DT-KST: Distributed Top-k Similarity Query on Big Trajectory Streams

Zhigang Zhang[1], Yilin Wang[1], Jiali Mao[1], Shaojie Qiao[2], Cheqing Jin[1(✉)], and Aoying Zhou[1]

[1] School of Data Science and Engineering,
East China Normal University, Shanghai, China
{zgzhang,ylwang,jlmao1231}@stu.ecnu.edu.cn,
{cqjin,ayzhou}@sei.ecnu.edu.cn
[2] College of Information Security Engineering,
Chengdu University of Information Technology, Chengdu, China
sjqiao@cuit.edu.cn

Abstract. During the past decade, with the widespread use of smartphones and other mobile devices, big trajectory data are generated and stored in a distributed way. In this work, we focus on the distributed top-k similarity query over big trajectory streams. Processing such a distributed query is challenging due to the limited network bandwidth. To overcome this challenge, we propose a communication-saving algorithm DT-KST (Distributed Top-K Similar Trajectories). DT-KST utilizes the multi-resolution property of Haar wavelet, and devises a level-increasing communication strategy to tighten the similarity bounds. Then, a local pruning strategy is imported to reduce the amount of data returned from distributed nodes. Theoretical analysis and extensive experiments on a real dataset show that DT-KST outperforms the state-of-the-art approach in terms of communication cost.

Keywords: Top-k similarity query · Trajectory stream · Communication cost

1 Introduction

Recently, the explosive development of positioning techniques leads to the widespread of various location-acquisition devices. These devices, monitoring the motions of vehicles, people, animals, and goods, are producing massive and high-speed distributed trajectory streams. Analyzing this kind of stream data enables the understanding and forecasting of moving behaviors, and brings out novel applications and services.

In this paper, we are aiming at processing such a distributed query: "given a reference trajectory $\mathcal{Q}$, compare it against a crowd of trajectory streams stored on spatially distributed nodes, and find the top-k similar ones". $\mathcal{Q}$ can be an actual trajectory of a moving object or a virtual movement describing a desired moving pattern. This kind of query is practical in many scenarios. For example,

S. Candan et al. (Eds.): DASFAA 2017, Part I, LNCS 10177, pp. 199–214, 2017.
DOI: 10.1007/978-3-319-55753-3_13

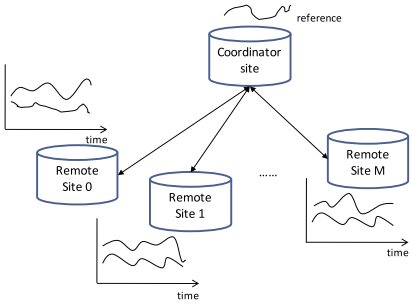


Fig. 1. Distributed processing model

video cameras are set up in many roads to capture the moving traces of vehicles continuously. The transport department is interested in finding trajectories similar to a given driving pattern such as "waving" or "swerving" to detect potential drunk drivers [2]. Another typical scenario is that, when a city manager plans to add or remove a bus route, he needs to know whether a specific route is taken by at least k passengers during 8:00–9:00. To deal with this issue, he may ask a crowd of organizations such as the bus, taxi companies and even smart phone owners to get trajectories similar to the given route [14]. In the above cases, it is inefficient to gather all distributed trajectory streams into a central site in advance, due to the limited network resources and privacy issues. Hence, centralized methods cannot be applied directly [3,8,10,11].

We firstly abstract the distributed model as a network which consists of a coordinator site and M remote sites (shown in Fig. 1). Each remote site maintains some local trajectory streams, and only communicates with the coordinator site. Query references are submitted to the coordinator to get the query results. In a naive solution, the coordinator directly transmits the query reference $\mathcal{Q}$ to all remote sites. Then, each remote site computes the similarity between $\mathcal{Q}$ and its local streams to report the k closest ones to the coordinator site. Finally, the coordinator site determines the final top-k results after receiving the candidates from all remote sites. Despite the simplicity, the communication overhead of this method is huge, because the coordinator site needs to send $\mathcal{Q}$ to all remote sites. The overhead becomes unacceptable when a lot of remote sites exist. To reduce the communication cost, multi-resolution based techniques that decompose the original data into different resolutions have been proposed [4,13]. In these works, the original data are decomposed into different resolutions, and data in a coarser resolution provide a rough outline of the original data, while those in a finer resolution disclose more details. LEEWAVE exploits the multi-resolution property of Haar wavelet that can tighten the similarity bounds gradually after iterations [13]. [4] utilizes multi-resolution property of "bounding envelope". The core idea of LEEWAVE is to compute Euclidean distance based similarity bounds for each candidate trajectory, while [4] computes a DTW distance based lower bound. Compared with [4], LEEWAVE is specially designed for processing stream data

which matches our problem. However, it suffers the following problems: (i) It aims at processing one-dimensional time series, while trajectories are essentially multi-dimensional. (ii) It collects data from remote sites to prune candidates in the coordinator, which requires much communication cost when the number of candidate trajectories or that of remote sites is large.

In this work, we show that the Haar wavelet can be used to compress trajectory data, and we can compute a similarity bound for the compressed data. Then, we propose an iterative algorithm, called DT-KST, to process the distributed top-k similarity query. DT-KST prunes the candidate trajectories in a level-increasing manner and gradually tightens the similarity bound for candidates. In comparison with LEEWAVE-CL — an improved version of LEEWAVE by adopting our tighter lower bound, DT-KST outperforms it in two aspects: (i) Only the local top-k upper bounds are required to be sent to the coordinator, while two parameters of each candidate are required in LEEWAVE-CL. (ii) In each iteration, DT-KST only sends the global k-th smallest upper bound to remote sites, while LEEWAVE-CL needs to send a list containing IDs of all candidates. The main contributions are summarized as follows:

- We show that the Haar wavelet based technology can compress the trajectory streams.
- We propose a new algorithm DT-KST to process top-k similarity query over distributed trajectory streams. DT-KST sends the coefficients of the query reference one level at a time in a top-down manner, and prunes the candidates progressively. In comparison of LEEWAVE-CL which collects information from remote sites and prunes results in the coordinator site, DT-KST prunes candidates in the remote sites directly.
- We give theoretical analysis and extensive experimental results to show that DT-KST can save more communication cost than LEEWAVE-CL.

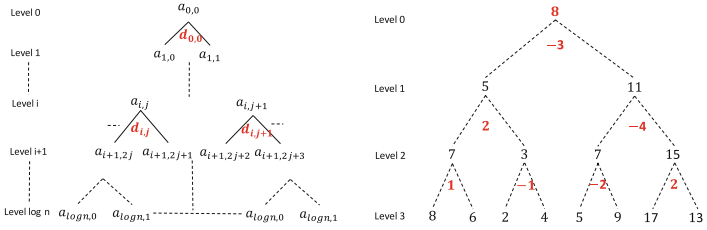
The rest of the paper is organized as follows. Section 2 discusses the related work. In Sect. 3, we define the problem formally and show that normalized Haar wavelet can be used to compress trajectory data. Furthermore, Sect. 4 proposes DT-KST algorithm and analyzes the performance in theory. Section 5 shows the experimental results. Finally, a brief conclusion is given in Sect. 6.

2 Related Work

In this section, we review recent works related to ours, including distributed top-k query on data streams and distributed trajectory similarity query.

2.1 Distributed Top-k Query on Data Streams

There exist some works on reducing the communication cost for distributed streaming top-k query. [9] proposes two schemes similar to the naive idea in Sect. 1, called CP and PRP. However, both of them suffer from heavy communication overhead due to the necessity to send the query reference to all remote



(a) Error tree (b) Example

Fig. 2. Haar wavelet transform

sites. To overcome this weakness, LEEWAVE, a level-wise approach, can tighten the similarity bound gradually by leveraging the multi-resolution property of Haar wavelet [13]. It only sends a fraction of the query reference to remote sites iteratively to approach the final result. A more compact lower bound is proposed in [5]. These two works use the Euclidean distance as the similarity metric. [4] investigates the same problem using DTW distance. As all of them aim to process one-dimensional time series, they cannot be adopted to process multi-dimensional trajectories directly.

2.2 Distributed Trajectory Similarity Query

[6,12] study how to deal with similarity join by utilizing the MapReduce framework, with the goal to reduce the data transmission between map and reduce tasks, and they are optimized for reducing the communication cost among data nodes. But in our distributed environment, remote sites only communicate with the coordinator site. In [15], each trajectory is divided into a few subsequences and these subsequences are stored in different remote sites. This work does not take communication into consideration. Communication cost is considered in Smart Trace [1] and Smart Trace$^+$ [14]. However, they only consider a special case where each remote cite (represented by a smartphone) only contains one trajectory. In contrary, we consider a more general scenario where each trajectory is fully stored in a remote site, and each site can maintain multiple trajectories. So, the aforementioned techniques [1,14,15] cannot be applied to solve our problem.

3 Preliminary

3.1 Problem Statement

We assume the time span is divided into a series of basic "time-bins" (i.e., 10 s), and each object will generate a location per time-bin. A trajectory point $\boldsymbol{p}_i$ is the location of a moving object at time-bin t_i. For each object, a trajectory stream is an infinite sequence of trajectory points, $\{\boldsymbol{p}_0, \boldsymbol{p}_1, \cdots\}$, generated at time-bins $\{t_0, t_1, \cdots\}$. To process infinite stream data, we only maintain data in

recent n time-bins in cache. The query reference $\mathcal{Q}$ and a candidate trajectory are represented as: $\mathcal{Q} = \{\boldsymbol{q}_0, \boldsymbol{q}_1, \cdots, \boldsymbol{q}_{n-1}\}$, $\mathcal{C} = \{\boldsymbol{c}_0, \boldsymbol{c}_1, \cdots, \boldsymbol{c}_{n-1}\}$ respectively. Each trajectory point is in d-dimensional space, i.e., $\boldsymbol{q}_i, \boldsymbol{c}_i \in R^d$. The Euclidean Distance between a pair of trajectory points, and that between a pair of trajectories are computed as follows:

$$ED(\boldsymbol{q}_i, \boldsymbol{c}_i) = ||\boldsymbol{q}_i - \boldsymbol{c}_i|| = \sqrt{||\boldsymbol{q}_i||^2 + ||\boldsymbol{c}_i||^2 - 2\boldsymbol{q}_i \cdot \boldsymbol{c}_i} \quad (1)$$

$$ED(\mathcal{Q}, \mathcal{C}) = \sqrt{\sum_{i=0}^{n-1} ED(\boldsymbol{q}_i, \boldsymbol{c}_i)^2} \quad (2)$$

Here, $||\boldsymbol{q}_i||$ (or $||\boldsymbol{c}_i||$) denotes the L^2-norm value of vector $\boldsymbol{q}_i$ (or $||\boldsymbol{c}_i||$), and $\boldsymbol{q}_i \cdot \boldsymbol{c}_i$ denotes the dot/inner product of two vectors $\boldsymbol{q}_i$ and $\boldsymbol{c}_i$. For simplicity, we also use Squared Euclidean Distance (SED) to measure the similarity, i.e., $SED(\boldsymbol{q}_i, \boldsymbol{c}_i) = ED(\boldsymbol{q}_i, \boldsymbol{c}_i)^2$ and $SED(\mathcal{Q}, \mathcal{C}) = ED(\mathcal{Q}, \mathcal{C})^2$. We formalize our query below.

Definition 1 (Distributed Top-k Similarity Query, DTSQ($\mathcal{Q}, TS$)). *Given a query reference $\mathcal{Q}$ and a set of all trajectory streams TS which are maintained in all remote sites, this query returns a result set S such that (i) $|S| = k$, $S \subseteq TS$, and (ii) $\forall \mathcal{C} \in S$, $\mathcal{C}' \in TS - S$, $SED(\mathcal{Q}, \mathcal{C}) \leq SED(\mathcal{Q}, \mathcal{C}')$.*

3.2 Review of Haar Wavelet

Haar wavelet is effective to compress time series [5,13]. Its transforming procedure can be regarded as a series of averaging and differencing operations at different resolutions. Working in a bottom-up manner, this procedure won't stop until the average for the whole time series is obtained. This progress can be described as an error tree, as shown in Fig. 2(a). For simplicity, an arbitrary non-leaf node has two entries, a_i^j and d_i^j, where the former is the pair-wise average of its children and the latter is the corresponding difference. The leaf nodes contain the original time series. In Fig. 2(b), we show an example of transforming a one-dimensional time series $\{8, 6, 2, 4, 5, 9, 17, 13\}$. The first pair-wise averages in the time series are $\{\frac{8+6}{2} = 7, \frac{2+4}{2} = 3, \frac{5+9}{2} = 7, \frac{17+13}{2} = 15\}$, and the corresponding differences are $\{\frac{8-6}{2} = 1, \frac{2-4}{2} = -1, \frac{5-9}{2} = -2, \frac{17-13}{2} = 2\}$. Next, based on $\{7, 3, 7, 15\}$, we obtain new averages $\{5, 11\}$ and new differences $\{2, -4\}$. Finally, the overall average $\{8\}$ and the corresponding difference $\{-3\}$ are obtained. Thus, the wavelet coefficients are $\{8, -3, 2, -4, 1, -1, -2, 2\}$, containing the overall average and all the differences. Note that the non-normalized factor, $\frac{1}{2}$, is used to compute the averages and differences in our example. In fact, $\frac{1}{\sqrt{2}}$, which is called the normalized factor, is adopted in this paper.

4 DT-KST Algorithm

4.1 Haar Wavelet for Trajectory

Trajectory is a special kind of time series in which each position is a data point in d-dimensional space. In this section, we show that the normalized Haar wavelet

can be extended to decompose trajectory data. For two trajectories $\mathcal{Q}$ and $\mathcal{C}$ of the same length n (n is a power of 2), the depth of their error trees are $L+1$, where $L = \log_2 n$, and the Haar wavelet coefficients for them are $H(\mathcal{Q}) = \{\boldsymbol{a}^{\mathcal{Q}}_{0,0}, \boldsymbol{d}^{\mathcal{Q}}_{0,0}, \boldsymbol{d}^{\mathcal{Q}}_{1,0}, \cdots, \boldsymbol{d}^{\mathcal{Q}}_{L-1,n/2-1}\}$ and $H(\mathcal{C}) = \{\boldsymbol{a}^{\mathcal{C}}_{0,0}, \boldsymbol{d}^{\mathcal{C}}_{0,0}, \boldsymbol{d}^{\mathcal{C}}_{1,0}, \cdots, \boldsymbol{d}^{\mathcal{C}}_{L-1,n/2-1}\}$, where $\boldsymbol{a}^{\mathcal{Q}}_{0,0}$ and $\boldsymbol{a}^{\mathcal{C}}_{0,0}$ are the overall average of $H(\mathcal{Q})$ and $H(\mathcal{C})$ respectively, $\boldsymbol{d}^{\mathcal{Q}}_{i,j}$ and $\boldsymbol{d}^{\mathcal{C}}_{i,j}$ are the differences, and $\boldsymbol{a}^{\mathcal{Q}}_{i,j}, \boldsymbol{a}^{\mathcal{C}}_{i,j}, \boldsymbol{d}^{\mathcal{Q}}_{i,j}, \boldsymbol{d}^{\mathcal{C}}_{i,j} \in R^d$.

According to the normalized Haar wavelet transform procedure, the pair-wise average and difference for two successive nodes of $\mathcal{Q}$'s error tree are computed as: $\boldsymbol{a}^{\mathcal{Q}}_{i,j} = \frac{\boldsymbol{a}^{\mathcal{Q}}_{i+1,2j} + \boldsymbol{a}^{\mathcal{Q}}_{i+1,2j+1}}{\sqrt{2}}, \boldsymbol{d}^{\mathcal{Q}}_{i,j} = \frac{\boldsymbol{a}^{\mathcal{Q}}_{i+1,2j} - \boldsymbol{a}^{\mathcal{Q}}_{i+1,2j+1}}{\sqrt{2}}$. Two adjacent averages in the $(i+1)$-th level of $\mathcal{Q}$'s error tree who share the same farther node are in the form of $\left\{\boldsymbol{a}^{\mathcal{Q}}_{i+1,2j}, \boldsymbol{a}^{\mathcal{Q}}_{i+1,2j+1}\right\}$. Then, the sum of pair-wise distance is:

$$
\begin{aligned}
&SED(\boldsymbol{a}^{\mathcal{Q}}_{i+1,2j}, \boldsymbol{a}^{\mathcal{C}}_{i+1,2j}) + SED(\boldsymbol{a}^{\mathcal{Q}}_{i+1,2j+1}, \boldsymbol{a}^{\mathcal{C}}_{i+1,2j+1}) \\
&= SED(\frac{\boldsymbol{a}^{\mathcal{Q}}_{i,j} + \boldsymbol{d}^{\mathcal{Q}}_{i,j}}{\sqrt{2}}, \frac{\boldsymbol{a}^{\mathcal{C}}_{i,j} + \boldsymbol{d}^{\mathcal{C}}_{i,j}}{\sqrt{2}}) + SED(\frac{\boldsymbol{a}^{\mathcal{Q}}_{i,j} - \boldsymbol{d}^{\mathcal{Q}}_{i,j}}{\sqrt{2}}, \frac{\boldsymbol{a}^{\mathcal{C}}_{i,j} - \boldsymbol{d}^{\mathcal{C}}_{i,j}}{\sqrt{2}}) \\
&= ||\frac{\boldsymbol{a}^{\mathcal{Q}}_{i,j} + \boldsymbol{d}^{\mathcal{Q}}_{i,j}}{\sqrt{2}}||^2 + ||\frac{\boldsymbol{a}^{\mathcal{C}}_{i,j} + \boldsymbol{d}^{\mathcal{C}}_{i,j}}{\sqrt{2}}||^2 - 2\frac{\boldsymbol{a}^{\mathcal{Q}}_{i,j} + \boldsymbol{d}^{\mathcal{Q}}_{i,j}}{\sqrt{2}} \cdot \frac{\boldsymbol{a}^{\mathcal{C}}_{i,j} + \boldsymbol{d}^{\mathcal{C}}_{i,j}}{\sqrt{2}} + ||\frac{\boldsymbol{a}^{\mathcal{Q}}_{i,j} - \boldsymbol{d}^{\mathcal{Q}}_{i,j}}{\sqrt{2}}||^2 \\
&+ ||\frac{\boldsymbol{a}^{\mathcal{C}}_{i,j} - \boldsymbol{d}^{\mathcal{C}}_{i,j}}{\sqrt{2}}||^2 - 2\frac{\boldsymbol{a}^{\mathcal{Q}}_{i,j} - \boldsymbol{d}^{\mathcal{Q}}_{i,j}}{\sqrt{2}} \cdot \frac{\boldsymbol{a}^{\mathcal{C}}_{i,j} - \boldsymbol{d}^{\mathcal{C}}_{i,j}}{\sqrt{2}} = SED(\boldsymbol{a}^{\mathcal{Q}}_{i,j}, \boldsymbol{a}^{\mathcal{C}}_{i,j}) + SED(\boldsymbol{d}^{\mathcal{Q}}_{i,j}, \boldsymbol{d}^{\mathcal{C}}_{i,j})
\end{aligned}
$$

Thus, we get the following theorem:

Theorem 1. *For two trajectories $\mathcal{Q}$ and $\mathcal{C}$, $SED(\mathcal{Q},\mathcal{C}) = SED(H(\mathcal{Q}), H(\mathcal{C}))$.*

Proof 1. *Let $S_i(\mathcal{Q},\mathcal{C})$ denote the sum of distances between averages at level i of $\mathcal{Q}$ and $\mathcal{C}$'s error trees, and $SED_i(\mathcal{Q},\mathcal{C})$ denote the sum of distances between differences. That is:*

$$S_i(\mathcal{Q},\mathcal{C}) = \sum\nolimits_{j=0}^{2^i-1} SED(\boldsymbol{a}^{\mathcal{Q}}_{i,j}, \boldsymbol{a}^{\mathcal{C}}_{i,j}) \quad SED_i(\mathcal{Q},\mathcal{C}) = \sum\nolimits_{j=0}^{2^i-1} SED(\boldsymbol{d}^{\mathcal{Q}}_{i,j}, \boldsymbol{d}^{\mathcal{C}}_{i,j})$$

Now, we have:

$$
\begin{aligned}
S_{i+1}(\mathcal{Q},\mathcal{C}) &= \sum\nolimits_{j=0}^{2^{i+1}-1} SED(\boldsymbol{a}^{\mathcal{Q}}_{i+1,j}, \boldsymbol{a}^{\mathcal{C}}_{i+1,j}) \\
&= \sum\nolimits_{j=0}^{2^i-1} SED(\boldsymbol{a}^{\mathcal{Q}}_{i,j}, \boldsymbol{a}^{\mathcal{C}}_{i,j}) + \sum\nolimits_{j=0}^{2^i-1} SED(\boldsymbol{d}^{\mathcal{Q}}_{i,j}, \boldsymbol{d}^{\mathcal{C}}_{i,j}) \\
&= S_i(\mathcal{Q},\mathcal{C}) + SED_i(\mathcal{Q},\mathcal{C})
\end{aligned}
$$

For the bottom level of the error trees, we have:

$$S_L(\mathcal{Q},\mathcal{C}) = S_0(\mathcal{Q},\mathcal{C}) + \sum\nolimits_{i=0}^{L-1} SED_i(\mathcal{Q},\mathcal{C}) = SED(H(\mathcal{Q}), H(\mathcal{C}))$$

As the bottom level of the error tree maintains the original data, we have $SED(\mathcal{Q},\mathcal{C}) = S_L(\mathcal{Q},\mathcal{C})$. Finally, $SED(\mathcal{Q},\mathcal{C}) = SED(H(\mathcal{Q}), H(\mathcal{C}))$ holds.

According to [13], if the length of the reference is not a power of 2, then it can be divided into a few subsequences, each with a length equal to a power of 2. In this way, the overall similarity can be computed by summing the distances. So, Haar wavelet can be used to decompose trajectories of any length.

4.2 Level-Increasing Bounds

The core idea of DT-KST is that the coordinator iteratively sends coefficients in a top-down manner and only one level is dispatched at a time. Remote sites gradually prune the candidates with more and more coefficients of $\mathcal{Q}$. To better illustrate our idea, we give the following definition:

Definition 2. *For two trajectories $\mathcal{Q}$ and $\mathcal{C}$, the accumulated distance from level 0 to level l is computed as: $accSED_l(\mathcal{Q},\mathcal{C}) = S_0(\mathcal{Q},\mathcal{C}) + \sum_{i=0}^{l} SED_i(\mathcal{Q},\mathcal{C})$.*

Then, we have the following equation:

$$\begin{aligned} SED(\mathcal{Q},\mathcal{C}) &= accSED_{L-1}(\mathcal{Q},\mathcal{C}) \\ &= accSED_l(\mathcal{Q},\mathcal{C}) + \sum\nolimits_{i=l+1}^{L-1} SED_i(\mathcal{Q},\mathcal{C}) \end{aligned} \tag{3}$$

According to Eq. 3, if only coefficients at the first few levels are received by remote sites, each site cannot determine how much those not-yet-seen coefficients at lower levels will contribute to the whole distance. So, we expand the latter part of Eq. 3 and get the following formula:

$$\sum_{i=l+1}^{L-1} SED_i(\mathcal{Q},B) = \sum_{i=l+1}^{L-1}\sum_{j=0}^{2^i-1}(||\boldsymbol{d}_{i,j}^{\mathcal{Q}}||^2 + ||\boldsymbol{d}_{i,j}^{\mathcal{C}}||^2 - 2\boldsymbol{d}_{i,j}^{\mathcal{Q}} \cdot \boldsymbol{d}_{i,j}^{\mathcal{C}}) \tag{4}$$

In Eq. 4, $\sum_{i=l+1}^{L-1}\sum_{j=0}^{2^i-1}||d_{i,j}^{\mathcal{Q}}||^2$ is computed by $SSQ - \sum_{i=0}^{l}\sum_{j=0}^{2^i-1}||\boldsymbol{d}_{i,j}^{\mathcal{Q}}||^2$ where SSQ is the sum of squared coefficients of $H(\mathcal{Q})$ and can be computed in advance by the coordinator, $\sum_{i=0}^{l}\sum_{j=0}^{2^i-1}||\boldsymbol{d}_{i,j}^{\mathcal{Q}}||^2$ can be computed according to the known coefficients received by remote sites. Similarly, the second part in Eq. 4 is computed in remote sites. The main challenge is how to compute the value of the third part of Eq. 4 without knowing coefficients below level l. In DT-KST, instead of computing the third part exactly, we compute a compact bound for it using the Cauchy-Schwarz inequality. Then, we have the following bound:

$$\begin{aligned} &-2\sqrt{\sum_{i=l+1}^{L-1}\sum_{j=0}^{2^i-1}||\boldsymbol{d}_{i,j}^{\mathcal{Q}}||^2}\cdot\sqrt{\sum_{i=l+1}^{L-1}\sum_{j=0}^{2^i-1}||\boldsymbol{d}_{i,j}^{\mathcal{C}}||^2} \\ &\le -2\sum_{i=l+1}^{L-1}\sum_{j=0}^{2^i-1}||\boldsymbol{d}_{i,j}^{\mathcal{Q}}||\cdot||\boldsymbol{d}_{i,j}^{\mathcal{C}}|| \le \sum_{i=l+1}^{L-1}\sum_{j=0}^{2^i-1} -2\boldsymbol{d}_{i,j}^{\mathcal{Q}}\cdot\boldsymbol{d}_{i,j}^{\mathcal{C}} \le 2\sum_{i=l+1}^{L-1}\sum_{j=0}^{2^i-1}||\boldsymbol{d}_{i,j}^{\mathcal{Q}}||\cdot||\boldsymbol{d}_{i,j}^{\mathcal{C}}|| \\ &\le 2\sqrt{\sum_{i=l+1}^{L-1}\sum_{j=0}^{2^i-1}||\boldsymbol{d}_{i,j}^{\mathcal{Q}}||^2}\cdot\sqrt{\sum_{i=l+1}^{L-1}\sum_{j=0}^{2^i-1}||\boldsymbol{d}_{i,j}^{\mathcal{C}}||^2} \end{aligned} \tag{5}$$

In combination of Eqs. 3, 4 and Ineq. 5, we get the following similarity bound:

$$\begin{aligned}
&\sum_{i=l+1}^{L-1}\sum_{j=0}^{2^i-1}(||\boldsymbol{d}_{i,j}^{\mathcal{Q}}||^2+||\boldsymbol{d}_{i,j}^{\mathcal{C}}||^2)-2\sqrt{\sum_{i=l+1}^{L-1}\sum_{j=0}^{2^i-1}||\boldsymbol{d}_{i,j}^{\mathcal{Q}}||^2}\cdot\sqrt{\sum_{i=l+1}^{L-1}\sum_{j=0}^{2^i-1}||\boldsymbol{d}_{i,j}^{\mathcal{C}}||^2}\\
&+accSED_l(\mathcal{Q},\mathcal{C})\le SED(\mathcal{Q},\mathcal{C})\le accSED_l(\mathcal{Q},\mathcal{C})\\
&+\sum_{i=l+1}^{L-1}\sum_{j=0}^{2^i-1}(||\boldsymbol{d}_{i,j}^{\mathcal{Q}}||^2+||\boldsymbol{d}_{i,j}^{\mathcal{C}}||^2)+2\sqrt{\sum_{i=l+1}^{L-1}\sum_{j=0}^{2^i-1}||\boldsymbol{d}_{i,j}^{\mathcal{Q}}||^2}\cdot\sqrt{\sum_{i=l+1}^{L-1}\sum_{j=0}^{2^i-1}||\boldsymbol{d}_{i,j}^{\mathcal{C}}||^2} \quad (6)
\end{aligned}$$

In Ineq. 6, we maintain both lower and upper bounds of the similarity in a level-increasing manner. Note that the same upper bound is used by LEEWAVE and DT-KST. But LEEWAVE directly uses $accSED_l(\mathcal{Q},\mathcal{C})$ as the lower bound. Obviously, our lower bound is larger than $accSED_l(\mathcal{Q},\mathcal{C})$. The correctness of DT-KST is based on the assumption that both bounds become progressively tighter. We give the proof of this assumption below.

Theorem 2 (Lower Bound Theorem). *The lower bound of a similarity range is non-decreasing when we move from level l to level $l+1$.*

Proof 2. *Let Lb_l denote the lower bound of the similarity on level l, then:*

$$\begin{aligned}
Lb_{l+1}-Lb_l=SED_{l+1}(\mathcal{Q},\mathcal{C})&-\sum\nolimits_{j=0}^{2^{l+1}-1}(||\boldsymbol{d}_{l+1,j}^{\mathcal{Q}}||^2+||\boldsymbol{d}_{l+1,j}^{\mathcal{C}}||^2)\\
&+2(\sqrt{\sum_{i=l+1}^{L-1}\sum_{j=0}^{2^i-1}||\boldsymbol{d}_{i,j}^{\mathcal{Q}}||^2}\cdot\sqrt{\sum_{i=l+1}^{L-1}\sum_{j=0}^{2^i-1}||\boldsymbol{d}_{i,j}^{\mathcal{C}}||^2}\\
&-\sqrt{\sum_{i=l+2}^{L-1}\sum_{j=0}^{2^i-1}||\boldsymbol{d}_{i,j}^{\mathcal{Q}}||^2}\cdot\sqrt{\sum_{i=l+2}^{L-1}\sum_{j=0}^{2^i-1}||\boldsymbol{d}_{i,j}^{\mathcal{C}}||^2}) \quad (7)
\end{aligned}$$

To show that the lower bound will not decrease as level goes down, we need to prove that $Lb_{l+1}-Lb_l\ge 0$. In Eq. 7, $SED_{l+1}(\mathcal{Q},\mathcal{C})$ can be substituted with $\sum_{j=0}^{2^{l+1}-1}(||\boldsymbol{d}_{l+1,j}^{\mathcal{Q}}||^2+||\boldsymbol{d}_{l+1,j}^{\mathcal{C}}||^2-2\boldsymbol{d}_{l+1,j}^{\mathcal{Q}}\cdot\boldsymbol{d}_{l+1,j}^{\mathcal{C}})$. Then our problem is transformed into proving the following inequation holds:

$$\begin{aligned}
\sum_{j=0}^{2^{l+1}-1}\boldsymbol{d}_{l+1,j}^{\mathcal{Q}}\cdot\boldsymbol{d}_{l+1,j}^{\mathcal{C}}\le&\sqrt{\sum_{i=l+1}^{L-1}\sum_{j=0}^{2^i-1}||\boldsymbol{d}_{i,j}^{\mathcal{Q}}||^2}\cdot\sqrt{\sum_{i=l+1}^{L-1}\sum_{j=0}^{2^i-1}||\boldsymbol{d}_{i,j}^{\mathcal{C}}||^2}\\
&-\sqrt{\sum_{i=l+2}^{L-1}\sum_{j=0}^{2^i-1}||\boldsymbol{d}_{i,j}^{\mathcal{Q}}||^2}\cdot\sqrt{\sum_{i=l+2}^{L-1}\sum_{j=0}^{2^i-1}||\boldsymbol{d}_{i,j}^{\mathcal{C}}||^2} \quad (8)
\end{aligned}$$

In Ineq. 5, we have $\sum_{j=0}^{2^{l+1}-1}\boldsymbol{d}_{l+1,j}^{\mathcal{Q}}\cdot\boldsymbol{d}_{l+1,j}^{\mathcal{C}}\le\sqrt{\sum_{j=0}^{2^{l+1}-1}||\boldsymbol{d}_{l+1,j}^{\mathcal{Q}}||^2}\cdot\sqrt{\sum_{j=0}^{2^{l+1}-1}||\boldsymbol{d}_{l+1,j}^{\mathcal{C}}||^2}$.

So, our target changes to prove that $\sqrt{\sum_{j=0}^{2^{l+1}-1}||\boldsymbol{d}_{l+1,j}^{\mathcal{Q}}||^2}\cdot\sqrt{\sum_{j=0}^{2^{l+1}-1}||\boldsymbol{d}_{l+1,j}^{\mathcal{C}}||^2}$

is less than the right part of Ineq. 8. For ease of expression, we set $x = \sum_{j=0}^{2^{l+1}-1} ||\boldsymbol{d}_{l+1,j}^{\mathcal{Q}}||^2$, $y = \sum_{j=0}^{2^{l+1}-1} ||\boldsymbol{d}_{l+1,j}^{\mathcal{C}}||^2$, $\alpha = \sum_{i=l+2}^{L-1} \sum_{j=0}^{2^{l}-1} ||\boldsymbol{d}_{i,j}^{\mathcal{Q}}||^2$, $\beta = \sum_{i=l+2}^{L-1} \sum_{j=0}^{2^{l}-1} ||\boldsymbol{d}_{i,j}^{\mathcal{C}}||^2$. *Ineq. 8 is transformed to:*

$$\sqrt{x \cdot y} + \sqrt{\alpha \cdot \beta} \leq \sqrt{(\alpha + x) \cdot (\beta + y)} \tag{9}$$

We square both sides of Ineq. 9 and get the following inequation:

$$2\sqrt{x \cdot y \cdot \alpha \cdot \beta} \leq \alpha \cdot y + \beta \cdot x \tag{10}$$

Ineq. 10 holds according to the arithmetic-geometric mean inequality, so does Ineq. 8.

Theorem 3 (Upper Bound Theorem). *The upper bound of a similarity range is non-increasing when we move from level* l *to level* $l + 1$.

The proof of Theorem 3 is omitted because it is similar to that of Theorem 2.

4.3 Implementation of DT-KST Algorithm

Algorithm 1 shows the level-increasing pruning work in the coordinator site. Initially, coordinator site transforms the reference $\mathcal{Q}$ using the normalized Haar wavelet and sends the sum of squared coefficients $SS\mathcal{Q}$ to all remote sites (line 1–2). Then, it runs an iterative pruning procedure (line 3–11) until the *Done* flag (initialized with $false$) is *true*. During the i-th iteration, it sends the level i coefficients to the candidate sites (line 4). Here, a candidate trajectory implies that it still has a chance to be the result. Similarly, candidate sites refer to the remote sites that contain at least one candidate trajectory. It receives local upper bounds from candidate sites, and sends the global k-th smallest one to candidate sites for pruning (line 5–6). After pruning procedure, the coordinator site recomputes the total number of candidate trajectories. If there are still more than k candidates, the iteration continues. Otherwise, DT-KST terminates the iteration and informs the candidate sites to stop running (line 9). Finally, DT-KST receives the overall top-k trajectories from remote sites and returns the result (line 12).

Algorithm 2 details the work in the remote site. Initially, the remote site extracts the coefficients for each local trajectory, and maintains the upper and lower bounds for each trajectory. The remote site stores such information in set S_r (line 2). When receiving the level i coefficients of the query reference, the remote site updates the similarity bounds for each candidate according to Ineq. 6 and sends k smallest upper bounds to the coordinator (line 6–8). After receiving the global k-th smallest upper bound, it prunes candidates and sends the number of candidates to the coordinator (line 11–12). Two cases will lead to the termination of iteration in Algorithm 2: (i) No candidate is left after pruning (line 13–15); (ii) Remote site receives the finish signal (line 4). Finally, the remote site will send the final result to the coordinator before stopping.

Algorithm 1. DT-KST in coordinator site

Input: reference trajectory $\mathcal{Q}$, k;
Output: the k most similar trajectories to $\mathcal{Q}$;
1: Extract coefficients of $\mathcal{Q}$ and get $H(\mathcal{Q})$, $SS\mathcal{Q} = \sum_{i=0}^{n-1} ||H(\mathcal{Q})_i||^2$;
2: Send $SS\mathcal{Q}$ to all remote sites;
3: **for** $i = 0$; !**Done**; $i++$ **do**
4: Send the level i coefficients of $H(\mathcal{Q})$ to all candidate sites;
5: **if** Receive local upper bounds from all candidate sites **then**
6: Sort these bounds and send the k-th smallest one, $gkub$, to candidate sites;
7: **else**
8: /* Receive $|S_r|$ from all candidate remote sites; */
9: If the total number of candidate trajectories is equal to k, send the $finish$ signal to candidate sites and set **Done** to $true$;
10: **end if**
11: **end for**
12: return the final k trajectories;

4.4 Performance Analysis

We compare DT-KST with LEEWAVE-CL (an improved version of LEEWAVE with our tighter lower bound) in terms of communication cost and time complexity. We use M to denote the number of remote sites, N to denote the number of trajectories, and n to denote the length of trajectories. Moreover, we use $|C_i|$ to refer to the number of candidate trajectories after the i-th iteration, and $|CS_i|$ to denote the number of remote sites that contain candidates.

The iterative processing strategy of LEEWAVE-CL inherits from LEEWAVE. In each iteration, it computes two summary parameters: $\sqrt{\sum_{i=l+1}^{L-1} \sum_{j=0}^{2^i-1} ||\boldsymbol{d}_{i,j}^{\mathcal{C}}||^2}$ and $accSED_l(\mathcal{Q}, \mathcal{C})$ at the remote sites for each candidate. Then, the coordinator receives the two parameters to generate a tighter bound for each candidate. Finally, it prunes candidates according to the updated similarity bounds.

Time complexity: The running time of DT-KST and LEEWAVE-CL consist of two parts: time for Haar wavelet transforming which is $O(N \cdot n)$ and the iterative pruning time. Since both of them compute in iterative way and tighten the similarity bounds in each iteration, the pruning time complexity is $O(N \cdot \log_2 n)$ for the two algorithms (the sorting time is omitted as k is usually small). As $\log_2 n \ll n$, the whole running time is dominated by the transforming time. In conclusion, the overall time complexity is $O(N \cdot n)$ for both of them.

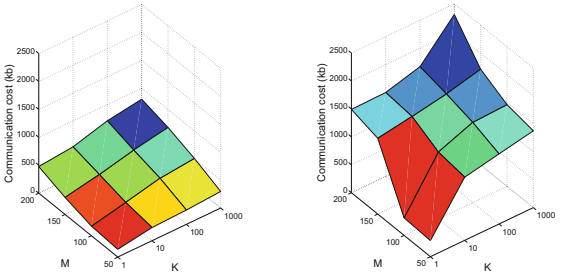
Communication cost of DT-KST: In the 0-th iteration, the coordinator sends the level 0 efficients to all M remote sites and receives at most N upper bounds from these sites, which requires a bandwidth cost of $O(d \cdot (M + N))$. In the i-th iteration ($i \geq 1$), the main cost lies in that the coordinator sends the i-th level coefficients to candidate sites, then receives at most $|C_{i-1}|$ upper bounds, which requires a communication cost of $O(d \cdot (2^i \cdot |CS_{i-1}| + |C_{i-1}|))$ bytes. So, after λ iterations, the total communication cost upper bound is $O(d \cdot (M + N + \sum_{i=1}^{\lambda-1}(2^i \cdot |CS_{i-1}| + |C_{i-1}|)))$.

Algorithm 2. DT-KST in remote site r

Input: a set of trajectory TS_r;
1: Extract coefficients for each trajectory;
2: Create a set S_r, which maintains a triple $< ID, ub, lb >$ for each trajectory;
3: Receive SSQ from the coordinator;
4: **while** The received value is not the $finish$ signal **do**
5: **if** Receive the level i coefficients of $H(\mathcal{Q})$ **then**
6: Update the similarity bounds for trajectories in S_r;
7: Sort triples in S_r according to the upper bounds;
8: Send k smallest upper bounds to coordinator;
9: **else**
10: /* Receive $gkub$; */
11: Remove the triples whose lower bounds are greater than $gkub$ from S_r;
12: Send the number of triples in S_r to the coordinator;
13: **if** $|S_r| = 0$ **then**
14: break;
15: **end if**
16: **end if**
17: **end while**
18: Send the contents of trajectories whose IDs are in S_r;

Communication cost of LEEWAVE-CL: The communication cost in the 0-th iteration is $O(d \cdot (M + 2 \cdot N))$, because the coordinator site sends the level 0 coefficients to each site and receives the two summary parameters that are computed for each trajectory stream. In the i-th iteration ($i \geq 1$), there are three steps. Firstly, the coordinator transmits the i-th level coefficients and the IDs of candidates to candidate remote sites, which leads to a communication cost of $O(|CS_{i-1}| \cdot (d \cdot 2^i + |C_{i-1}|))$. Then, each remote site sends two summary parameters for each candidate trajectory which costs $O(d \cdot |C_{i-1}|)$. Finally, the coordinator computes the global candidates C_i for next iteration. Hence the communication cost in the i-th iteration is $O(|CS_{i-1}| \cdot (d \cdot 2^i + |C_{i-1}|) + d \cdot |C_{i-1}|)$. After λ iterations, the accumulated communication cost reaches $O(d \cdot (M + 2N + \sum_{i=1}^{\lambda-1} 2^i \cdot |CS_{i-1}| + \sum_{i=1}^{\lambda-1} |C_{i-1}|) + \sum_{i=1}^{\lambda-1} |C_{i-1}| \cdot |CS_{i-1}|)$.

Discussion: We summarize our key findings regarding to the communication comparison with DT-KST and LEEWAVE-CL. Firstly, in the 0-th iteration, the communication cost of DT-KST is smaller than that of LEEWAVE-CL. Secondly, in other iterations, both of them are same in $|C_i|$ and $|CS_i|$, because they use the same bound to prune candidates. However, their communication strategies are different. In LEEWAVE-CL, each candidate sends two summary values to the coordinator firstly. Then, the coordinator sends the list of pruned results to each remote site. However, DT-KST computes the bounds for candidate trajectories in the candidate remote sites, and only sends local k smallest upper bounds to the coordinator. So, the amount of data received in the coordinator is reduced. Moreover, LEEWAVE-CL sends the list of candidate trajectories to all candidate remote sites. DT-KST only needs to send the



(a) DT-KST (b) LEEWAVE-CL

Fig. 3. Communication cost in respect of k and M

global k-th upper bound to candidate sites. So, the amount of data sent by the coordinator is also reduced in DT-KST. In summary, DT-KST saves at least $O(d \cdot (N + \sum_{i=1}^{\lambda-1} |C_{i-1}|) + \sum_{i=1}^{\lambda-1} |C_{i-1}| \cdot |CS_{i-1}|)$ communication cost than LEEWAVE-CL.

5 Experiments

5.1 Experimental Setup

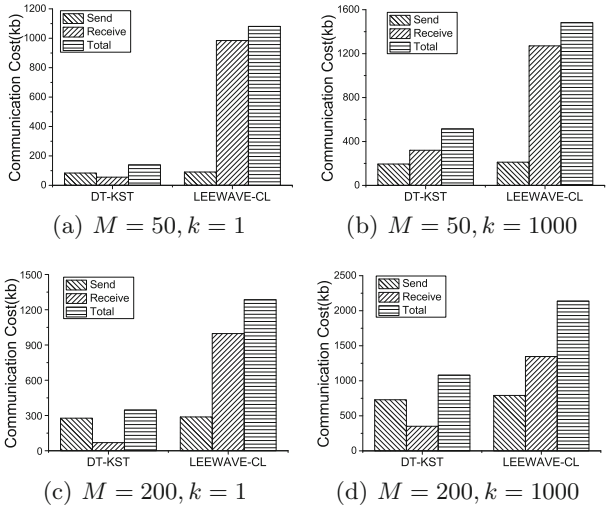
We evaluate the performance of DT-KST in this section. We compare DT-KST with LEEWAVE-CL algorithm. Both algorithms are implemented in Java, and all experiments are conducted on a Server with an 8 core Intel E5335 2.0 GHz processor and 16 GB memory.

We use a real world trajectory dataset of Beijing taxis [7] for experiment. Each trajectory point contains position (longitude and latitude), speed and other information of the taxi at the given timestamp. We select the trajectories from 1 to 7 October, 8:00 to 10:00 am and choose 10,000 longest ones for experiment. We align the length of trajectories to 10,000 using the dead-reckoning technique [16].

5.2 Results

We first examine the impact of k and M on bandwidth consumption when the length of all trajectories is set to 1,024. The value of k varies from 1 to 1,000, and M varies from 50 to 200. The results are shown in Fig. 3. For both algorithms, a larger k or M indicates the increment of communication cost. Because when k grows, both the number of candidate remote sites and that of candidate trajectories increase in each iteration. Similarly, when M increases, there are more candidate sites in each iteration. Finally, we can conclude that in comparison with LEEWAVE-CL, DT-KST is more communication-efficient in respect of k and M.

The communication cost of DT-KST and LEEWAVE-CL is determined by two factors: sending and receiving data by the coordinator site. Figure 4 evaluates



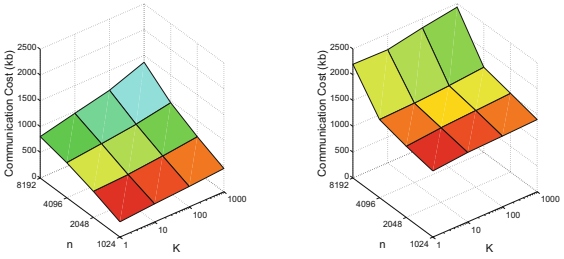
(a) $M = 50, k = 1$ (b) $M = 50, k = 1000$

(c) $M = 200, k = 1$ (d) $M = 200, k = 1000$

Fig. 4. Communication cost comparison

the performance from these two aspects. Figure 4 shows that the difference in the amount of data sent by the coordinator site is small. But, the difference in the amount of received data is obviously large. In LEEWAVE-CL, the amount of received data is much larger than that of sent data. As DT-KST only receives the upper bound of a subset of whole candidate trajectories, the amount of received data is largely reduced. Especially, when k is small (in Fig. 4(a) and (c)), the amount of data received in DT-KST is 10% less than that of LEEWAVE-CL.

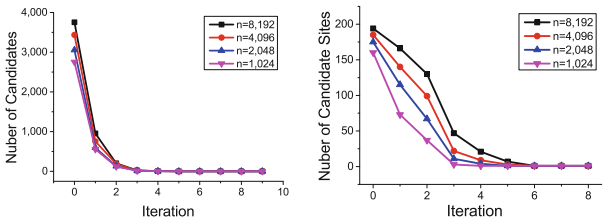
We next evaluate the impacts of n and k on bandwidth consumption when $M = 200$, as shown in Fig. 5. In this experiment, k varies from 1 to 1,000, and n varies from 1,024 to 8,192. The bandwidth consumption of both DT-KST and LEEWAVE-CL increase steadily as the length of query reference increases. This is due to that more iterations are required and more relevant coefficients



(a) DT-KST (b) LEEWAVE-CL

Fig. 5. Communication cost in respect of k and n, $M = 200$

are sent to the remote sites. LEEWAVE-CL requires more communication cost than DT-KST, mainly due to the following two reasons: (1) In the initial step, LEEWAVE-CL requires all candidates to send their bound-related information to the coordinator site which is rather costly (about 469 KB in this experiment); (2) In each iteration, DT-KST prunes candidates in each candidate remote site instead of sending them back directly. Meanwhile, as n increases from $1,024$ to $8,192$, the communication cost in DT-KST only increases to 2 times. This confirms the superiority of DT-KST for processing long trajectories.

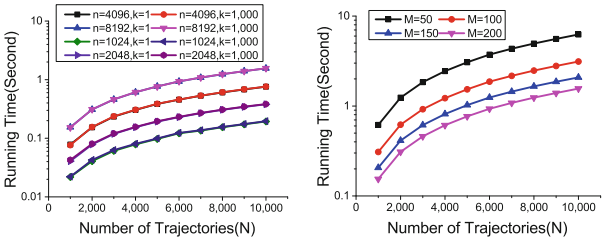


(a) (b)

Fig. 6. Pruning performance in each iteration

We then evaluate the pruning effect of DT-KST in respect of the following metrics: the number of candidates and that of candidate sites, which have been used in [13,14]. Figure 6 shows the result when n varies from 1,024 to 8,192. At the first few iterations, the number of candidate trajectories decreases exponentially (in Fig. 6(a)) and the number of candidate sites also decreases fast at the first few iterations (in Fig. 6(b)). That's because with the increment of iterations, the coefficients of the query reference obtained by remote sites also increase exponentially. More coefficients lead to tighter similarity bounds for pruning candidates. The number of candidates and that of candidate sites keep steady after the first few iterations because the top-k results have been found (after the 6-th iteration). In combination of communication cost analysis in Sect. 4.4 and Fig. 6, longer query references usually mean more communication cost.

Finally, we evaluate the time efficiency of DT-KST. Figure 7(a) shows the running time in terms of n, k and N when $M = 200$. We find that the running time increases linearly with the increment of n and N, which validates that the time complexity is proportional to n and N. That's due to the following reasons: (i) The Haar wavelet decomposition time is linear to n and N; (ii) The Haar wavelet transforming time dominates the total running time. So, although with the increase of k, the iterative pruning time increases accordingly. The overall running time is not significantly affected. Next, we evaluate the running time in respect of N and M when $k = 1, n = 8,192$. The result in Fig. 7(b) shows that for a given query, when data are distributed in more sites, the time efficiency is improved.



(a) (b)

Fig. 7. Time efficiency

6 Conclusion

In this paper, we firstly show that Haar wavelet can be used to compress multi-dimensional time series, and we can compute a similarity bound from the compressed data. Then, we present DT-KST — a communication cost-saving approach to process distributed top-k similarity query over trajectory streams. To be specific, DT-KST distributes the relevant wavelet coefficients of query reference to remote sites in a level-increasing fashion. Starting from the top level and moving down by one level at a time, DT-KST tightens the similarity bounds for candidates and prunes results accordingly. Theoretical analysis and extensive experiment results show that DT-KST saves more bandwidth than state-of-the-art algorithms.

Acknowledgement. Our research is supported by the National Key Research and Development Program of China (2016YFB1000905), NSFC (61370101, 61532021, U1501252, U1401256 and 61402180), Shanghai Knowledge Service Platform Project (No. ZF1213).

References

1. Costa, C., Laoudias, C., Zeinalipour-Yazti, D., Gunopulos, D.: SmartTrace: finding similar trajectories in smartphone networks without disclosing the traces. In: Proceedings of the 27th ICDE, pp. 1288–1291 (2011)
2. Dai, J., Teng, J., Bai, X., Shen, Z., Xuan, D.: Mobile phone based drunk driving detection. In: Proceedings of the 2010 International Conference on Pervasive Computing Technologies for Healthcare, pp. 1–8. IEEE (2010)
3. Ding, H., Trajcevski, G., Scheuermann, P.: Efficient similarity join of large sets of moving object trajectories. In: Proceedings of the 15th TIME, pp. 79–87. IEEE (2008)
4. Hsu, C.C., Kung, P.H., Yeh, M.Y., Lin, S.D., Gibbons, P.B.: Bandwidth-efficient distributed k-nearest-neighbor search with dynamic time warping. In: Proceedings of the 2015 IEEE ICBD, pp. 551–560. IEEE (2015)
5. Kashyap, S., Karras, P.: Scalable kNN search on vertically stored time series. In: 2011 Proceedings of the 17th ACM SIGKDD, pp. 1334–1342 (2011)

6. Kim, Y., Shim, K.: Parallel top-k similarity join algorithms using MapReduce. In: Proceedings of the IEEE 28th ICDE, pp. 510–521. IEEE (2012)
7. Liu, H., Jin, C., Zhou, A.: Popular route planning with travel cost estimation. In: Navathe, S.B., Wu, W., Shekhar, S., Du, X., Wang, X.S., Xiong, H. (eds.) DASFAA 2016. LNCS, vol. 9643, pp. 403–418. Springer, Heidelberg (2016). doi:10.1007/978-3-319-32049-6_25
8. Ma, C., Lu, H., Shou, L., Chen, G.: KSQ: top-k similarity query on uncertain trajectories. TKDE **25**(9), 2049–2062 (2013)
9. Papadopoulos, A.N., Manolopoulos, Y.: Distributed processing of similarity queries. Distrib. Parallel Databases **9**(1), 67–92 (2001)
10. Sacharidis, D., Skoutas, D., Skoumas, G.: Continuous monitoring of nearest trajectories. In: Proceedings of the 22nd ACM SIGSPATIAL, pp. 361–370. ACM (2014)
11. Skoumas, G., Skoutas, D., Vlachaki, A.: Efficient identification and approximation of k-nearest moving neighbors. In: Proceedings of the 21st ACM SIGSPATIAL, pp. 264–273. ACM (2013)
12. Vernica, R., Carey, M.J., Li, C.: Efficient parallel set-similarity joins using MapReduce. In: Proceedings of the 16th ACM SIGMOD, pp. 495–506. ACM (2010)
13. Yeh, M.Y., Wu, K.L., Yu, P.S., Chen, M.S.: LeeWave: level-wise distribution of wavelet coefficients for processing kNN queries over distributed streams. PVLDB **1**(1), 586–597 (2008)
14. Zeinalipour-Yazti, D., Laoudias, C., Costa, C., Vlachos, M., Andreou, M.I., Gunopulos, D.: Crowdsourced trace similarity with smartphones. TKDE **25**(6), 1240–1253 (2013)
15. Zeinalipour-Yazti, D., Lin, S., Gunopulos, D.: Distributed spatio-temporal similarity search. In: Proceedings of the 2006 CIKM, pp. 14–23 (2006)
16. Zheng, Y., Zhou, X.: Computing with Spatial Trajectories. Springer, New York (2011)