailing the index information stored in each tuple of Tu_w^j, we introduce the concept of final vertex.

Definition 9. *A* final vertex *of a trajectory instance Tu_w^j w.r.t. a region re is a vertex in G that is traversed by the trajectory instance immediately before reaching re, denoted as $Tu_w^j.fv$ of re.*

For example, in Fig. 5b, v_3 is $Ref_1^1.fv$ w.r.t. re_4. Then, we introduce the format of a tuple for a reference Ref_i^j associated with a region *re*, if $(Ref_i^j.Rrs \cup Ref_i^j) \cap re \ne \emptyset$. Accordingly, there are two possible cases for Ref_i^j if it has a tuple corresponding to *re*: i) Ref_i^j passed *re* itself; ii) Ref_i^j did not pass *re* but $\exists Nref_{ik}^j \in Ref_i^j.Rrs$ s.t. $Nref_{ik}^j$ passed *re*.

For the first case, the tuple corresponding to *re* is stored as (*fv.id*, *fv.no*, *d.pos*, p_{total}, $p_{\max}$), where 1) *fv.id* (≥ 0) is the ID of $Ref_i^j.fv$ w.r.t. *re*; 2) *fv.no* indicates the position of *fv.id* in $E(Ref_i^j)$; 3) *d.pos* is the matching position of the $d.no^{th}$ relative distance in $\hat{D}(Ref_i^j)$, where *d.no*=$\gamma_{Ref_i^j}[fv.no]$; 4) with Ω defined as the subset of all trajectory instances $Ref_i^j \cup Ref_i^j.Rrs$ that overlap *re*, p_{total} is then the sum of the probabilities of all instances in Ω; and 5) $p_{\max} = Nref_{ik'}^j.p$ such that $\forall Nref_{ik}^j \in \Omega$: $Nref_{ik'}^j.p \ge Nref_{ik}^j.p$. If $\forall Nref_{ik}^j \in Ref_i^j.Rrs$, $Nref_{ik}^j$ does not overlap *re*, $p_{\max}$ is set to 0.

For the second case, each tuple has the form $(fv.id, p_{total}, p_{\max})$. Specifically, we set $fv.id = \infty$, which indicates that Ref_i^j itself did not traverse *re*, and p_{total} and $p_{\max}$ are the same as for the first case.

The tuple for a non-reference $Nref_{ik}^j$ w.r.t. *re* has the form (*rv.id*, *rv.no*, *ma.pos*), where 1) *rv.id* is the ID of the first vertex *rv* represented in $E_{ik}^j(Ma_h)$ (the h^{th} factor of $Com_E(Nref_{ik}^j, Ref_i^j)$), such that $E_{ik}^j(Ma_h)$ contains $Nref_{ik}^j.fv$ of *re*; 2) *rv.no* indicates the position of *rv* in $E(Nref_{ik}^j)$; and 3) *ma.pos* is the matching position of $E_{ik}^j(Ma_h)$ in $\hat{Com}_E(Nref_{ik}^j, Ref_i^j)$. In the case when a factor "crosses" more than one region, we only keep the tuple for the region that the trajectory instance traverses first. In the case when *re* is the first region traversed by Tu_w^j or $Tu_w^j.fv$ of *re* is exactly $SV(Tu_w^j)$, we store $(SV(Tu_w^j), 0, 0, p_{\max}, p_{total})$ if Tu_w^j is a reference, and we store $(SV(Tu_w^j), 0, 0)$ if Tu_w^j is a non-reference.

5.3 Probabilistic Queries

Based on StIU, three representative types of queries, namely probabilistic where, when, and range queries, can be performed.

Definition 10. **(Probabilistic where query)** *Given a timestamp t, a probability α, and a compressed uncertain trajectory Tu^j, a probabilistic where query **where**(Tu^j, t, α) returns the set of mapped locations at time t of the instances $Tu_w^j \in Tu^j$ with $Tu_w^j.p \geq \alpha$. Each location is given as $\langle (v_s \rightarrow v_e), ndist \rangle$, where $(v_s \rightarrow v_e)$ is the edge traversed by Tu_w^j, and ndist is the network distance between v_s and the location at t.*

Definition 11. **(Probabilistic when query)** *Given a mapped location $\langle (v_s \rightarrow v_e), rd \rangle$, a probability α, and a compressed uncertain trajectory Tu^j, a probabilistic when query **when**$(Tu^j, \langle (v_s \rightarrow v_e), rd \rangle, \alpha)$ returns the set of timestamps, where rd is the relative distance of the location w.r.t. $(v_s \rightarrow v_e)$, and each timestamp t corresponds to a instance Tu_w^j of Tu^j with $Tu_w^j.p \geq \alpha$, such that Tu_w^j passed $\langle (v_s \rightarrow v_e), rd \rangle$ at t.*

Example 3. (Probabilistic where and when query) *Let the IDs of v_6 and v_7 be* 228477 *and* 228478*, respectively, and the length of the edge $(v_6 \rightarrow v_7)$ be* 200*. Given a query **where**$(Tu^1,$ 5:21:25, 0.25$)$ and assume that the time partition duration of StIU is* 15 *minutes, we locate the tuple* $(5{:}15{:}26, 3, 23)$ *through binary search, as its t.start is the closest timestamp to* 5:21:25 *with t.start*$\leq$ 5:21:25*, and then decompress $\hat{T}(Tu^1)$ from its 23^{th} bit. In this way, we get the result $\langle 228477 \rightarrow 228478, 150 \rangle$ without full decompression. Similarly, given the road network partition shown in Fig. 5b and a query **when**$(Tu^1, \langle 228477 \rightarrow 228478, 0.75 \rangle, 0.25)$, we search from v_5 according to the tuple $(228476, 3, 12, 1, 0.2)$ in StIU and return* 5:21:25*.*

Definition 12. **(Probabilistic range query)** *Given a query region RE, a timestamp t_q, and a collection of compressed uncertain trajectories* **Tu***, a probabilistic range query **range**$(\mathbf{Tu}, RE, t_q, \alpha)$ returns the set of uncertain trajectories $Tu^j (1 \leq j \leq M)$ in* **Tu***, such that $\sum_{Tu_w^j \in Tu^j \wedge Tu_w^j \cap RE \neq \emptyset} Tu_w^j.p \geq \alpha$ at t_q.*

Example 4. (Probabilistic range query) *A query **range**(**Tu**, $re_3 \cup re_4$,* 5:05:25*,* 0.5*) returns Tu^1, as Tu_1^1, Tu_2^1, Tu_3^1 overlaps $re_3 \cup re_4$ at* 5:05:25 *and $\sum_{w=1}^{3} Tu_w^1.p = 1 (> 0.5)$. However, a **range**(**Tu**, RE_1,* 5:05:25*,* 0.5*) returns empty due to $Tu^1 \cap RE_1 = \emptyset$.*

5.4 Filtering and Validating Lemmas

When querying using the StIU, we can effectively avoid unnecessary decompression by exploiting p_{total} and $p_{\max}$ that are maintained for each reference.

Lemma 1. *Given a query **when**$(Tu^j, \langle (v_s \rightarrow v_e), rd \rangle, \alpha)$, if $p_{\max} < \alpha$ holds for all the tuples of reference Ref_i^j in the StIU corresponding to the region where $\langle (v_s \rightarrow v_e), rd \rangle$ is located then we do not need to fully decompress Ref_i^j.*

Proof. Let *re* be the region where $\langle (v_s \rightarrow v_e), rd \rangle$ is located, and Ω' be the subset of $Ref_i^j.Rrs$ that overlaps *re*. If $p_{\max} < \alpha$ holds for every tuple (under each time interval) associated with region *re*, it follows that $\forall Nref_{ik}^j \in \Omega'$, $Nref_{ik}^j.p < \alpha$ due to $Nref_{ik}^j.p \leq p_{\max}$. As a result, the timestamps of all non-references within Ω' will not be returned in accordance with the definition of a probabilistic when query. Therefore, Ref_i^j does not need to be fully decompressed. □

Example 5. (Filtering by Lemma 1) *Given a query **when**$(Tu^1,$ $\langle (185191 \rightarrow 185192), 0.25 \rangle, 0.5)$ in Fig. 5b, Ref_1^1 does not need to be fully decompressed. This is because $Ref_1^1.p_{\max}$ w.r.t. re_3 where $\langle (185191 \rightarrow 185192), 0.25 \rangle$ falls is* 0.2*, implying that $Nref_{1k}^1.p <$ $0.5\,(k = 1, 2)$.*

Lemma 2. *Given a spatial region RE, a timestamp t_q, and two edges $(v_s \rightarrow v_e)$ and $(v_{s'} \rightarrow v_{e'})$ where an uncertain trajectory instance Tu_i^j is located at timestamps t_b and $t_{b'}$ $(t_b \leq t_q \leq t_{b'})$, (i) if the subpath sp from v_s to $v_{e'}$ satisfies $sp \in RE$ then Tu_i^j overlaps RE at t_q; (ii) if the subpath sp from v_s to $v_{e'}$ satisfies $sp \cap RE = \emptyset$ then Tu_i^j does not overlap RE at t_q.*

Proof. Let the location where Tu_i^j is located at t_q be l. Then, l must be located on *sp* due to $t_b \leq t_q \leq t_{b'}$. Hence, Tu_i^j overlaps *RE* at t_q if $sp \in RE$; Tu_i^j does not overlap *RE* at t_q if $sp \cap RE = \emptyset$. □

Lemma 3. *Given a query **range**$(\mathbf{Tu}, RE, t_q, \alpha)$ and a set Can^j that contains the instances of $Tu^j \in$ **Tu** satisfying condition (i) in Lemma 2, if the sum of the probabilities of all the instances in Can^j is not smaller than α then Tu^j should be in the query result.*

We omit the proof of Lemma 3 as it is straightforward.

Lemma 4. *Given a query **range**$(\mathbf{Tu}, RE, t_q, \alpha)$, a region re_{total} $(RE \subseteq re_{total})$, and a set Can^j that contains all the instances of $Tu^j \in$ **Tu** that overlap re_{total} during $[t_b, t_{b'}]$ $(t_b \leq t_q \leq t_{b'})$, if the sum of probabilities of all the instances in Can^j is smaller than α then Tu^j does not qualify as a query result.*

Proof. Let Can'^j be a set of instances of Tu^j that overlaps *RE* at t_q. We have $Can'^j \subseteq Can^j$ due to $RE \subseteq re_{total}$ and $t_q \in [t_b, t_{b'}]$. As a result, the sum of probabilities of instances in Can'^j can not be greater than that in Can^j. Thus, if $\sum_{Tu_i^j \in Can^j} Tu_i^j.p < \alpha$ then Tu^j does not qualify as a query result. □

Example 6. (Filtering by Lemmas 2, 3, and 4) *Given a query **range**(**Tu**, $re_3 \cup re_4$,* 5:05:25*,* 0.5*) in Fig. 5 and $\eta_p = \frac{1}{2048}$, we can get the subpath sp_1 from v_1 to v_4 after partially decompressing $T(Tu^1)$ and $E(Ref_1^1)$, where Ref_1^1 is located on $(v_1 \rightarrow v_2)$ at* 5:03:25 *and located on $(v_3 \rightarrow v_4)$ at* 5:07:25*. According to Lemma 2, we can ensure that Ref_1^1 must overlap $re_3 \cup re_4$ at* 5:05:25 *without decompressing $D(Ref_1^1)$. Since $Ref_1^1.p \geq 0.5$, Tu^1 can be directly returned by Lemma 3. Given a query **range**(**Tu**, RE_1,* 5:05:25*,* 0.5*), we can infer that Ref_1^1, $Nref_{11}^1$ and $Nref_{12}^1$ do not overlap RE_1 without decompressing $D(Ref_1^1)$, $D(Nref_{11}^1)$ and $D(Nref_{12}^1)$ according to Lemma 2. This is because $sp_1 \cap RE_1 = \emptyset$ and $sp_2 \cap RE_1 = \emptyset$, where sp_2 is the subpath of $Nref_{11}^1$ from v_1 to v_{10}. Then, since the sum of the probabilities of instances in $Can^1 (= \emptyset)$ w.r.t. $re_{total} (= RE_1)$ is $0\,(< 0.5)$, Tu^1 can be safely pruned by Lemma 4. Consider another example **range**(**Tu**, RE_2,5:05:25, 0.8). Assume that only Ref_1^1 traversed re_1, Tu^1 can be safely pruned by Lemma 4 without checking any of its other instances, as the sum of the probabilities of instances in $Can^1 (= \{Ref_1^1\})$ w.r.t. $re_{total} (= re_1)$ is $0.75\,(< 0.8)$.*

Due to the space limitation, the detailed algorithms for probabilistic where, when, and range quires are omitted.

6. EXPERIMENTS

We report on extensive experiments aimed at evaluating the performance of the proposed framework.

6.1 Experimental Setting

Datasets. We use three real-life datasets, i.e., Denmark (DK), Chengdu (CD), and Hangzhou (HZ), as described in Table 5, while the road network information is shown in Table 6. The DK dataset is collected from 162 vehicles over about 2 years (Jan. 2007 to Dec. 2008) in Denmark. The CD dataset is collected from 14,864 taxis over one month (Aug. 2014) in Chengdu, China. The HZ

Table 5: Trajectory datasets

Datasets	Storage of NCUTs	# of trajectories	# of trajectory instances	# of edges per trajectory	Default sample interval
Denmark	0.97 GB	266,913	Average 9 (2 to 434)	Average 14 (2 to 139)	1s
Chengdu	5.00 GB	1,956,640	Average 3 (2 to 192)	Average 11 (2 to 148)	10s
Hangzhou	20.20 GB	1,807,895	Average 13 (2 to 1,500)	Average 13 (2 to 189)	20s

Table 6: Road network information

Road network	# of edges	# of vertices	Out degree
Denmark	818,020	667,950	Average 2.449
Chengdu	125,929	88,868	Average 2.834
Hangzhou	85,949	61,581	Average 2.791

Table 7: Parameter ranges and default values

Parameter	Range
Number of instances	20%, 40%, 60%, 80%, **100%**
Trajectory length	20%, 40%, 60%, 80%, **100%**
Number of pivots	**1**, **2**, 3, 4, 5
Number of grid cells	8^2, 16^2, 32^2, $\mathbf{64^2}$, 128^2
Time partition duration (min)	10, 20, 30, 40, 50, **60**
Error bound of distance (meter)	$\frac{1}{8}$, $\frac{1}{16}$, $\frac{1}{32}$, $\frac{1}{64}$, $\mathbf{\frac{1}{128}}$
Error bound of probability	$\frac{1}{128}$, $\frac{1}{256}$, $\mathbf{\frac{1}{512}}$, $\frac{1}{1024}$, $\mathbf{\frac{1}{2048}}$

dataset is collected from 24,515 taxis over one month (Nov. 2011) in Hangzhou, China.

Comparison Algorithm. As this is the first study on the compression of uncertain trajectories, we adapt the state-of-the-art work for the compression of accurate trajectories, i.e., the TED framework [40], to compress each uncertain trajectory instance while using the same to compress probability as our UTCQ. We omit bitmap compression [40], as it is time consuming and it is also applicable to UTCQ.

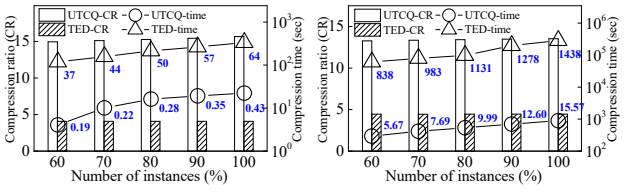
Parameter Setting. In the experiments, we study the effect on the performance of the parameters summarized in Table 7. In addition, due to the use of the PDDP-tree [40], the error bound for representing the relative distance η_D is set to $\frac{1}{128}$, while the error bound w.r.t. probability η_p is set to $\frac{1}{512}$ for the DK and CD datasets and to $\frac{1}{2048}$ for the HZ dataset. As the HZ dataset contains more instances for each uncertain trajectory, it is given a lower η_p.

Performance Metrics. For compression, we use the compression ratio, compression time, and maximum memory cost as the performance metrics. For query processing, we use the index size, query time, average difference, and F_1 score as performance metrics. All algorithms are implemented in C++ and run on a computer with Intel Core i9-9880H CPU (2.30 GHz) and 32 GB memory.

6.2 Performance of Compression

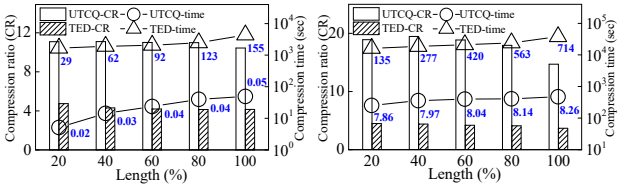
We first compare UTCQ and TED in terms of compression ratio and time. Table 8 shows the results, where T, E, D, T′, and p refer to the compression ratios of *time*, *edge*, *relative distance*, *time flag bit-string*, and *probability*, respectively, and Total denotes the total compression ratio. As observed, UTCQ outperforms TED more than 2–3 times in terms of compression ratio. The compression ratio of *time* offers evidence of the effectiveness of the SIAR scheme, while the compression ratios of *edge*, *relative distance*, and *time flag bit-string* reveal the effectiveness of referential compression. Moreover, the compression time of UTCQ is always more than 1–2 orders of magnitude smaller than that of TED, which validates the efficiency of UTCQ.

Effect of the Number of Instances. Fig. 6 shows the compression ratio and time when varying the number of trajectory instances. Specifically, we filter the trajectories with fewer than 20 instances in the datasets. As observed, the compression ratio of UTCQ improves slightly when increasing the number of instances, while that



(a) DK dataset (b) HZ dataset

Figure 6: Effect of the number of instances



(a) CD dataset (b) HZ dataset

Figure 7: Effect of the trajectory length

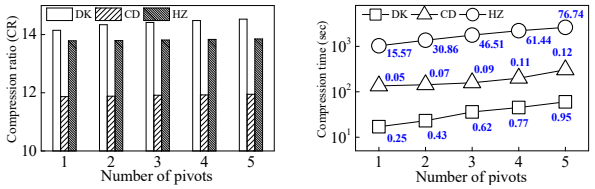
of TED is unaffected. The reason is that the more instances we have, the more can be referentially represented by UTCQ. In contrast, TED compression is independent of the number of instances. Moreover, the time of UTCQ and TED grows with the number of instances, and UTCQ is 1–2 orders of magnitude faster than TED. Finally, the digits along with the compression time are the maximum memory cost during compression. As can be seen, the maximum memory cost of TED is always 1–2 orders of magnitude higher than that of UTCQ. This is because UTCQ processes uncertain trajectories one by one, while TED has to load all the $E(\cdot)$ for the preparation of matrix transformation and partitioning [40]. In addition, the memory cost grows with the number of instance, in accordance with the space complexity.

Effect of the Trajectory Length. Fig. 7 reports the compression performance results for different lengths of trajectories. We eliminate trajectories with fewer than 20 edges and vary the trajectory length from 20% to 100% of the total number of edges. As can be seen, the compression ratios of UTCQ on both CD and HZ first increase slightly and then drop. This is because, on the one hand, the compression ratio of *time* increases with the trajectory length, while, on the other hand, the referential compression performance drops due to the larger dissimilarity among longer sequences. Moreover, the compression ratio of TED decreases slightly as the highest bits of the entry path representation in one column [40] are more unlikely to be all 0. Finally, both the compression time and maximum memory cost increase slightly with the trajectory length, and UTCQ always uses 1–3 orders of magnitude less space and 1–2 orders of magnitude less time than TED.

Effect of the Number of Pivots. Fig. 8 reports on the impact of the number of pivots on the compression performance. It can be seen that the compression ratio increases with the number of pivots used. The reason is that the more pivots, the higher the accuracy of the proposed similarity measure. On the other hand, the compression time and the maximal memory cost increase. As can be observed, the maximum memory cost on the CD dataset is the smallest, since it has the shortest and the least instances for each uncertain trajectory on average among the three datasets. Specifically, we set the default pivot count to 1 on CD and HZ datasets, while set

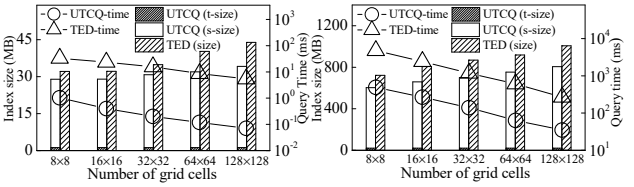
Table 8: Comparison on three datasets

Datasets	UTCQ							TED						
	Compression ratio						Time(s)	Compression ratio						Time(s)
	Total	T	E	D	T′	p		Total	T	E	D	T′	p	
Denmark	14.342	7.685	14.861	26.171	15.843	7.111	23	4.439	4.545	11.888	9.143	1	7.111	1823
Chengdu	11.867	3.128	13.589	15.141	18.061	7.111	135	4.287	1.707	11.247	9.143	1	7.111	65310
Hangzhou	13.787	3.193	16.092	17.815	14.592	5.818	1031	4.008	1.418	9.376	9.143	1	5.818	980447

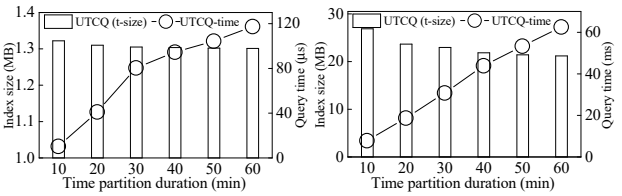


(a) Compression ratio (b) Compression time

Figure 8: Effect of pivots



(a) DK dataset (b) HZ dataset



(c) DK dataset (d) HZ dataset

Figure 9: Effect of spatial and temporal partition granularity on probabilistic range queries

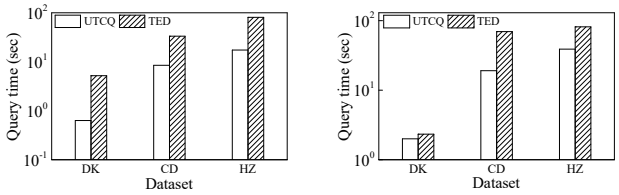
that to 2 on DK dataset. We do this because on DK dataset, the compression ratio improves significantly from 1 to 2 without reducing the efficiency considerably.

6.3 Query Performance

Probabilistic Range Query. Fig. 9 reports the performance of probabilistic range queries when varying the spatio-temporal partition granularity. Here, we omit coverage of probabilistic where and when queries, as they are largely unaffected by variations in the spatio-temporal partitioning. Fig. 9 indicates that the proposed index size is smaller than that of TED, which is due to the referential compression. It is clear that the query time decreases as road networks and time intervals are partitioned at finer granularities. Moreover, UTCQ is faster than TED, which is due to the index structure and the filtering and validating techniques.

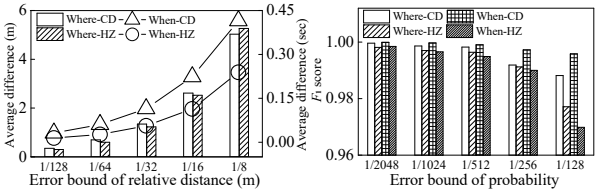
Probabilistic Where and When Queries. We also report the performance (i.e., the query time) of probabilistic where and when queries. Fig. 10 shows that UTCQ is faster than TED for both probabilistic where and when queries, due to the temporal index of StIU and Lemma 1 that are used for filtering. However, the superiority of UTCQ is not obvious on DK dataset for probabilistic when query, because the query performance (i.e., the pruning ability of Lemma 1) relies on the distribution of the dataset.

Effect of Error Bound. Fig. 11 studies the effect of the error bounds of the PDDP-tree [40] on the query accuracy, where the average difference and the $F_1 (= 2\frac{\text{precision}\cdot\text{recall}}{\text{precision}+\text{recall}})$ score are the per-



(a) Probabilistic where query (b) Probabilistic when query

Figure 10: Probabilistic where and when query performance



(a) Relative distance (b) Probability

Figure 11: Effect of error bound on query accuracy

formance metrics. Specifically, the average difference is the deviation between the query results derived from the original versus the compressed datasets. It is measured in meters (m) for probabilistic where queries and in seconds (sec) for probabilistic when queries. Fig. 11 shows that the average difference is small, especially when η_D is set to the default value; and the F_1 score is always close to 1. These indicate that the error caused by PDDP-tree encoding is small. We omit the results on range queries because they achieve similar performance.

6.4 Scalability

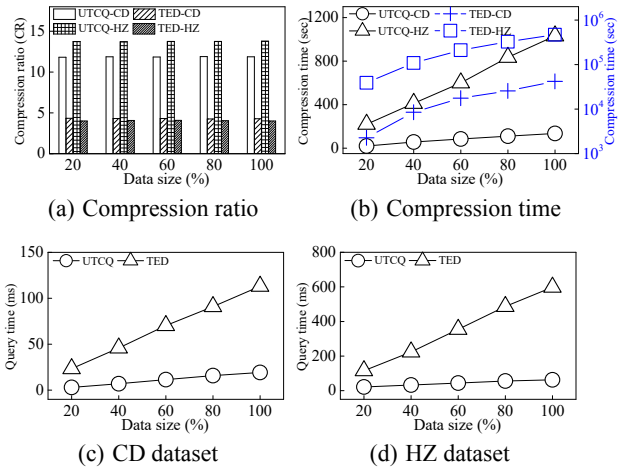
Fig. 12 reports on the scalability of compression and query processing, where the data size is varied from 20% to 100% of the total dataset storage size. Fig. 12a shows that the compression ratios achieved by both UTCQ and TED are roughly independent of the dataset size. This is because the compression ratio is unaffected by the number of uncertain trajectories, but rather depends on the number and the lengths of instances. In Fig. 12b, the compression time of UTCQ is measured by the left y axis in black color, while that of TED is measured by the right y axis in blue color. Two axes are used to show more clearly the time for both solutions that differ substantially. We see that the compression time of UTCQ increases linearly as it processes trajectories one by one, while that of TED increases super linearly due to its matrix operations. As expected, the query time of both UTCQ and TED increase linearly with the growth in the data size, as shown in Figs. 12c and 12d.

7. RELATED WORK

Trajectory compression can be classified into raw data-oriented compression and road network-embedded compression.

7.1 Raw Data-oriented Compression

Raw data-oriented compression techniques are designed to compact trajectories that have not been map-matched. A typical approach is to use trajectory simplification that approximates an orig-



(a) Compression ratio (b) Compression time

(c) CD dataset (d) HZ dataset

Figure 12: Scalability of compression and query processing

inal trajectory by a subsequence of the trajectory while attempting to minimize the information loss according to certain distance measures. The *Bellaman* [1] method uses dynamic programming to find a subsequence with the minimum spatial distance error. The *MRPA* algorithm [8] employs a distance measure called Integral Square Synchronous Euclidean Distance to simplify trajectories.To minimize the simplification error under a storage budget, Min-Error [25] is proposed. It protects the direction information of trajectories using a direction-based error measurement to detect the sharp change of directions of trajectories. In addition, trajectory simplication methods can be classified into offline methods [1, 8, 25] and online methods [4, 22, 23, 24, 29], where the offline methods require that full trajectories are available before compression starts, while the online methods can compress trajectories in streaming settings. Comprehensive experimental evaluations of trajectory simplification techniques are available [41].

Approaches based on other strategies also exist. Philippe et al. [9] propose two strategies, Single Trajectory Delta compression and Cluster-based compression. The former compresses each single trajectory by encoding the deviation between successive values. The latter clusters similar subtrajectories and only stores one summary trajectory per cluster. Wandelt and Sun [36] propose a lossless compression technique for 4D trajectories that exploits the similarities between subtrajectories by predicting the next point in a trajectory based on previous trajectories. Cai et al. [3] assume that moving objects are likely to maintain a certain mode during a period and extract this mode as a state vector based on sampling. Zhao et al. [44] construct a reference trajectory set and represent a raw trajectory as a concatenation of a series reference trajectories within a given spatio-temporal deviation threshold. More recently, Gao et al. [11] also study semantic-based compression.

The above methods do not consider the road network embedding and are not competitive in our setting.

7.2 Road Network-embedded Compression

Road network-embedded compression leverages an underlying road network to achieve better trajectory compression. GPS points are first map-matched to road segments [2, 14, 15, 26]. Since a sequence of successive points can often be mapped to, and represented by, the same segment, spatial redundancy can be reduced, which yields a higher compression ratio. Auxiliary information, such as frequent travel paths and shortest travel paths are also utilized in existing studies [13, 16, 18, 30] to improve compression. Road network-embedded compression can be classified as spatial compression or spatio-temporal compression.

Spatial compression. Krogh et al. [18] compress a trajectory by only storing the first and last edge of each shortest path in the trajectory. Koide et al. [17] present a compression technique for spatial information of trajectories and support the retrieval of subpaths. Specifically, they store path information in a Huffman-based Wavelet Tree (HWT) that counts the frequency of each label in advance. Chen et al. [5] compress trajectories by retaining out-edges with remarkable heading changes. Sui et al. [31] assign each GPS point to the middle point of a segment and propose a road-network partitioning strategy on which the compression ratio depends.

Spatio-temporal compression. Most existing studies [6, 12, 13, 16, 30] represent the temporal information of trajectories as pairs (d, t), where d is the network distance traveled at the timestamp t since the start of the trajectory. Sun et al. [13, 30] propose a two-stage spatial compression algorithms encompassing shortest path and frequent sub-trajectory compression. Ji et al. [16] encode outgoing road segments clockwise based on a pre-computed clockwise code table. Chen et al. [6] focus mainly on reducing the frequency of data transmission and adopt existing integer encoding approaches [19, 39] to compress trajectories.

The work closest to ours is TED [40]. Instead of representing (d, t) together, TED represents them individually, leading to a lossless compression in terms of t and a higher compression ratio. In addition, it provides an index structure for facilitating queries of compressed trajectories. However, TED cannot solve our problem efficiently as it is not designed to take the uncertainty of trajectories into account. In contrast, we exploit the similarities between uncertain trajectory instances to achieve high performance. To the best of our knowledge, we propose the first framework for compressing and querying uncertain trajectories. Moreover, we propose an effective index structure to support efficient queries against compressed uncertain trajectories.

8. CONCLUSION

We propose a novel framework for compressing and querying uncertain trajectories. We referentially represent uncertain trajectories by exploiting similarities between trajectory instances. To achieve this, we propose a reference selection algorithm that uses a new similarity measure, as well as several referential representation formats that make it possible to represent trajectories at a high compression ratio. As part of this, we propose an effective representation of the temporal information in trajectories that addresses fluctuations in the sampling time intervals as seen in real-life data. In addition, we propose an effective index for compressed trajectories and develop filtering techniques to accelerate probabilistic where, when, and range queries over compressed data, where *flag array* and *original array* structures are constructed to extract necessary information without full decompression. Extensive experiments conducted on three real datasets show that the UTCQ framework is 2–3 times better than the state-of-the-art method in terms of compression ratio, uses 1–3 orders of magnitude less memory, is 1–2 orders of magnitude faster in terms of compression time, and is always faster in terms of query time. In the future, it is of interest to introduce a multiple-order representation that may further improve the compression performance, and it may also be possible to develop techniques that can recover a non-reference without decompressing its reference.

ACKNOWLEDGMENTS

This work was supported by the DiCyPS project, funded by Innovation Fund Denmark. Lu Chen is the corresponding author of the work.

9. REFERENCES

[1] R. Bellman and B. Kotkin. On the approximation of curves by line segments using dynamic programming. ii. Technical report, RAND CORP SANTA MONICA CALIF, 1962.

[2] M. Bierlaire, J. Chen, and J. Newman. A probabilistic map matching method for smartphone GPS data. *TRANSPORT RES C-EMER*, 26:78–98, 2013.

[3] Z. Cai, F. Ren, J. Chen, and Z. Ding. Vector-based trajectory storage and query for intelligent transport system. *TITS*, 19(5):1508–1519, 2017.

[4] W. Cao and Y. Li. Dots: An online and near-optimal trajectory simplification algorithm. *J Syst Softw*, 126:34–44, 2017.

[5] C. Chen, Y. Ding, X. Xie, S. Zhang, Z. Wang, and L. Feng. Trajcompressor: An online map-matching-based trajectory compression framework leveraging vehicle heading direction and change. *TITS*, 2019.

[6] J. Chen, Z. Xiao, D. Wang, D. Chen, V. Havyarimana, J. Bai, and H. Chen. Toward opportunistic compression and transmission for private car trajectory data collection. *IEEE Sens. J.*, 19(5):1925–1935, 2018.

[7] L. Chen, Y. Gao, X. Li, C. S. Jensen, and G. Chen. Efficient metric indexing for similarity search. In *ICDE*, pages 591–602, 2015.

[8] M. Chen, M. Xu, and P. Franti. A fast $o(n)$ multiresolution polygonal approximation algorithm for GPS trajectory simplification. *TIP*, 21(5):2770–2785, 2012.

[9] P. Cudre-Mauroux, E. Wu, and S. Madden. Trajstore: An adaptive storage system for very large trajectory data sets. In *ICDE*, pages 109–120, 2010.

[10] S. Deorowicz and S. Grabowski. Robust relative compression of genomes with random access. *Bioinformatics*, 27(21):2979–2986, 2011.

[11] C. Gao, Y. Zhao, R. Wu, Q. Yang, and J. Shao. Semantic trajectory compression via multi-resolution synchronization-based clustering. *Knowl Based Syst*, 174:177–193, 2019.

[12] Y. Gao, B. Zheng, G. Chen, Q. Li, C. Chen, and G. Chen. Efficient mutual nearest neighbor query processing for moving object trajectories. *Inf. Sci*, 180(11):2176–2195, 2010.

[13] Y. Han, W. Sun, and B. Zheng. Compress: A comprehensive framework of trajectory compression in road networks. *TODS*, 42(2):11, 2017.

[14] G. Hu, J. Shao, F. Liu, Y. Wang, and H. Shen. If-matching: Towards accurate map-matching with information fusion. *TKDE*, 29(1):114–127, 2016.

[15] G. R. Jagadeesh and T. Srikanthan. Probabilistic map matching of sparse and noisy smartphone location data. In *ITSC*, pages 812–817, 2015.

[16] Y. Ji, Y. Zang, W. Luo, X. Zhou, Y. Ding, and L. M. Ni. Clockwise compression for trajectory data under road network constraints. In *ICBDA*, pages 472–481, 2016.

[17] S. Koide, Y. Tadokoro, C. Xiao, and Y. Ishikawa. CiNCT: Compression and retrieval for massive vehicular trajectories via relative movement labeling. In *ICDE*, pages 1097–1108, 2018.

[18] B. Krogh, C. S. Jensen, and K. Torp. Efficient in-memory indexing of network-constrained trajectories. In *SIGSPATIAL*, pages 17–26, 2016.

[19] D. Lemire and L. Boytsov. Decoding billions of integers per second through vectorization. *SOFTWARE PRACT EXPER*, 45(1):1–29, 2015.

[20] Y. Li, K. Gai, L. Qiu, M. Qiu, and H. Zhao. Intelligent cryptography approach for secure distributed big data storage in cloud computing. *Inf. Sci*, 387:103–115, 2017.

[21] Y. Li, H. Zhang, X. Liang, and B. Huang. Event-triggered-based distributed cooperative energy management for multienergy systems. *IEEE T IND INFORM*, 15(4):2008–2022, 2018.

[22] X. Lin, J. Jiang, S. Ma, Y. Zuo, and C. Hu. One-pass trajectory simplification using the synchronous Euclidean distance. *arXiv preprint arXiv:1801.05360*, 2018.

[23] J. Liu, K. Zhao, P. Sommer, S. Shang, B. Kusy, and R. Jurdak. Bounded quadrant system: Error-bounded trajectory compression on the go. In *ICDE*, pages 987–998, 2015.

[24] J. Liu, K. Zhao, P. Sommer, S. Shang, B. Kusy, J.-G. Lee, and R. Jurdak. A novel framework for online amnesic trajectory compression in resource-constrained environments. *TKDE*, 28(11):2827–2841, 2016.

[25] C. Long, R. C.-W. Wong, and H. Jagadish. Trajectory simplification: on minimizing the direction-based error. *PVLDB*, 8(1):49–60, 2014.

[26] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for low-sampling-rate gps trajectories. In *SIGSPATIAL*, pages 352–361, 2009.

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

[28] D. A. Peixoto, H. Q. V. Nguyen, B. Zheng, and X. Zhou. A framework for parallel map-matching at scale using Spark. *DISTRIB PARALLEL DAT*, 37(4):697–720, 2019.

[29] M. Potamias, K. Patroumpas, and T. Sellis. Sampling trajectory streams with spatiotemporal criteria. In *SSDBM*, pages 275–284, 2006.

[30] R. Song, W. Sun, B. Zheng, and Y. Zheng. Press: A novel framework of trajectory compression in road networks. *PVLDB*, 7(9):661–672, 2014.

[31] P. Sui and X. Yang. A privacy-preserving compression storage method for large trajectory data in road network. *J. Grid Comput.*, 16(2):229–245, 2018.

[32] J. Teuhola. A compression method for clustered bit-vectors. *INFORM PROCESS LETT*, 7(6):308–311, 1978.

[33] S. J. van Schaik and O. de Moor. A memory efficient reachability data structure through bit vector compression. In *SIGMOD*, pages 913–924, 2011.

[34] S. Wandelt and U. Leser. Adaptive efficient compression of genomes. *ALGORITHM MOL BIOL*, 7(1):30, 2012.

[35] S. Wandelt and U. Leser. Fresco: Referential compression of highly similar sequences. *TCBB*, 10(5):1275–1288, 2013.

[36] S. Wandelt and X. Sun. Efficient compression of 4D-trajectory data in air traffic management. *TITS*, 16(2):844–853, 2014.

[37] J. Wang, J. Feng, and G. Li. Trie-join: Efficient trie-based string similarity joins with edit-distance constraints. *PVLDB*, 3(1–2):1219–1230, 2010.

[38] L.-Y. Wei, Y. Zheng, and W.-C. Peng. Constructing popular routes from uncertain trajectories. In *KDD*, pages 195–203, 2012.

[39] H. Yan, S. Ding, and T. Suel. Inverted index compression and query processing with optimized document ordering. In

WWW, pages 401–410, 2009.

[40] X. Yang, B. Wang, K. Yang, C. Liu, and B. Zheng. A novel representation and compression for queries on trajectories in road networks. *TKDE*, 30(4):613–629, 2017.

[41] D. Zhang, M. Ding, D. Yang, Y. Liu, J. Fan, and H. Shen. Trajectory simplification: an experimental study and quality analysis. *PVLDB*, 11(9):934–946, 2018.

[42] H. Zhang, Y. Li, D. W. Gao, and J. Zhou. Distributed optimal energy management for energy internet. *IEEE T IND INFORM*, 13(6):3081–3097, 2017.

[43] Z. Zhang, M. Hadjieleftheriou, B. C. Ooi, and D. Srivastava. Bed-tree: an all-purpose index structure for string similarity search based on edit distance. In *SIGMOD*, pages 915–926, 2010.

[44] Y. Zhao, S. Shang, Y. Wang, B. Zheng, Q. V. H. Nguyen, and K. Zheng. Rest: A reference-based framework for spatio-temporal trajectory compression. In *KDD*, pages 2797–2806, 2018.

[45] Y. Zheng. Trajectory data mining: an overview. *TIST*, 6(3):29, 2015.