Fast Error-Bounded Distance Distribution Computation

Jiahao Zhang, Man Lung Yiu, Bo Tang, Qing Li

Abstract—In this work we study the distance distribution computation problem. It has been widely used in many real-world applications, e.g., human genome clustering, cosmological model analysis, and parameter tuning. The straightforward solution for the exact distance distribution computation problem is unacceptably slow due to (i) massive data size, and (ii) expensive distance computation. In this paper, we propose a novel method to compute approximate distance distributions with error bound guarantees. Furthermore, our method is generic to different distance measures. We conduct extensive experimental studies on three widely used distance measures with real-world datasets. The experimental results demonstrate that our proposed method outperforms the sampling-based solution (without error guarantees) by up to three orders of magnitude.

Index Terms—cumulative distance distribution, error-bound guaranteed approximation, lower and upper bounds

1 Introduction

In recent years, more and more datasets become available in many research areas (e.g., trajectory data [1], scientific celestial data [2], points-of-interests data [3]), making it possible to apply data-driven exploratory analysis in various applications. In this work, we focus on computing the distance distribution in a dataset, which can be used in dozens of exploratory analysis applications. We study cumulative distance distribution and distance distribution histogram in this paper. Specifically, a given dataset $\mathcal{D}$ is converted to a collection of distances among objects. Those distances are sorted, then plotted as the cumulative distance distribution or distance distribution histogram. Data analysts may exploit the distance distribution among data objects to reveal the characteristics/insights of a dataset. We introduce four applications of distance distributions in different areas below.

Human genome clustering: Biologists examine the cumulative distance distribution of human genes to explore the characteristics of different gene patterns [4]. Specifically, the cumulative distance distribution of genes provides clear and intuitive insights for human genome clustering, aggregation degree, etc. For example, in Figure 1(a), snRNAs (a class of human microgenes) have several rapid ascents in its cumulative distance distribution (the blue curve). Hence, these distance values at rapid ascents may be used as cluster radii in clustering algorithms.

Cosmological model analysis: Given billions of celestial objects, astronomy scientists could plot the distance distribution histogram of those celestial objects in order to analyze cosmological models [5]. Figure 1(b) is an exact distance distribution histogram among 4.5 billion celestial objects, which takes hundreds of hours of computation. This statistical function can be applied to classify unknown celestial objects into different systems (e.g., inner solar system, outer solar system), and assist astronomy scientists in identifying relationships among galaxies.

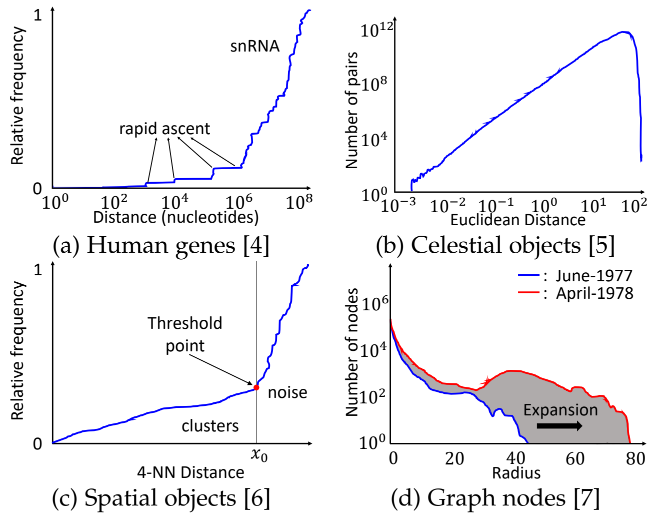


Fig. 1. Distance distribution applications, (a) and (c) are cumulative distance distributions, (b) and (d) are distance distribution histograms

Parameter tuning: Data mining algorithms may utilize cumulative distance distributions to determine parameter values, instead of trial-and-error. DBSCAN [6], a well-known clustering algorithm in spatial databases, adopts the cumulative distribution of k-NN distance to determine the parameter ϵ, the radius of a neighborhood with respect to some points. Consider the example in Figure 1(c), ϵ can be set as the x-axis value (x_0) of the "transition" point (see the red dot) in the cumulative distribution of 4-NN distance.

Graph analysis: The distance distribution histogram be-

- *Jiahao Zhang, Man Lung Yiu and Qing Li are with the Department of Computing, The Hong Kong Polytechnic University. E-mail: {csjhzhang, csmlyiu}@comp.polyu.edu.hk, qing-prof.li@polyu.edu.hk*
- *Bo Tang is with Research Institute of Trustworthy Autonomous Systems, Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering in Southern University of Science and Technology, and PCL Research Center of Networks and Communications in Peng Cheng Laboratory. He is the corresponding author. E-mail: tangb3@sustech.edu.cn*

tween node pairs from large graphs is an effective tool to discover fascinating properties (e.g., temporal variation tendency and central nodes) [7]. Figure 1(d) shows two distance distribution histograms of the graph dataset YahooWeb [8] in June-1977 and April-1978, respectively. The radius (x-axis) means the distance between one node and its farthest reachable node. The gray region shows that YahooWeb experienced a significant expansion from June-1977 to April-1978.

In addition to the above applications, distance distributions are also adopted to capture the characteristics of high dimensional datasets, e.g., building indexes for metric spaces [9], estimating CPU and I/O cost models for M-tree [10].

Given a set of object pairs $\mathcal{D}_{pair}$, its distance distribution histogram can be derived from its cumulative distance distribution easily. Thus, we focus on computing the cumulative distance distribution problem first, then discuss how to derive its corresponding distance distribution histogram efficiently. A naive solution for the cumulative distance distribution problem is: (i) compute the distance between every pair of objects, then (ii) sort those distances and plot the cumulative distance distribution. The time complexity of the naive solution is $O(|\mathcal{D}_{pair}| \cdot cost_{dist} + |\mathcal{D}_{pair}| \cdot \log |\mathcal{D}_{pair}|)$, where $cost_{dist}$ is the time complexity of computing the distance of an object pair.

Obviously, the naive solution is unacceptably slow due to (i) massive data size and (ii) expensive distance computation. We observe that the distance computation dominates the computation time. Distance measures like dynamic time warping (DTW) and discrete Fréchet distance (DFD), require $O(m^2)$ computation time, where m denotes the number of attributes per data object. For example, the OSM-FULL [11] dataset contains 2.4 million trajectories. The top-2 longest trajectories in the dataset OSM-FULL contain 960,160 and 733,523 GPS points, respectively. It takes 2484.81 seconds to compute the discrete Fréchet distance (DFD) between these two trajectories. Moreover, given a random selected trajectory in OSM-FULL, it takes 8.68 hours to compute its exact distance distribution with DFD as the distance measure.

In the database community, approximate query processing techniques [12] are widely used for answering queries approximately over voluminous data. Specifically, such techniques help data scientists identify insights from data that needs further exploration, enabling visualization, and supporting interactive exploratory analysis. For applications that cost too much time to compute the distance distributions, the approximate approach can be adopted to accelerate them with acceptable results. Besides, subject to time limit, some existing applications have to generate distance distribution offline or use synthetic distributions (e.g., random distributions) to simulate and predict results. Approximate distance distributions enable them to update distance distribution interactively and produce distance distributions for real datasets. To support interactive and scalable data analysis applications, this paper aims at computing an approximate distance distribution quickly with worst-case error guarantees.

Sampling [13], [14], [15] is the gold standard of approximate query processing in the world of big data. For example, the SQL:2008 standard allows user to choose the sampling fraction. The sampling idea can be adapted to compute the approximate cumulative distance distribution. Specifically, it first picks the objects uniformly-at-random with the given sampling fraction, then computes the distances (and distance distribution) on the sampled dataset and scale-up the distribution. Unfortunately, it does not provide any worst-case guarantee on the error of approximate results.

The challenge of our problem is how to offer worst-case error guarantees on approximate results (without computing the exact results). In this work, we propose a novel computation framework that renders fast approximate result computation with worst-case error guarantees. Moreover, our proposal is generic, i.e., it can be used with different distance measures. The major contributions of this paper are summarized as follows.

- To our knowledge, this is the first work that studies the approximate distance distribution computation problem with error bound guarantees. We formally define the approximate cumulative distance distribution problem and the approximate distance distribution histogram problem in Sections 2 and 5, respectively.
- We design a computation framework for the error-bounded cumulative distance distribution computation problem and devise a suite of optimization techniques in Sections 3 and 4, respectively.
- We extend the solution in Sections 3 and 4 to compute error-bounded distance distribution histogram efficiently in Section 5.
- We evaluate the efficiency and effectiveness of our proposal on three real datasets with widely used distance measures in Section 6. Comparing with the sampling-based solution, our solution not only provides worst-case error guarantees but also is up to three orders of magnitude faster.

The rest of this paper is organized as follows. We define the approximate cumulative distance distribution computation problem in Section 2. Our solution framework is presented in Section 3, together with a suite of performance optimization techniques in Section 4. In Section 5, we extend our proposed techniques to solve the error-bounded distance distribution histogram computation problem. We evaluate the effectiveness and efficiency of our proposal on real datasets with various distance measures in Section 6. Finally, we review the related work in Section 7 and conclude the paper in Section 8.

2 Problem Definition

We first define our research problem in Section 2.1, then present an existing sampling-based solution in Section 2.2.

2.1 Problem Formulation

In data mining, similarity or distance measures are used to quantify how much alike two data objects are. Let $dist(o_i, o_j)$ denote the distance between two data objects o_i and o_j. It is expensive to compute distance functions for complex data objects (e.g., time series, trajectories, image color histograms).

TABLE 1
Distance measures for complex data objects

Data type	Distance measure	Cost
Time series	Euclidean distance (ED)	$O(m)$
Time series	dynamic time warping (DTW)	$O(m^2)$
Trajectories	discrete Fréchet distance (DFD)	$O(m^2)$

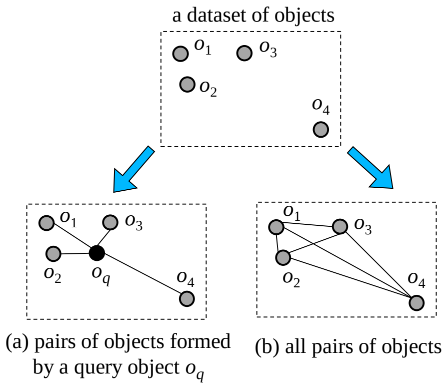


Fig. 2. The set $\mathcal{D}_{pair}$ of object pairs

Table 1 shows three representative distance measures and computation costs for such data objects. Observe that computation costs grow linearly or super-linearly with m, the number of attributes per object. We are aware that Agarwal et al. [16] proposed an algorithm to compute the exact DFD with $O(m^2 \frac{\log \log m}{\log m})$ cost recently, which is slightly lower than $O(m^2)$ in Table 1. The reasons why we do not apply it in this work are as follows: (i) the development of faster exact distance algorithms is orthogonal to our work; they can be directly applied in our proposed framework, and (ii) the algorithm in [16] achieves $O(m^2 \frac{\log \log m}{\log m})$ cost theoretically, however it is hard to implement in practice.

Given a dataset $\mathcal{D}$ of objects, the user could formulate a set $\mathcal{D}_{pair}$ of object pairs and then study the distance distribution of object pairs in $\mathcal{D}_{pair}$. Figure 2 illustrates two examples for formulating a set $\mathcal{D}_{pair}$ of object pairs.

- **By a query object (e.g., human genome clustering).** In Figure 2(a), we take a query object o_q and form the set $\mathcal{D}_{pair}$ as
$$\mathcal{D}_{pair} = \{(o_q, o_i) : o_i \in \mathcal{D}\}.$$
- **By correlation (e.g., cosmological model analysis).** In Figure 2(b), we form the set $\mathcal{D}_{pair}$ by using all pairs of objects from $\mathcal{D}$, i.e.,
$$\mathcal{D}_{pair} = \{(o_i, o_j) : o_i \in \mathcal{D}, o_j \in \mathcal{D}, o_i \neq o_j\}.$$

With the above concepts, we define a generic problem for cumulative distance distribution as follows.

Problem 1 (Cumulative Distance Distribution). *Given a set $\mathcal{D}_{pair}$ of object pairs and a distance measure dist, the cumulative distance distribution (CDD) problem returns a function of distance d, i.e., $F : \mathbb{R}^+ \to \mathbb{N}$ such that*

$$F(d) = |\{(o_i, o_j) \in \mathcal{D}_{pair} : dist(o_i, o_j) \leq d\}|.$$

This problem is generic with respect to (i) distance function, and (ii) the way of formulating the set $\mathcal{D}_{pair}$.

Observe that the function $F(d)$ is a piecewise step function because it is defined by finite pairs in $\mathcal{D}_{pair}$. Hence, it requires the range and the function value of every interval.

A straightforward method for this problem is shown in Algorithm 1. First, we compute the distance $dist(o_i, o_j)$ for each object pair $(o_i, o_j) \in \mathcal{D}_{pair}$, and insert it into a distance array $\mathcal{L}$. We then sort the distance values in $\mathcal{L}$, and compute the cumulative distance distribution $F(d)$. Specifically, given the sorted array $\mathcal{L}$ which includes the distance values of all pairs, the cumulative distance distribution $F(d)$ is expressed as follows.

$$F(d) = \begin{cases} 0, & d \in [0, \mathcal{L}[1]) \\ 1, & d \in [\mathcal{L}[1], \mathcal{L}[2]) \\ ... & \\ n, & d \in [\mathcal{L}[n], +\infty) \end{cases} \tag{1}$$

Algorithm 1 CDD($\mathcal{D}_{pair}$)

```
1: initialize distance value array $\mathcal{L}$
2: for each object pair $(o_i, o_j) \in \mathcal{D}_{pair}$ do
3:     append $dist(o_i, o_j)$ to $\mathcal{L}$
4: sort distance values in $\mathcal{L}$ by ascending order
5: compute $F(d)$ with $\mathcal{L}$ by using Equation (1)
6: return $F(d)$
```

The cost of Algorithm 1 consists of: (i) computing all distance values, i.e., $O(|\mathcal{D}_{pair}| \cdot cost_{dist})$, where $cost_{dist}$ denotes the cost per distance computation, (ii) sorting all distance values, i.e., $O(|\mathcal{D}_{pair}| \cdot \log |\mathcal{D}_{pair}|)$, and (iii) computing cumulative distance distribution $F(d)$, it is linear to $O(|\mathcal{D}_{pair}|)$. In total, this method takes $O(|\mathcal{D}_{pair}| \cdot cost_{dist} + |\mathcal{D}_{pair}| \cdot \log |\mathcal{D}_{pair}|)$ time. Clearly, this method is not scalable when $\mathcal{D}_{pair}$ has a massive size or distance computation $cost_{dist}$ is expensive, as discussed before.

To enable interactive exploratory analysis and visualization on cumulative distance distribution, we propose to compute a function $\widetilde{F}(d)$ quickly such that it approximates to $F(d)$ with worst-case error guarantee. We define this approximate version of Problem 1 as follows:

Problem 2 (Error-bounded Approximate Cumulative Distance Distribution). *Given a set $\mathcal{D}_{pair}$ of object pairs, a distance measure dist and an error threshold δ, the approximate cumulative distance distribution (ACDD) returns a function of distance d, i.e., $\widetilde{F} : \mathbb{R}^+ \to \mathbb{N}$ such that the condition*

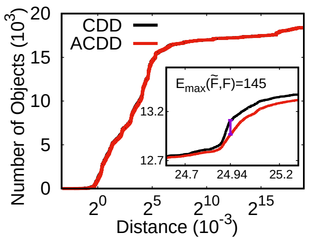
$$|\widetilde{F}(d) - F(d)| \leq \delta$$

holds for every value of d.

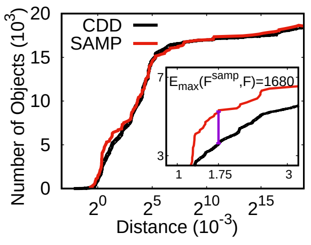
Example: In this example, we use a trajectory dataset (GeoLife [17]) with DFD as the distance measure. Figure 3(a) shows the exact (black) and approximate (red) cumulative distance distribution. For the approximate curve in Figure 3(a), the error threshold δ is fixed to 1% of the size of $\mathcal{D}_{pair}$.

2.2 Sampling-based Solution

Sampling is a common approach to deal with massive datasets [15]. In this section, we present a sampling-based



(a) Error-bounded solution ($\delta = |\mathcal{D}_{pair}| \cdot 1\%$)



(b) Sampling-based Solution ($\alpha = 1\%$)

Fig. 3. Exact and approximate cumulative distance distributions on GeoLife with DFD

method, which can be used to address our problem partially, i.e., it provides approximate cumulative distance distributions but without worst-case error guarantees. Specifically, the sampling-based solution picks a subset $\mathcal{D}_{pair}^{samp}$ from $\mathcal{D}_{pair}$ based on a given sampling fraction α. Next, it applies the straightforward method (see Algorithm 1) to compute the cumulative distance distribution on $\mathcal{D}_{pair}^{samp}$, and then scale-up the distribution by the factor $\frac{1}{\alpha}$ to obtain the approximate cumulative distance distribution $F^{samp}(d)$.

The sampling-based solution allows users to control response time via the sampling fraction α. However, it cannot provide worst-case error guarantee on the approximate cumulative distance distribution. In order to quantify the result qualities of the sampling-based solution, we define the maximum error between a given approximate cumulative distance distribution function $G(d)$ and exact cumulative distance distribution function $F(d)$ as follows.

Definition 1 (Maximum Error). *The maximum error between a given approximate cumulative distance distribution function $G(d)$ and exact cumulative distance distribution function $F(d)$ is defined as*

$$E_{max}(G,F) = \max_{0 \le d < \infty} |G(d) - F(d)|$$

Example: Take the approximate cumulative distance distribution function $\widetilde{F}(d)$ as an example, we have $E_{max}(\widetilde{F},F) = \max_{0 \le d < \infty} |\widetilde{F}(d) - F(d)| \le \delta$. Thus, the maximum error between $\widetilde{F}(d)$ and $F(d)$ does not exceed δ, e.g., the maximum error is 145 in Figure 3(a), which is less than the given $\delta = 1\% \cdot |\mathcal{D}_{pair}| = 186$. Consider the approximate cumulative distance function $F^{samp}(d)$ in Figure 3(b), which is returned by the sampling-based solution (SAMP) with sampling-ratio $\alpha = 1\%$. The maximum error between $F^{samp}(d)$ and $F(d)$ is 1680.

Observe that $\widetilde{F}(d)$ is better than $F^{samp}(d)$ to approximate $F(d)$ on GeoLife with DFD (see Figure 3) because $E_{max}(\widetilde{F},F) = 145 < E_{max}(F^{samp},F) = 1680$.

3 Our Proposed Method

In this section, we propose a novel framework to compute the approximate cumulative distance distribution $\widetilde{F}(d)$ with error-bounded guarantees (see Problem 2). Our idea is to develop lower and upper bound functions for the exact cumulative distance distribution $F(d)$, as defined below:

Definition 2 (Distribution Bound Functions). *The functions $F_{\downarrow}(d)$ and $F^{\uparrow}(d)$ are said to be the lower bound and the upper bound of $F(d)$, respectively, if*

$$F_{\downarrow}(d) \le F(d) \le F^{\uparrow}(d)$$

holds for every value $d \in [0, +\infty)$.

The above concept enables our proposed framework as illustrated in Figure 4. First, we compute the bound functions $F_{\downarrow}(d)$ and $F^{\uparrow}(d)$. An intuitive way is to take the approximate distance distribution as:

$$\widetilde{F}(d) = \lambda F^{\uparrow}(d) + (1-\lambda)F_{\downarrow}(d), \quad \lambda \in [0,1] \tag{2}$$

As we will prove in Section 3.2, Equation (2) offers worst-case error guarantees on the approximate result without computing the exact result $F(d)$ when $\lambda = \frac{1}{2}$. We next verify whether $\widetilde{F}(d) = \frac{F^{\uparrow}(d)+F_{\downarrow}(d)}{2}$ satisfies the error requirement. If the verification is not passed, then we tighten the bound functions $F_{\downarrow}(d)$ and $F^{\uparrow}(d)$. This process repeats until the verification can be passed.

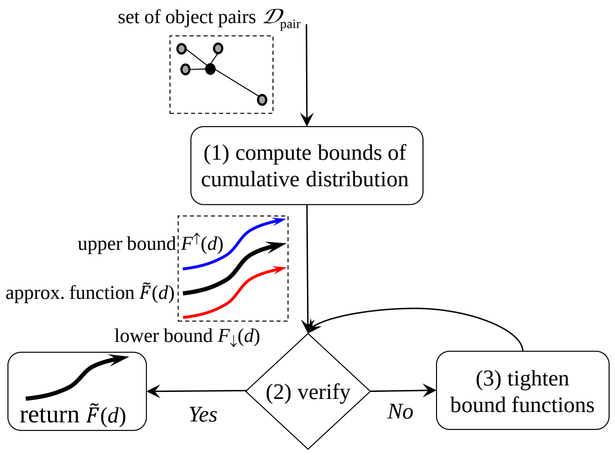

set of object pairs $\mathcal{D}_{pair}$
(1) compute bounds of cumulative distribution
upper bound $F^{\uparrow}(d)$
approx. function $\tilde{F}(d)$
lower bound $F_{\downarrow}(d)$
return $\tilde{F}(d)$
Yes
(2) verify
No
(3) tighten bound functions

Fig. 4. Our proposed solution framework

```
Algorithm 2 ACDD($\mathcal{D}_{pair}$, $\delta$)
1: compute $F_{\downarrow}$ and $F^{\uparrow}$                          ▷ Section 3.1
2: while Verify($F_{\downarrow}, F^{\uparrow}, \delta$) = False do     ▷ Section 3.2
3:     tighten $F_{\downarrow}$ and $F^{\uparrow}$                     ▷ Section 3.3
4: return $\widetilde{F}(d) = \frac{F_{\downarrow}(d)+F^{\uparrow}(d)}{2}$
```

The pseudocode of our proposed solution is summarized in Algorithm 2. Our research questions are as follows:

- How to compute the bound functions $F_{\downarrow}(d)$ and $F^{\uparrow}(d)$ efficiently? (Section 3.1)
- How to verify whether the approximate distance distribution $\widetilde{F}(d)$ satisfies the error requirement without computing the exact result $F(d)$? (Section 3.2)
- How to tighten those bound functions? (Section 3.3)

3.1 $F^{\uparrow}(d)$ and $F_{\downarrow}(d)$ Computation

Recall that the cumulative distance distribution $F(d)$ involves many calls to distance computation, e.g., DTW(o_i, o_j), which are expensive as shown in Table 1. To avoid expensive distance computation, we leverage bounding functions for distance computation to construct $F_{\downarrow}(d)$ and $F^{\uparrow}(d)$ efficiently. We first introduce the concept of distance bound functions.

Definition 3 (Distance Bound Functions)**.** *$\forall o_i \in \mathcal{D}, \forall o_j \in \mathcal{D}$, $LB_{dist}(o_i, o_j)$ and $UB_{dist}(o_i, o_j)$ are said to be the lower bound and upper bound functions of $dist(o_i, o_j)$, respectively. It holds that*

$$LB_{dist}(o_i, o_j) \le dist(o_i, o_j) \le UB_{dist}(o_i, o_j).$$

For instance, according to the literature, the representative distance lower and upper bounds for several distance measures (e.g., ED, DTW, DFD) are listed in Table 2. These bound functions incur lower computation cost than exact distance measures. For example, exact DFD computation takes $O(m^2)$ time but the lower bound $LB_{cell}(o_i, o_j)$ takes only $O(1)$ time [18]. For the sake of presentation, we omit the exact definition of these distance bound functions and refer interested readers to the corresponding references shown in Table 2.

TABLE 2
Bounds for distance measures

Distance	lower bounds	upper bounds
ED	LB_{PAA} [19], LB_{FNN} [20] $\cdots$	UB_{PAA} [19], UB_{FNN} [20] $\cdots$
DTW	LB_{KimFL} [21], LB^{EQ}_{Keogh} [22] $\cdots$	UB_{Keogh} [22], UB_{ED} [23] $\cdots$
DFD	LB_{cell} [18], LB_{row} [18] $\cdots$	UB_{g} [18], UB_{greedy} [24] $\cdots$

We next exploit distance bound functions, $LB_{dist}(o_q, o_i)$ and $UB_{dist}(o_q, o_i)$, to construct the bound functions $F_{\downarrow}(d)$ and $F^{\uparrow}(d)$ for cumulative distance distribution $F(d)$, as described by the following lemma.

Lemma 1. *Define the following functions:*

$$F_{\downarrow}(d) = |\{(o_q, o_i) \in \mathcal{D}_{pair} : UB_{dist}(o_q, o_i) \le d\}|$$
$$F^{\uparrow}(d) = |\{(o_q, o_i) \in \mathcal{D}_{pair} : LB_{dist}(o_q, o_i) \le d\}|$$

$\forall d \in [0, +\infty)$, it holds

$$F_{\downarrow}(d) \le F(d) \le F^{\uparrow}(d).$$

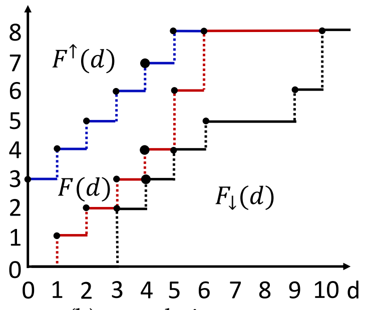
Proof. We first prove that $\forall d \in [0, +\infty), F_{\downarrow}(d) \le F(d)$. Given a distance value d_0, suppose $S_{\downarrow} = \{(o_q, o_i) \in \mathcal{D}_{pair} : UB_{dist}(o_q, o_i) \le d_0\}$ and $S = \{(o_q, o_i) \in \mathcal{D}_{pair} : dist(o_q, o_i) \le d_0\}$. We then have $S_{\downarrow} \subseteq S$ as $dist(o_q, o_i) \le UB_{dist}(o_q, o_i)$. Hence, $\forall d \in [0, +\infty), F_{\downarrow}(d) = |S_{\downarrow}| \le |S| = F(d)$. Similarly, we can prove $\forall d \in [0, +\infty), F(d) \le F^{\uparrow}(d)$. $\square$

It is worth to note that, like the function $F(d)$, both $F_{\downarrow}(d)$ and $F^{\uparrow}(d)$ can be expressed as piecewise step functions. Moreover, it is cheap to derive $F_{\downarrow}(d)$ and $F^{\uparrow}(d)$, e.g., reduce $O(|\mathcal{D}_{pair}| \cdot cost_{dist})$ to $O(|\mathcal{D}_{pair}|)$ if it employs $O(1)$ cost lower and upper bounds.

Example: Consider the demo example in Figure 5(a). Let $S_{\downarrow}, S$ and $S^{\uparrow}$ be the set of object pairs with $UB_{dist}(o_q, o_i) \le 4$, $dist(o_q, o_i) \le 4$ and $LB_{dist}(o_q, o_i) \le 4$, respectively. We have $S_{\downarrow} = \{(o_q, o_1), (o_q, o_2), (o_q, o_3)\}, S = \{(o_q, o_1), (o_q, o_2), (o_q, o_3), (o_q, o_4)\}$, and $S^{\uparrow} = \{(o_q, o_1), (o_q, o_2), (o_q, o_3), (o_q, o_4), (o_q, o_5), (o_q, o_6), (o_q, o_7)\}$. We have $S_{\downarrow} \subseteq S \subseteq S^{\uparrow}$, thus $F_{\downarrow}(4) = |S_{\downarrow}| = 3 \le F(4) = |S| = 4 \le F^{\uparrow}(4) = |S^{\uparrow}| = 7$, as shown in Figure 5(b). The black, red and blue curves in Figure 5(b) show the lower bound, exact and upper bound of cumulative distance distribution in dataset $\mathcal{D}_{pair}$, respectively.

Pair	LB_{dist}	$dist$	UB_{dist}
(o_q, o_1)	0	1	3
(o_q, o_2)	0	2	3
(o_q, o_3)	0	3	4
(o_q, o_4)	1	4	5
(o_q, o_5)	2	5	6
(o_q, o_6)	3	5	9
(o_q, o_7)	4	6	10
(o_q, o_8)	5	6	10

(a) distances



(b) cumulative curves

Fig. 5. Illustration example of $F_{\downarrow}(d)$ and $F^{\uparrow}(d)$

3.2 Verification of $\widetilde{F}(d)$

Our next issue is to verify whether the approximate distance distribution $\widetilde{F}(d)$ satisfies the error requirement. The challenge is to avoid computing the exact distance distribution $F(d)$, which is expensive. Lemma 2 offers a simple verification condition; it suffices to test whether the function $F^{\uparrow}(d) - F_{\downarrow}(d)$ is below the error threshold 2δ for all d. When this verification condition evaluates to true, it guarantees that the approximate function $\widetilde{F}(d) = \frac{F^{\uparrow}(d)+F_{\downarrow}(d)}{2}$ (i.e., $\lambda = \frac{1}{2}$ in Equation (2)) satisfies the error requirement.

Lemma 2 (Error-bounded Guarantee)**.** *Suppose $\widetilde{F}(d) = \lambda F^{\uparrow}(d) + (1-\lambda) F_{\downarrow}(d)$ (see Equation (2)). If $F^{\uparrow}(d) - F_{\downarrow}(d) \le 2\delta$ for every $d \in [0, +\infty)$, then it satisfies $|\widetilde{F}(d) - F(d)| \le \delta$ for every $d \in [0, +\infty)$ when $\lambda = \frac{1}{2}$.*

Proof.

$$\begin{aligned}
&|\widetilde{F}(d) - F(d)| = |\lambda F^{\uparrow}(d) + (1-\lambda) F_{\downarrow}(d) - F(d)| \\
&= |\lambda (F^{\uparrow}(d) - F(d)) + (1-\lambda)(F_{\downarrow}(d) - F(d))| \\
&\le \lambda |F^{\uparrow}(d) - F(d)| + (1-\lambda)|F_{\downarrow}(d) - F(d)| \\
&= \lambda (F^{\uparrow}(d) - F(d)) + (1-\lambda)(F(d) - F_{\downarrow}(d)), \text{ by Definition 2} \\
&= \lambda F^{\uparrow}(d) + (1 - 2\lambda) F(d) - (1-\lambda) F_{\downarrow}(d) \\
&= \frac{F^{\uparrow}(d) - F_{\downarrow}(d)}{2}, \text{ by setting } \lambda = \frac{1}{2} \\
&\le \frac{2\delta}{2} = \delta, \text{ since } F^{\uparrow}(d) - F_{\downarrow}(d) \le 2\delta.
\end{aligned}$$

$\square$

It remains to discuss how to efficiently compute the continuous function $F^{\uparrow}(d) - F_{\downarrow}(d)$, which involves infinitely-many possible values of d in the range $[0, +\infty)$. The main challenge is to do verification exactly, yet in a form of finite distance terms. Since both $F^{\uparrow}(d)$ and $F_{\downarrow}(d)$ can be expressed as piecewise step functions, it is possible to compute the function $F^{\uparrow}(d) - F_{\downarrow}(d)$ efficiently, by applying the following lemma.

Lemma 3. *Let $F_{\downarrow}(d)$ and $F^{\uparrow}(d)$ be the lower bound and the upper bound of cumulative distance distribution function for the dataset $\mathcal{D}_{pair}$. It holds that the difference function*

$$\mathsf{diff}(d) = F^{\uparrow}(d) - F_{\downarrow}(d) \tag{3}$$

can be expressed as a piecewise step function with at most $2|\mathcal{D}_{pair}| + 1$ intervals.

Proof. $F_{\downarrow}(d)$ and $F^{\uparrow}(d)$ are non-decreasing piecewise step functions because they are cumulative distance distribution functions defined by finite pairs in $\mathcal{D}_{pair}$. Thus, we express $F_{\downarrow}(d)$ and $F^{\uparrow}(d)$ as follows:

$$F_{\downarrow}(d) = \begin{cases} l_0, d \in [0, x_1) \\ l_1, d \in [x_1, x_2) \\ \ldots \\ n, d \in [x_n, +\infty) \end{cases} \qquad F^{\uparrow}(d) = \begin{cases} u_0, d \in [0, y_1) \\ u_1, d \in [y_1, y_2) \\ \ldots \\ n, d \in [y_n, +\infty) \end{cases}$$

where $n = |\mathcal{D}_{pair}|$, x_i, y_i, l_i, u_i are constants.

The endpoints of intervals of $\mathsf{diff}(d)$ must be in the following set: $\{0, +\infty\} \cup \{x_1, \cdots, x_n\} \cup \{y_1, \cdots, y_n\}$. There are at most $2n + 2$ endpoints. Thus, there are at most $2n + 1$ intervals. □

The above observation suggests that the merge-join algorithm can be used to compute the piecewise step function $\mathsf{diff}(d)$ and verify whether each step satisfies the error requirement $\mathsf{diff}(d) \leq 2\delta$ efficiently.

For example, in Figure 5(b), $F^{\uparrow}(d)$ and $F_{\downarrow}(d)$ are piecewise step functions with at most 8+1 intervals, as the set $\mathcal{D}_{pair}$ contains 8 pairs. The function $\mathsf{diff}(d)$ contains at most $2 \cdot 8 + 1 = 17$ intervals (in fact it contains only 9 intervals).

3.3 Tightening of $F_{\downarrow}(d)$ and $F^{\uparrow}(d)$

We proceed to discuss how to tighten the bound functions $F_{\downarrow}(d)$ and $F^{\uparrow}(d)$ efficiently if $\widetilde{F}(d)$ cannot pass the verification. For a given data object pair (o_q, o_i), the lower and upper bound will be tighter during the refinement process. For the widely used distance measures, e.g., dynamic time warping (DTW), discrete Fréchet Distance (DFD), there are many research works devised lots of lower and upper bounds for them [18], [21], [22], as shown in Table 2.

However, it is not trivial to decide which bound should be used. We benchmark all DFD bounds by the tightness and computation cost. Figures 6(a) and (b) illustrate the computation cost (x-axis) and the tightness (y-axis) of DFD lower bounds (LB/DFD) and upper bounds (DFD/UB), respectively.

A straightforward strategy is that all bounds are applied from quick-and-dirty one to slow-and-accurate one in Figure 6. However, some of bounds are dominated by others, for example, LB_{row} is dominated by LB_g in Figure 6(a). Intuitively, LB_{row} can be ignored if it employs LB_g. To exploit this property, we propose another strategy, called Prioritizing bounds, to tighten the bounds of each object pair (o_q, o_i) efficiently. In particular, we only employ these bounds in the skyline of Figure 6, instead of all bounds in the straightforward strategy. For example, only LB_{cell}, LB_g, UB_g will be used for DFD. We evaluate the effect of both strategies in Section 6.5. For ED and DTW bounds, we use the same idea and refer the reader to the bound benchmark result of DTW

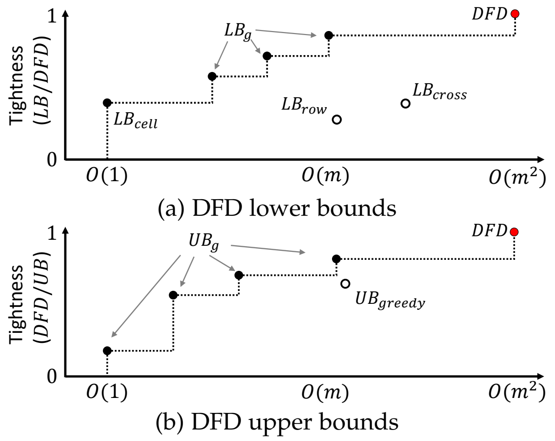


Fig. 6. The evaluation of DFD lower and upper bounds

in [22]. Moreover, our solution is generic to the distance measures which have existing lower or upper bounds.

In summary, our solution exploits the lower and upper bound functions of each distance measure (which incur cheap computation cost) to render error-guaranteed approximate results efficiently. It provides worst-case error guarantees on the distance distribution for different distance measures, which do not rely on the specific lower and upper bound functions. Obviously, the distance lower and upper bound functions will affect the performance of our solution. We will evaluate their effectiveness in the experimental section shortly.

4 Optimization Techniques

In this section, we devise a suite of optimization techniques to improve the performance of our proposal in Section 3. In particular, the performance improvements are from (i) we exploit data index to compute the lower and upper bounds of a group of data objects, instead of computing them one by one (see Section 4.1); (ii) we devise a lazy lower and upper bound refinement method, which only refine the lower and upper bounds of data object pairs which may lead to invalid intervals (see Section 4.2); and (iii) we reduce the exact distance computations to ensure the approximate distance distribution $\widetilde{F}(d)$ satisfies the error requirement (see Section 4.3).

4.1 Computing Index-based Bounds

We first compute the distance lower and upper bound of every object $(o_q, o_i) \in \mathcal{D}_{pair}$, then obtain $F_{\downarrow}(d)$ and $F^{\uparrow}(d)$ accordingly. When the data set is large enough, the cost to compute $F_{\downarrow}(d)$ and $F^{\uparrow}(d)$ is also very expensive even the $O(1)$ cost bounds are used. In this section, we employ well-studied data object index to accelerate $F_{\downarrow}(d)$ and $F^{\uparrow}(d)$ computation. For example, R-tree [25] is a standard index to maintain spatial objects. Given a dataset $\mathcal{D}$ of spatial points, it first constructs the R-tree index r. For each entry e in r has a minimum bounded rectangle, i.e. $\mathsf{MBR}(e)$, which could provide a common distance lower and upper bound for all object pairs (o_q, o_i), where $o_i \in e$. Through this, $F_{\downarrow}(d)$ and $F^{\uparrow}(d)$ can be derived without computing the lower and upper bound of each object pair $(o_q, o_i) \in \mathcal{D}_{pair}$. Moreover, the

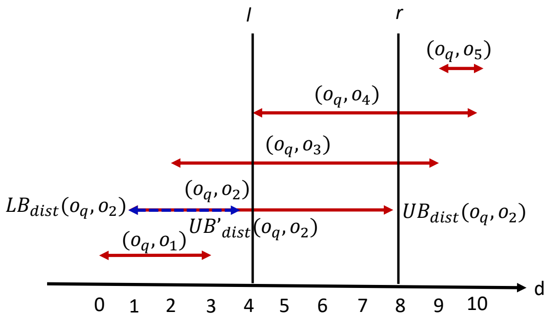


Fig. 7. Distance lower and upper bound updates

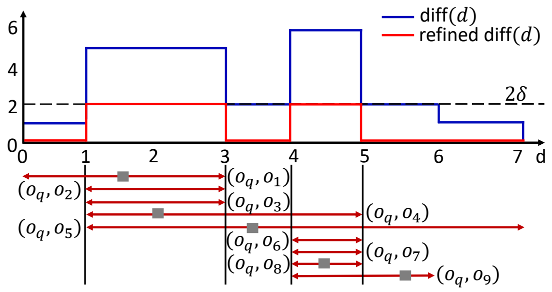


Fig. 8. Exact distance computation example

tightness of the lower and upper bound between MBR(e) and o_q depends on the height of entry e in R-tree r. In other words, different levels in the index r serve as different lower and upper bound functions in our framework. Obviously, the effectiveness of index-based bounds are relying on the underlying objects. For example, the performance of R-tree index will degenerate due to *the curse of dimensionality* in high dimensional data objects. In this case, users may disable index-based bound option in our framework as it does not improve the overall performance.

4.2 Refining Optimized Invalid Interval

In this section, we consider how to tighten $F_{\downarrow}(d)$ and $F^{\uparrow}(d)$ with tighter $LB_{dist}(o_q, o_i)$ and $UB_{dist}(o_q, o_i)$ efficiently. The major contributions are: (i) it only updates the lower and upper bounds of data object pairs which affect the output of verification function, (ii) it employs a lazy bound updating method which computes the tighter lower and upper bound if and only if it is necessary.

Consider the example shown in Figure 7 with error threshold $\delta = 1$. $\forall d \in [4, 8), F^{\uparrow}(d) = 4$ and $F_{\downarrow}(d) = 1$. Thus, $\forall d \in [4, 8), \mathsf{diff}(d) = F^{\uparrow}(d) - F_{\downarrow}(d) = 4 - 1 = 3 \geq 2\delta = 2$. Its approximate distance distribution $\widetilde{F}(d)$ does not satisfy the requirement, i.e., Verify$(F_{\downarrow}(d), F^{\uparrow}(d), \delta)$ return false. Our method will refine $F_{\downarrow}(d)$ and $F^{\uparrow}(d)$ by tighter lower and upper bound functions $LB'_{dist}(o_q, o_i)$ and $UB'_{dist}(o_q, o_i)$, respectively. A simple strategy is to compute the lower and upper bound by $LB'_{dist}(o_q, o_i)$ and $UB'_{dist}(o_q, o_i)$ for every object pair in dataset $\mathcal{D}_{pair}$. However, this strategy wastes computation cost as many object pairs in $\mathcal{D}_{pair}$ do not affect the verification result.

Returning to the example in Figure 7, the interval $[4, 8)$ is invalid. The functions $F_{\downarrow}(d)$ and $F^{\uparrow}(d)$ will be refined by tighter $LB_{dist}(o_q, o_i)$ and $UB_{dist}(o_q, o_i)$. However, it is not necessary to update all object pairs' lower and upper bounds. For example, object pair (o_q, o_1) and (o_q, o_5) can be ignored because $UB_{dist}(o_q, o_1) = 3 < 4$ and $LB_{dist}(o_q, o_5) = 9 \geq 8$, as shown in Figure 7.

Let $[l, r)$ be an interval of the function diff(d) such that it violates the error bound requirement. With the above observation, we have only the object pairs which lower and upper bounds overlap with $[l, r)$ should be refined. Formally, the refinement object pair set R is defined as:

$$\mathsf{R} = \{(o_q, o_i) : [LB_{dist}(o_q, o_i), UB_{dist}(o_q, o_i)) \cap [l, r) \neq \emptyset\}.$$

For example, the refinement set for interval $[l = 4, r = 8)$ is $\mathsf{R} = \{(o_q, o_2), (o_q, o_3), (o_q, o_4)\}$ in Figure 7.

Moreover, we propose to update the distance bounds of $(o_q, o_i) \in \mathsf{R}$ in a lazy manner. Specifically, its tighter distance bounds will be computed if and only if the distance value interval $[l, r)$ is still invalid. Take Figure 7 as an example, the distance value interval $[l = 4, r = 8)$ is invalid and the refinement object set is $\mathsf{R} = \{(o_q, o_2), (o_q, o_3), (o_q, o_4)\}$. The upper bound of pair (o_q, o_2) is updated to $UB'_{dist}(o_q, o_2)$, shown as the blue line. With this update, we have: $F^{\uparrow}(4) - F_{\downarrow}(4) = 4 - 2 = 2 \leq 2\delta$. Now the interval $[l = 4, r = 8)$ becomes valid, then the refinement process terminates. It improves the performance as it does not compute the updated distance bounds of (o_q, o_3) and (o_q, o_4) by the lazy manner.

4.3 Reducing Exact Distance Computations

We consider the case that the approximate distance distribution $\widetilde{F}(d)$ still violates the error bound requirement after all lower and upper bounds have been applied in this section. The exact distances of object pairs in refinement set R will be computed in this situation. Our proposed optimization reduces the exact distance computations then the approximate distance distribution $\widetilde{F}(d)$ still satisfies the error requirement, i.e., $\forall d \in [0, \infty), \mathsf{diff}(d) \leq 2\delta$.

We commence the presentation by considering the example in Figure 8 with error bound $\delta = 1$. After applying all possible lower and upper bounds, $\forall d \in [1, 3)$, $F_{\downarrow}(d) = |\emptyset| = 0$ and $F^{\uparrow}(d) = |\{(o_q, o_1), (o_q, o_2), (o_q, o_3), (o_q, o_4), (o_q, o_5)\}| = 5$. Thus, $\forall d \in [1, 3), \mathsf{diff}(d) = F^{\uparrow}(d) - F_{\downarrow}(d) = 5 > 2\delta = 2$, it violates the error bound requirement. Consider data object (o_q, o_1) in Figure 8, its distance lower and upper bound are 0 and 3, respectively. Suppose its exact distance is $dist(o_q, o_1) = 1.5$, we are able to reduce diff(d) from 5 to 4 $\forall d \in [1, 3)$ due to (i) $\forall d \in [1, 1.5), \mathsf{diff}(d) = F^{\uparrow}(d) - F_{\downarrow}(d) = 4 - 0 = 4$ and (ii) $\forall d \in [1.5, 3), \mathsf{diff}(d) = F^{\uparrow}(d) - F_{\downarrow}(d) = 5 - 1 = 4$. That is, computing the exact distance of (o_q, o_1) reduces 1 from diff(d) for every $d \in [1, 3)$. Through the above observation, we have Lemma 4 as follows.

Lemma 4. *Given an invalid interval $[l, r)$ in* diff *function, i.e., $\forall d \in [l, r), \mathsf{diff}(d) = F^{\uparrow}(d) - F_{\downarrow}(d) > 2\delta$. It requires* diff$(l) - 2\delta$ *exact distance computations to guarantee $\forall d \in [l, r), \mathsf{diff}(d) \leq 2\delta$.*

Proof. We have that each exact distance computation will reduce 1 from diff(d) for every $d \in [l, r)$ via the above discussion. In order to turn diff(d) to 2δ, i.e., to guarantee that $\forall d \in [l, r), \mathsf{diff}(d) = 2\delta$, it needs diff$(d) - 2\delta$ exact distance computation among data object pairs. □

Returning to the example in Figure 8, $\forall d \in [1,3), \mathsf{diff}(d) = F^{\uparrow}(d) - F_{\downarrow}(d) = 5$. According to Lemma 4, it requires $\mathsf{diff}(l) - 2\delta = 5 - 2 = 3$ exact distance computations, then $\forall d \in [1,3), \mathsf{diff}(d) \leq 2\delta$, e.g., compute the exact distances of (o_q, o_1), (o_q, o_2) and (o_q, o_3).

With Lemma 4, we conclude the corollary as follows, we omit its proof as it is applying Lemma 4 directly.

Corollary 1. *Let $\Phi = \{[l,r) : \forall d \in [l,r), \mathsf{diff}(d) > 2\delta\}$, it requires at most $\sum_{\forall [l,r) \in \Phi} (\mathsf{diff}(l) - 2\delta)$ exact distance computations, then the approximate distance distribution $\widetilde{F}(d)$ satisfies the error requirement, i.e., $\forall d \in [0, \infty), \mathsf{diff}(d) \leq 2\delta$.*

Consider $\mathsf{diff}(d)$, as the blue curve shown in Figure 8, we have $\Phi = \{[1,3), [4,5)\}$. It requires at most $(\mathsf{diff}(1) - 2) + (\mathsf{diff}(4) - 2) = (5-2) + (6-2) = 7$ exact distance computations then $\widetilde{F}(d)$ satisfies the error requirement, i.e., $\forall d \in [0, \infty), \mathsf{diff}(d) \leq 2$. For example, compute exact distances of (o_q, o_1), (o_q, o_2) and (o_q, o_3) for $\forall d \in [1,3)$, and (o_q, o_4), (o_q, o_5), (o_q, o_6) and (o_q, o_7) for $\forall d \in [4,5)$, thus, the approximate distance distribution $\widetilde{F}(d)$ satisfies the error requirement.

Through Corollary 1, for any $\mathsf{diff}(d) = F^{\uparrow}(d) - F_{\downarrow}(d)$ function, we can obtain the number of exact distance computations then let $\mathsf{diff}(d)$ satisfy the error bound requirement. But the qualified $\mathsf{diff}(d)$ can be derived with fewer exact distance computations. Take the example in Figure 8, we can compute exact distances of (o_q, o_1), (o_q, o_4) and (o_q, o_5) for $\forall d \in [1,3)$. $\forall d \in [4,5)$, $\mathsf{diff}(d)$ changes from 6 to 4 at the same time as the lower and upper bounds of (o_q, o_4) and (o_q, o_5) also overlap with interval $[4,5)$, as shown in Figure 8. Thus, we only need compute 2 more exact distance pairs for interval $[4,5)$, e.g., compute (o_q, o_8) and (o_q, o_9). In summary, $\forall d \in [0, +\infty), \mathsf{diff}(d) \leq 2\delta$ (shown as the red curve in Figure 8) holds through 5 exact distance computations (i.e., (o_q, o_1), (o_q, o_4), (o_q, o_5), (o_q, o_8) and (o_q, o_9)), it is less than 7 (computed by Corollary 1) in this example.

With the above observation, we propose a simple greedy strategy to reducing the exact distance computations as follows.

1) identify $(o_q, o_i) \in \mathcal{D}_{pair}$ whose distance lower and upper bound has the largest number of intersected intervals in Φ;
2) compute $dist(o_q, o_i)$ to update diff function and Φ;
3) repeat step 1) until $\forall d \in [0, \infty), \mathsf{diff}(d) \leq 2\delta$.

This greedy strategy reduces the exact distance computations as it refines $\mathsf{diff}(d)$ in multiple intervals in Φ by one exact distance computation.

5 Distance Distribution Histogram

As the example shown in Figures 1(b) and (d) (see Section 1), distance distribution histograms also have been widely used in many applications. In this section, we extend our solution for error-bounded ACDD problem to error-bounded approximate distance distribution histogram computation problem (ADDH). We first define the exact distance distribution histogram (DDH) in Definition 4 formally.

Definition 4 (Distance Distribution Histogram)**.** *Given a set $\mathcal{D}_{pair}$ of object pairs and a distance measure $dist$, the distance distribution histogram H is defined as an array of buckets which covers a domain interval, e.g., $[0, d_{max}]$[1]. All buckets in H have the equi-width μ. The total number of buckets in H is $B = \lceil \frac{d_{max}}{\mu} \rceil$. The ith bucket stores the total number of object pairs which distance value in the interval, i.e., $\mathsf{H}(i) = |\{(o_q, o_j) \in \mathcal{D}_{pair} : (i-1)\mu \leq dist(o_q, o_j) < i\mu\}|$. Without loss of generality, we have $\mathsf{H}(0) = 0$.*

We formally define the error-bounded approximate distance distribution histogram $\widetilde{\mathsf{H}}$ as follows.

Problem 3 (Error-bounded Approximate Distance Distribution Histogram)**.** *Given a set $\mathcal{D}_{pair}$ of object pairs, a distance measure $dist$, an error threshold ϵ and a histogram bucket width μ, the approximate distance distribution histogram (ADDH) returns a function of bucket i such that the condition*

$$|\widetilde{\mathsf{H}}(i) - \mathsf{H}(i)| \leq \epsilon$$

holds for every value of $i \in [1, B]$.

Lemma 5 provides a solution for Problem 3 by utilizing the result of approximate cumulative distance distribution $\widetilde{F}(d)$ in Problem 2.

Lemma 5. *Given $\epsilon = 2\delta$, suppose $\widetilde{F}(d)$ is the returning result of Problem 2 with inputs dataset $\mathcal{D}_{pair}$ and a distance measure $dist$. The result of Problem 3, $\widetilde{\mathsf{H}}(i)$, can be derived from $\widetilde{F}(d)$ by $O(B)$ time.*

Proof. We first prove that $\forall i \in [1, B], |\widetilde{\mathsf{H}}(i) - \mathsf{H}(i)| \leq \epsilon$. Combining with the definition of cumulative distance distribution function $F(d)$ in Problem 1, the distance distribution histogram function $\mathsf{H}(i)$ in Definition 4 could be transformed to

$$\mathsf{H}(i) = F(i * \mu) - F((i-1) * \mu), \forall i \in [1, B].$$

Similarly, the approximate distance distribution histogram function $\widetilde{\mathsf{H}}(i)$ could be computed by

$$\widetilde{\mathsf{H}}(i) = \widetilde{F}(i * \mu) - \widetilde{F}((i-1) * \mu), \forall i \in [1, B].$$

Let $d = i * \mu$ and $d' = (i-1) * \mu$, $\forall i \in [1, B]$, we have

$$\begin{aligned} |\widetilde{\mathsf{H}}(i) - \mathsf{H}(i)| &= |(\widetilde{F}(d) - \widetilde{F}(d')) - (F(d) - F(d'))| \\ &= |(\widetilde{F}(d) - F(d)) + (F(d') - \widetilde{F}(d'))| \\ &\leq |\widetilde{F}(d) - F(d)| + |\widetilde{F}(d') - F(d')| \\ &\leq \delta + \delta = 2\delta = \epsilon \end{aligned}$$

Next, we analyze the time complexity to derive $\widetilde{\mathsf{H}}(i)$ from $\widetilde{F}(d)$. Obviously, it is $O(B)$ as i is from 1 to B and for each i it takes $O(1)$ time to compute $\widetilde{\mathsf{H}}(i)$ from $\widetilde{F}(d)$. □

Example: Given a set of object pairs $\mathcal{D}_{pair}$ in GeoLife dataset with histogram bucket width $\mu = 0.005$. We use DFD to measure the distance between each pair $(o_q, o_j) \in \mathcal{D}_{pair}$. The solid bars in Figure 9 show the exact distance distribution histogram of $\mathcal{D}_{pair}$. The patterned bars are the approximate distance distribution histogram $\widetilde{\mathsf{H}}(i)$ of $\mathcal{D}_{pair}$ with error bound $\epsilon = |\mathcal{D}_{pair}| \cdot 2\%$.

1. d_{max} is the largest distance of object pair $(o_q, o_j) \in \mathcal{D}_{pair}$

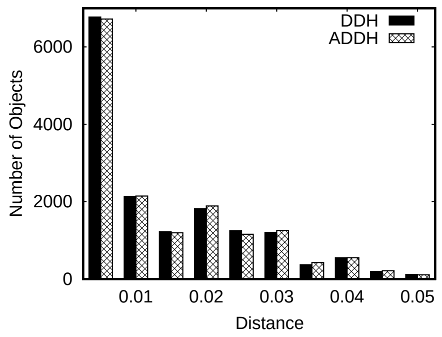


Fig. 9. Exact and approximate ($\epsilon = |\mathcal{D}_{pair}| \cdot 2\%$) distance distribution histogram on GeoLife with DFD

TABLE 3
Real dataset information

Dataset	Data type	# of objects	# attributes	Size
TAO	Time series	41,149	1,008	312 MB
ECG	Time series	380,522	421	1.2 GB
OSM-FULL	Trajectories	2,433,433	1096	39.7 GB

6 Experimental Evaluation

In this section, we present our empirical findings. In Section 6.1, we describe the experimental setting. In Section 6.2, we conduct a case study on a real taxi trajectory dataset to demonstrate the applicability of ACDD and ADDH. In Section 6.3, we investigate the accuracy of our proposed approximate solutions for ACDD and ADDH, respectively. In Section 6.4, we evaluate the efficiency of our solutions with three distance measures: (i) Euclidean distance (ED), (ii) dynamic time warping (DTW) and (iii) discrete Fréchet distance (DFD). Finally, in Section 6.5, we investigate the effectiveness of the optimizations proposed in Section 4. For the sake of experimental reproducibility, we have posted the datasets and source codes at [26].

6.1 Experimental Setting

Dataset: We used three real world datasets with diverse fields and scales of size, i.e., TAO[2], ECG[3] and OSM-FULL[4] for ED, DTW, and DFD, respectively. They have been studied in some existing work (e.g., [27], [22] and [11]). Table 3 summarizes the information of these datasets.

Implementation and methods: In the experimental study, we compare the following methods:

- SAMP. The sampling-based solution picks objects randomly to calculate approximate distance distributions with a fixed sampling fraction α (see Section 2.2).
- ACDD. Our proposed error bound guaranteed solution, which is presented in Section 3.
- ACDD*. ACDD with all proposed optimization techniques in Section 4.

Table 4 summarizes the indices and bounds used in ACDD and ACDD*. The PAA-based and group-based

2. https://tao.ndbc.noaa.gov/
3. https://www.physionet.org/content/edb/1.0.0/
4. https://ftp5.gwdg.de/pub/misc/openstreetmap/

TABLE 4
Index and bounds for different distance measures

Measure	Index	Lower bound	Upper bound
ED	M-tree [28]	LB_{PAA}, LB_{FNN}	UB_{PAA}, UB_{FNN}
DTW	R-tree	$LB_{KimFL}, LB^{EQ}_{Keogh}$	UB_{ED}, UB_{FNN}
DFD	R-tree	LB_{cell}, LB_g	UB_g

TABLE 5
Tested parameters in experiments

Parameter	Tested values
No. of groups in LB_{PAA} for ED	16, 32, 64
No. of groups in LB_g, UB_g for DFD	1, 2, 4, 8, 16, 32, 64, 128
Error threshold (δ/n)	1%,2%,3%,4%,5%
Cardinality of $\mathcal{D}_{pair}$ (n)	$10^4, 10^5, 10^6, 10^7, 10^8$

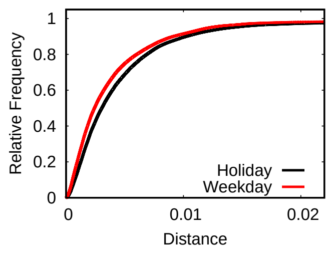
bounds (e.g., LB_{PAA}, UB_g) are parameterized bounds that take the number of groups as a parameter. We summarized the tested number of groups for PAA-based bounds (ED) and group-based bounds (DFD) in Table 5. In all subsequent experiments, we tighten $F_{\downarrow}(d)$ and $F^{\uparrow}(d)$ by enumerating these values from small to large for both PAA-based and group-based bounds. In addition, Table 5 also includes the tested values of error threshold and the cardinality of $\mathcal{D}_{pair}$ for scalability evaluation.

All methods are implemented in C++. We measure the performance of our proposal in three widely used distance measures: ED, DTW and DFD. The reported response time includes both index construction and error-bounded distance distribution computation. The experiments (with single thread) run on a PC with Intel Xeon Gold 5122 3.60GHz processor and 32 GB main memory.

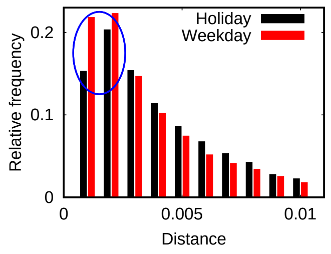
6.2 Case Study

We start by presenting the utility of ACDD and ADDH with a case study on the real trajectory dataset, Taxi Service Trajectory (TST) [29]. It includes 1,710,660 taxi trip trajectories generated by 442 vehicles in Porto, Portugal from 1 July 2013 to 30 June 2014. The average number of GPS points in each trajectory is 48.76. We compute the ACDD and ADDH with DFD of all taxi trajectory pairs in December 18, 2013 (weekday) and December 25, 2013 (Christmas holiday), respectively. We compare the distribution histograms of weekday and holiday by relative frequency as they have different number of trajectories.

Figures 10(a) and (b) show the ACDD and ADDH in the two days with $\delta = n \cdot 3\%$ and $\epsilon = n \cdot 3\%$, respectively. As we will present shortly in Section 6.3, both ACDD and ADDH capture the trend of exact distance distributions precisely. Moreover, ACDD and ADDH have much cheaper computation cost. For example, it only takes 5.98s to compute the ACDD at Christmas holiday, as Holiday curve shown in Figure 10(a), but it incurs 28.9s to compute its corresponding exact cumulative distance distribution. It is interesting to note that for ADDH in Figure 10(b), the taxi trajectories in the weekday have more similar pairs than the holiday, especially when the distance threshold is less than 0.002, as the blue circle shown in Figure 10(b). On the one hand, the above observation indicates the distribution histograms of



(a) ACDD ($\delta = n \cdot 3\%$)



(b) ADDH ($\epsilon = n \cdot 3\%$)

Fig. 10. Taxi trip distance distribution with DFD on weekday (December 18, 2013) and holiday (December 25, 2013)

taxi trajectories in weekday and holiday are not the same, which can be exploited to improve the trajectory retrieval performance as different distance distribution histograms can affect the index performance [30]. On the other hand, it also reveals the difference of taxi trips in weekday and holiday in Porto, Portugal, which can be utilized to optimize the taxi scheduling and other smart-city applications.

6.3 Accuracy Evaluation

In this section, we study the accuracy of our proposed approximate solutions for ACDD and ADDH problem, as defined in Problems 2 and 3, respectively.

Figures 11(a), (b) and (c) illustrate the cumulative distance distributions with three distance measures ED, DTW and DFD, respectively. In each figure, it includes three cumulative distance distribution curves, i.e., black dashed exact cumulative distance distribution (CDD), solid red approximate cumulative distance distribution curve (ACDD with $\delta = n \cdot 1\%$), and solid blue sampling based distance distribution curve (SAMP with $\alpha = 1\%$). Obviously, ACDD curve demonstrates excellent approximation power to the exact cumulative distance distribution in all distance measures. The margins between CDD and SAMP are quite large, especially, in DTW. This also confirms the inefficiency of the sampling method on computing the cumulative distance distribution. Moreover, SAMP cannot provide the theoretical worst-case error guarantees.

The distance distribution histograms in Figures 12(a), (b) and (c) are derived from the cumulative distance distributions in Figure 11 with the histogram bucket width $\mu = 5, 1000$, and 70 for ED, DTW and DFD, respectively. Obviously, ADDH with $\epsilon = 2\delta = n \cdot 2\%$ captures the trend of exact distance distribution histogram (DDH) excellently. Visually, SAMP solution performs worse than ADDH in all three distance measures, as shown in Figure 12.

We then quantify the approximation error of our proposed solutions, by using E_{max} (defined in Section 2.2), in Figure 13. For each distance measure, we randomly selected 100 objects as queries. Figures 13(a) and (b) show the maximal E_{max} and average E_{max} between ACDD and CDD among all selected queries by varying δ from $n \cdot 1\%$ to $n \cdot 5\%$, respectively. Obviously, the maximal E_{max} and average E_{max} in all three distance measures are below the error-bounded requirement δ (see red dotted line) as our solution guarantees the worst-case error theoretically. In Figure 13(a), the maximal E_{max} of ACDD in DFD performs excellently. Specifically, the maximal E_{max} is less than $n \cdot 1\%$ even with $\delta = n \cdot 5\%$ setting. Considering the average E_{max} among all selected queries, ACDD performs even more better than expected. For example, the average E_{max} is always less than $n \cdot 2\%$ and $n \cdot 0.7\%$ with δ ranged from $n \cdot 1\%$ to $n \cdot 5\%$ in ED and DFD, respectively, as shown in Figure 13(b). We omit the approximate ability study for approximate distance distribution histogram (ADDH) as it has similar observations with its on ACDD in Figure 13.

6.4 Efficiency Evaluation

In this section, we focus on the efficiency of our methods on different datasets with various distance measures. We omit ADDH in Section 6.4 and 6.5 since it is derived from ACDD with low cost and all optimization techniques in ACDD can be extended into ADDH directly.

We compare our error-bounded approximate cumulative distance distribution methods (ACDD and ACDD*) with SAMP (the approximate method without error bound guarantee) in Figure 14. In each experiment, the maximum error E_{max} (see Definition 1) ranges from $n \cdot 1\%$ to $n \cdot 5\%$. Each value point in ACDD and ACDD* is plotted with its error threshold and the average response time of 100 randomly selected queries from the dataset. However, SAMP does not guarantee the maximum error bound. In order to provide meaningful comparison, SAMP method is incurred with different sample ratio from 1% to 99% with step size 1%. For each SAMP execution, we plot its average response time and average E_{max} (see squares in Figure 14) if and only if its average E_{max} is in the range of $n \cdot 1\%$ to $n \cdot 5\%$.

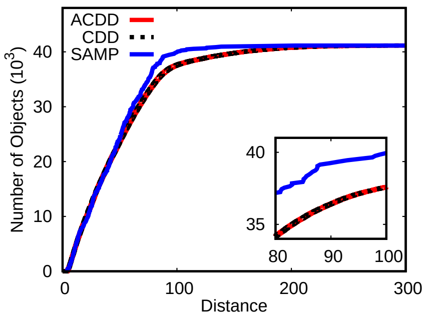
Performance of ACDD*: The experimental results of ED, DTW and DFD on real datasets TAO, ECG and OSM-FULL are shown in Figures 14(a), (b) and (c), respectively. First, ACDD* outperforms SAMP in all three datasets in terms of both performance and result quality. Especially, ACDD* is up to 278.6 times faster than SAMP on the largest dataset OSM-FULL. Second, the running time of ACDD and ACDD* falls with the rising of E_{max} as expected. Third, ACDD is slower than ACDD* in all cases as ACDD* is equipped all proposed optimizations. We omit the weaker methods SAMP and ACDD in subsequent experiments.

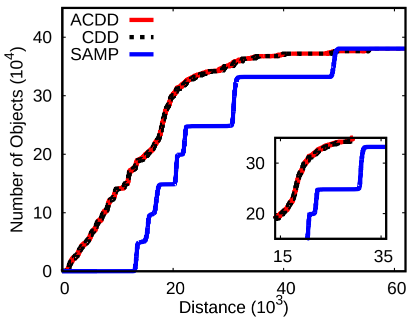
Scalability of ACDD*: We then evaluate the scalability of ACDD* by varying n with ED, DTW and DFD in Figures 15(a), (b) and (c), respectively. We randomly choose a set of objects in the corresponding dataset and construct $\mathcal{D}_{pair}$ by correlation (see Section 2.1). The number of pairwise distances in the subset is from 10^4 to 10^8. The error bound is fixed at $\delta = n \cdot 3\%$.

ACDD* demonstrates its superior scalability in all three distance measures. For example, with millions of object pairs and DFD as the distance measure, ACDD* outperforms the exact solution by 9217.1 times. Our method guarantees $n \cdot 3\%$ worst-case error on the result and returns it within 0.3 seconds. Moreover, the superiority of ACDD* is more obvious when the distance measure is more complex, e.g., the performance gain of ACDD* in DTW and DFD is larger than it in ED, as shown in Figure 15.

6.5 Effectiveness of Optimizations

Before we evaluate individual optimizations in our optimized solution ACDD*, we first evaluate the benefits gained


(a) ED on TAO


(b) DTW on ECG

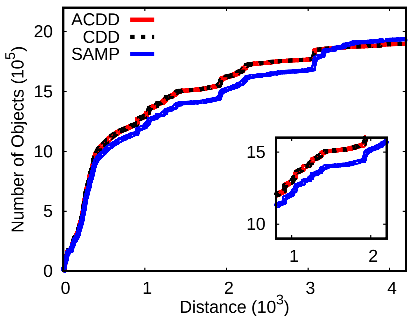
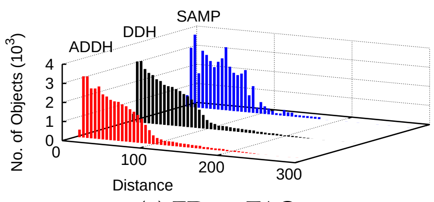
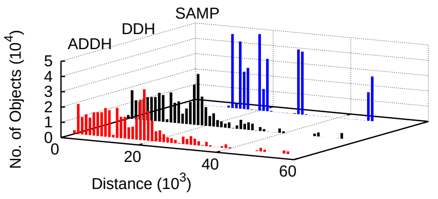

(c) DFD on OSM-FULL

Fig. 11. Cumulative distance distribution study ($\delta = n \cdot 1\%$, $\alpha = 1\%$)


(a) ED on TAO


(b) DTW on ECG

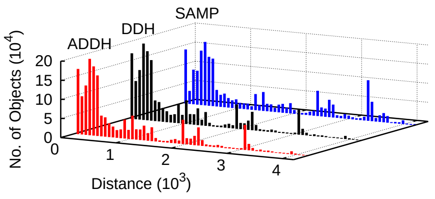
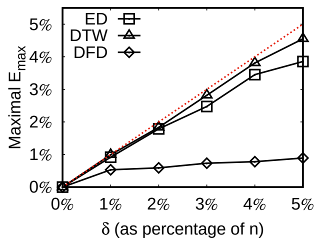

(c) DFD on OSM-FULL

Fig. 12. Distance distribution histogram study ($\epsilon = n \cdot 2\%$, $\alpha = 1\%$)


(a) Maximal E_{max}

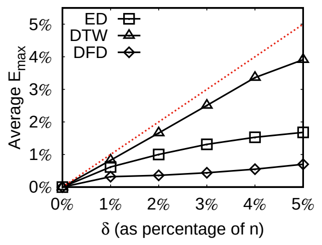

(b) Average E_{max}

Fig. 13. The maximal and average $E_{max}(F, \widetilde{F})$

from prioritizing bounds (see Section 3.3) in Figure 16. For each distance measure, we compare ACDD* with all bounds and ACDD* only with the prioritized bounds in the real dataset. We denote ACDD* with all bounds of the distance measures as ALL-ACDD*. The bounds applied strategy is from dirty-and-quick one to accurate-and-slow one. As shown in Figure 16, ACDD* performs better than ALL-ACDD* in all distance measures on three real datasets.

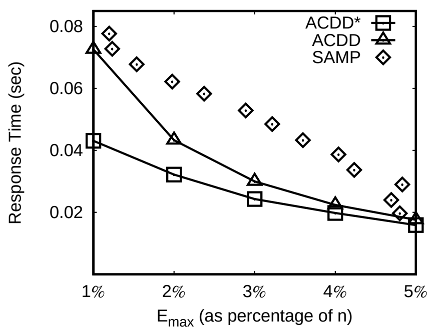
Next, we evaluate the effect of our proposed optimization techniques in Section 4 separately. To isolate the effect of each proposed optimization, we investigate the efficiency of ACDD* with and without the evaluated optimization technique in each experiment while enabling the rest optimization techniques. In other words, the evaluation results only demonstrate the effectiveness of each individual optimization in ACDD*. For sake of presentation and space limitation, we present the experimental results on DFD in Figure 17, and briefly mention the results on ED and DTW in context.

Effect of index-based bounds: We verify the effect of index-based bound techniques (see Section 4.1) on OSM-FULL with DFD in Figure 17(a). The experimental result of ACDD* with index and without index are plotted as ACDD* and ACDD*-w/o-index curves, respectively. Interestingly, ACDD* and ACDD*-w/o-index perform similarly when δ is small, e.g., $n \cdot 1\%$. The reason is the index-based bound is too loose when the error bound requirement is strict. However, ACDD* shows its superiority over ACDD*-w/o-index with the rising of δ. For example, the performance gap between ACDD*-w/o-index and ACDD* widens from $n \cdot 2\%$ to $n \cdot 5\%$, as illustrated in Figure 17(a).

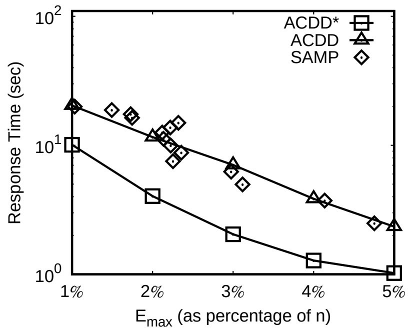
Effect of optimized interval refinement: We then evaluate the effect of optimized interval refinement strategy in Section 4.2 in Figure 17(b). We compare the time cost of bound updating with and without optimized interval refinement strategy in DFD, as ACDD* and ACDD*-non-opt bars shown in Figure 17(b). Obviously, ACDD* outperforms ACDD*-non-opt in all cases as it could save lots of unnecessary bound computations. In particular, the bound refining cost can be reduced up to 50.9% (from 0.0611s to 0.0302s) in DFD. For ED and DTW, it can be up to 62.5% (from 0.0056s to 0.0021s), and 40.5% (from 0.8661s to 0.5161s), respectively.

Effect of exact distance computation reduction: In Section 4.3, we propose an optimization to reduce exact distance computation. To evaluate the effectiveness of this optimization, we compare the time cost of exact distance computation in ACDD* with and without exact distance computation reducing technique in Figure 17(c). We set worst-case error guarantees $\delta = n \cdot 0.1\%, ..., n \cdot 0.5\%$ to show the gains clear [5]. As ACDD* and ACDD*-non-min shown in Figure 17(c), this optimization reduces 41.7% to 51.1% of the exact distance computation time in DFD. It reduces 28.6% to

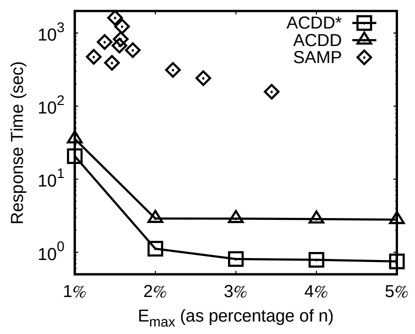
5. When δ is large, our method usually will return results without exact distance computation.



(a) ED on TAO

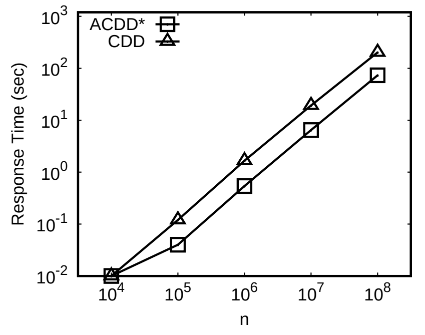


(b) DTW on ECG

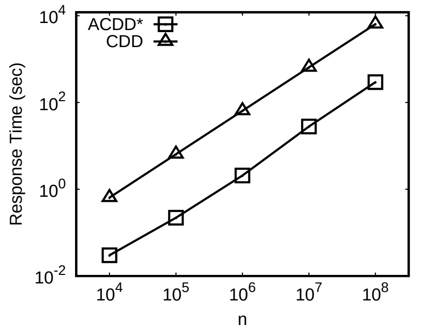


(c) DFD on OSM-FULL

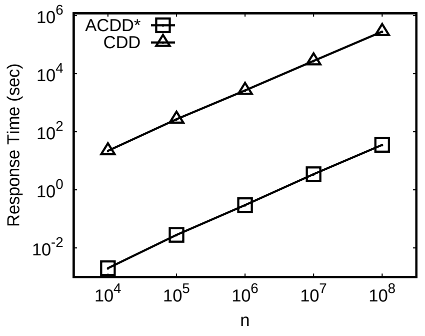
Fig. 14. Response time vs $E_{max}(F, \widetilde{F})$



(a) ED on TAO

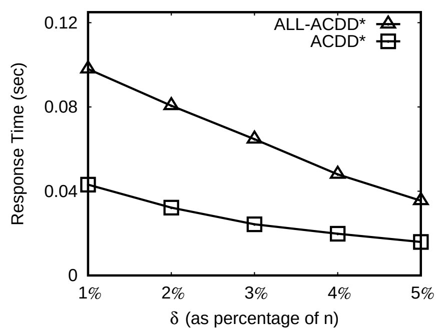


(b) DTW on ECG

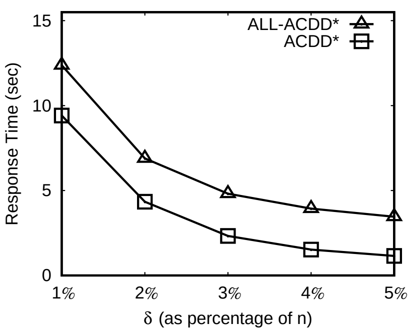


(c) DFD on OSM-FULL

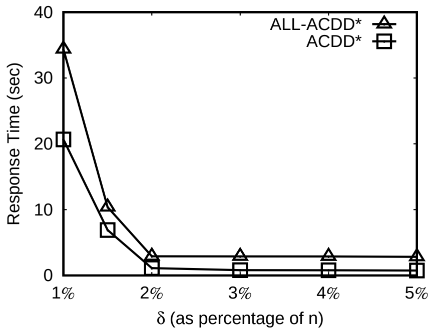
Fig. 15. Response time vs n



(a) ED on TAO

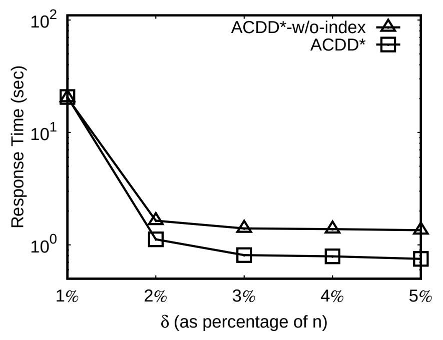


(b) DTW on ECG

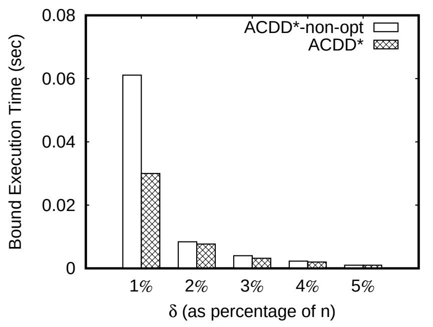


(c) DFD on OSM-FULL

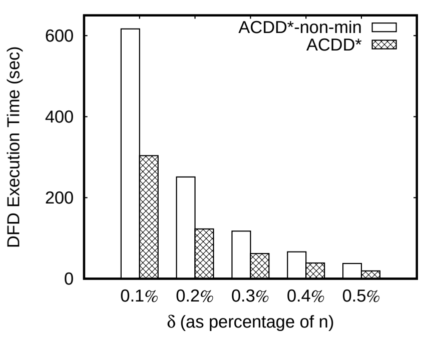
Fig. 16. Effect of prioritizing bounds in Section 3.3



(a) Effect of indexing in Section 4.1



(b) Effect of invalid interval refinement in Section 4.2



(c) Effect of distance computation reduction in Section 4.3

Fig. 17. Effect of optimizations in Section 4 on OSM-FULL with DFD

61.8% in ED and 14.6% to 41.9% in DTW, respectively.

7 Related Work

Multiple research areas are relevant to our problem. In the following, we briefly review the related work in two major areas.

Distance distribution computation problems: In applications like graph analysis [7], [31], time series analysis [32] and natural resource evaluation [33], the distribution of distances among data objects is exploited to reveal the characteristics of a dataset. Kang et al. [7] focus on social networks and develop the radius plot to show the distribution of the "effective radius" of nodes in a social network, where the effective radius of a node v is taken as the 90th-percentile of all distances from v. Qiao et al. [31] consider a wider variety of graphs (e.g., webgraphs, social networks, road networks) and plot a histogram of shortest path distances among nodes for each type of graph. In our problem context, we focus on the data objects in a dataset, instead of nodes in a graph. In addition, the distance between nodes (e.g., the number of edges, shortest path distances) are different from the distance measures (e.g., ED and DTW) in our problem.

Linardi et al. [32] explore time series datasets and plot the distribution of Euclidean distance among subsequences in time series datasets. In the geostatistics community [33], the Ripley's K function [34] is used to summarize a dataset $\mathcal{D}$ of points by the following cumulative distribution function $K(x) = c \cdot |(o_i, o_j) : o_i \in \mathcal{D}, o_j \in \mathcal{D}, dist(o_i, o_j) \leq x|$, where c is a constant and $dist(o_i, o_j)$ represents the distance function. All of above problems compute exact distance among objects, their solutions are not scalable with either billions of objects or expensive distance computations. Our solution in this work could provide a fast and accuracy guaranteed solution for the above problems. Fu et al. [5] compute the exact and approximate histogram of pairwise distances between astronomical objects. Our problem differs from [5] in two ways. First, [5] does not provide theoretical guarantee on its approximate result and it only works for Euclidean distance. Second, our proposed framework and optimization techniques have not been studied in [5].

Approximate query processing: Approximate query processing techniques are gold standards for massive data analytical applications, e.g., approximate kernel k-means [35], approximate clustering and outlier detection [36], [37], sampling cube [38], and histogram matching [39], etc. We omit the discussion of these works and refer readers to the recent overview papers [12], [40]. Unlike these works, we focus on computing the approximate cumulative distance distribution and approximate distance distribution histogram with accuracy guarantee, which is different from the above existing problem inherently. It also cannot be addressed by adapting the above proposed techniques for different problems. In big data era, sampling techniques [13], [14], [15] are widely used in approximate query processing applications. However, they do not provide worst-case guarantee on the error of the results. On the contrary, we propose a novel method to compute approximate distance distributions with error bound guarantees in this work.

8 Conclusion

In this paper, we propose the problem of computing approximate cumulative distance distribution ACDD and approximate distance distribution histogram ADDH with worst-case guarantees on the error of the result. They can be used as building bricks in various areas, e.g., data mining, geostatistics analysis, and cosmological analysis. We present a generic and scalable solution for them. We devise a suite of optimizations to improve its performance. Our experimental results confirm the efficiency and superiority of our proposal for three widely used distance measures, i.e., Euclidean distance (ED), dynamic time warping (DTW), and discrete Fréchet distance (DFD). The interesting directions for future work are: (i) devising other techniques to further speedup its performance, and (ii) supporting interactive error-bound distance distribution computation at arbitrary granularity.

Acknowledgment

Jiahao Zhang, Man Lung Yiu and Qing Li were supported by General Research Funds PolyU 152050/19E and PolyU 112114/17E from the Hong Kong Research Grants Council. Bo Tang was supported by the National Science Foundation of China (NSFC No. 61802163), the Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), the Education Department of Guangdong (Grant No. 2020KZDZX1184), and the PCL Future Greater-Bay Area Network Facilities for Large-scale Experiments and Applications (LZC0019).

References

[1] Y. Yu, L. Cao, E. A. Rundensteiner, and Q. Wang, "Detecting moving object outliers in massive-scale trajectory streams," in *KDD*, 2014, pp. 422–431.

[2] K. Heitmann, M. White, C. Wagner, S. Habib, and D. Higdon, "The coyote universe. i. precision determination of the nonlinear matter power spectrum," *The Astrophysical Journal*, vol. 715, no. 1, pp. 104–121, 2010.

[3] T. Guo, K. Feng, G. Cong, and Z. Bao, "Efficient selection of geospatial data on maps for interactive and visualized exploration," in *SIGMOD*, 2018, pp. 567–582.

[4] Y. Altuvia, P. Landgraf, G. Lithwick, N. Elefant, S. Pfeffer, A. Aravin, M. J. Brownstein, T. Tuschl, and H. Margalit, "Clustering and conservation patterns of human micrornas," *Nucleic acids research*, vol. 33, no. 8, pp. 2697–2706, 2005.

[5] B. Fu, E. Fink, G. Gibson, and J. Carbonell, "Exact and approximate computation of a histogram of pairwise distances between astronomical objects," in *Proceedings of the 2012 workshop on High-Performance Computing for Astronomy Date*, 2012, pp. 17–24.

[6] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *KDD*, 1996, pp. 226–231.

[7] U. Kang, C. E. Tsourakakis, A. P. Appel, C. Faloutsos, and J. Leskovec, "Radius plots for mining tera-byte scale graphs: Algorithms, patterns, and observations," in *SDM*, 2010, pp. 548–558.

[8] *G2. Yahoo! AltaVista Web Page Hyperlink Connectivity Graph, circa 2002*, http://webscope.sandbox.yahoo.com/.

[9] T. Bozkaya and M. Ozsoyoglu, "Distance-based indexing for high-dimensional metric spaces," in *SIGMOD*, 1997, pp. 357–368.

[10] P. Ciaccia, M. Patella, and P. Zezula, "A cost model for similarity queries in metric spaces," in *PODS*, 1998, pp. 59–68.

[11] D. Xie, F. Li, and J. M. Phillips, "Distributed trajectory similarity search," *PVLDB*, vol. 10, no. 11, pp. 1478–1489, 2017.

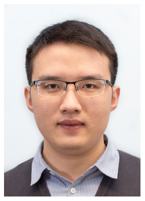
[12] S. Chaudhuri, B. Ding, and S. Kandula, "Approximate query processing: No silver bullet," in *SIGMOD*, 2017, pp. 511–519.

[13] H. Toivonen, "Sampling large databases for association rules," in *VLDB*, 1996, pp. 134–145.

[14] G. H. John and P. Langley, "Static versus dynamic sampling for data mining." in *KDD*, 1996, pp. 367–370.
[15] G. Cormode and N. G. Duffield, "Sampling for big data: a tutorial," in *KDD*, 2014, pp. 1975–1975.
[16] P. K. Agarwal, R. B. Avraham, H. Kaplan, and M. Sharir, "Computing the discrete fréchet distance in subquadratic time," *SIAM Journal on Computing*, vol. 43, no. 2, pp. 429–449, 2014.
[17] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, "Mining interesting locations and travel sequences from gps trajectories," in *WWW*, 2009, pp. 791–800.
[18] B. Tang, M. L. Yiu, K. Mouratidis, and K. Wang, "Efficient motif discovery in spatial trajectories using discrete fréchet distance," in *EDBT*, 2017, pp. 378–389.
[19] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Mining and knowledge discovery*, vol. 15, no. 2, pp. 107–144, 2007.
[20] Y. Hwang and H.-K. Ahn, "Convergent bounds on the euclidean distance," in *NIPS*, 2011, pp. 388–396.
[21] S.-W. Kim, S. Park, and W. W. Chu, "An index-based approach for similarity search supporting time warping in large sequence databases," in *ICDE*, 2001, pp. 607–614.
[22] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," in *KDD*, 2012, pp. 262–270.
[23] N. Begum, L. Ulanova, J. Wang, and E. Keogh, "Accelerating dynamic time warping clustering with a novel admissible pruning strategy," in *KDD*, 2015, pp. 49–58.
[24] K. Bringmann and W. Mulzer, "Approximability of the discrete fréchet distance," *Journal on Computational Geometry*, vol. 7, no. 2, pp. 46–76, 2016.
[25] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *SIGMOD*, 1984, pp. 47–57.
[26] *Source codes and datasets for experimental study*, 2021, http://bit.do/frdPe.
[27] Y. Li, L. H. U, M. L. Yiu, and Z. Gong, "Discovering longest-lasting correlation in sequence databases," *PVLDB*, vol. 6, no. 14, pp. 1666–1677, 2013.
[28] P. Ciaccia and M. Patella, "Pac nearest neighbor queries: Approximate and controlled search in high-dimensional and metric spaces," in *ICDE*, 2000, pp. 244–255.
[29] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas, "Predicting taxi–passenger demand using streaming data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1393–1402, 2013.
[30] H. Zhu, G. Kollios, and V. Athitsos, "A generic framework for efficient and effective subsequence retrieval," *PVLDB*, vol. 5, no. 11, pp. 1579–1590, 2012.
[31] M. Qiao, H. Cheng, L. Chang, and J. X. Yu, "Approximate shortest distance computing: A query-dependent local landmark scheme," in *ICDE*, 2012, pp. 462–473.
[32] M. Linardi, Y. Zhu, T. Palpanas, and E. J. Keogh, "Matrix profile X: VALMOD - scalable discovery of variable-length motifs in data series," in *SIGMOD*, 2018, pp. 1053–1066.
[33] P. Goovaerts, *Geostatistics for Natural Resources Evaluation*. Oxford University Press, 1997.
[34] B. D. Ripley, *Spatial Statistics*. Wiley-Interscience, 1981.
[35] R. Chitta, R. Jin, T. C. Havens, and A. K. Jain, "Approximate kernel k-means: solution to large scale kernel clustering," in *KDD*, 2011, pp. 895–903.
[36] G. Kollios, D. Gunopulos, N. Koudas, and S. Berchtold, "Efficient biased sampling for approximate clustering and outlier detection in large data sets," *IEEE TKDE*, vol. 15, no. 5, pp. 1170–1187, 2003.
[37] Q. Lin, W. Ke, J.-G. Lou, H. Zhang, K. Sui, Y. Xu, Z. Zhou, B. Qiao, and D. Zhang, "Bigin4: Instant, interactive insight identification for multi-dimensional big data," in *KDD*, 2018, pp. 547–555.
[38] X. Li, J. Han, Z. Yin, J. Lee, and Y. Sun, "Sampling cube: a framework for statistical olap over sampling data," in *SIGMOD*, 2008, pp. 779–790.
[39] S. Macke, Y. Zhang, S. Huang, and A. G. Parameswaran, "Adaptive sampling for rapidly matching histograms," *PVLDB*, vol. 11, no. 10, pp. 1262–1275, 2018.
[40] T. Milo and A. Somech, "Next-step suggestions for modern interactive data analysis platforms," in *KDD*, 2018, pp. 576–585.

Jiahao Zhang received his bachelor degree in computer science from Jilin University in 2017. He is a PhD candidate in the Department of Computing, The Hong Kong Polytechnic University. His research interest is trajectory similarity search.

Man Lung Yiu received the bachelor's degree in computer engineering and the PhD degree in computer science from the University of Hong Kong in 2002 and 2006, respectively. Prior to his current post, he worked at Aalborg University for three years starting in the Fall of 2006. He is now an associate professor in the Department of Computing, the Hong Kong Polytechnic University. His research focuses on the management of complex data, in particular query processing topics on spatiotemporal data and multidimensional data.



Bo Tang received the PhD degree in computer science from The Hong Kong Polytechnic University in 2017. He is currently an assistant professor in Southern University of Science and Technology. He was an visiting researcher at Centrum Wiskunde & Informatica and Microsoft Research Asia, respectively. His research interests include similarity search on high dimensional dataset and data exploration on multidimensional dataset.

Qing Li is a Chair Professor of Department of Computing, the Hong Kong Polytechnic University. He received his B.Eng. from Hunan University (Changsha), and M.Sc. and Ph.D. degrees from the University of Southern California (Los Angeles), all in computer science. His research interests include multi-modal data management, conceptual data modeling, social media, Web services, and e-learning systems. He has authored/co-authored over 400 publications in these areas. He is actively involved in the research community and has served as an associate editor of a number of major technical journals including IEEE Transactions on Knowledge and Data Engineering (TKDE), ACM Transactions on Internet Technology (TOIT), Data Science and Engineering (DSE), World Wide Web (WWW), and Journal of Web Engineering, in addition to being a Conference and Program Chair/Co-Chair of numerous major international conferences. He also sits/sat in the Steering Committees of DASFAA, ER, ACM RecSys, IEEE U-MEDIA, and ICWL. Prof. Li is a Fellow of IEE/IET (UK), a senior member of IEEE, and a distinguished member of CCF (China).