Fast Dataset Search with Earth Mover's Distance

Wenzhe Yang
School of Computer Science, Wuhan University
wenzheyang@whu.edu.cn

Sheng Wang*
School of Computer Science, Wuhan University
swangcs@whu.edu.cn

Yuan Sun
School of Computing and Information Systems, The University of Melbourne
yuan.sun@unimelb.edu.au

Zhiyong Peng*
School of Computer Science, Wuhan University
Big Data Institute, Wuhan University
peng@whu.edu.cn

ABSTRACT

The amount of spatial data in open data portals has increased rapidly, raising the demand for spatial dataset search in large data repositories. In this paper, we tackle spatial dataset search by using the Earth Mover's Distance (EMD) to measure the similarity between datasets. EMD is a robust similarity measure between two distributions and has been successfully applied to multiple domains such as image retrieval, document retrieval, multimedia, etc. However, the existing EMD-based studies typically depend on a common filtering framework with a single pruning strategy, which still has a high search cost. To address this issue, we propose a Dual-Bound Filtering (DBF) framework to accelerate the EMD-based spatial dataset search. Specifically, we represent datasets by Z-order histograms and organize them as nodes in a tree structure. During a query, two levels of filtering are conducted based on pooling-based bounds and a TICT bound on EMD to prune dissimilar datasets efficiently. We conduct experiments on four real-world spatial data repositories and the experimental results demonstrate the efficiency and effectiveness of our DBF framework.

PVLDB Reference Format:
Wenzhe Yang, Sheng Wang, Yuan Sun, Zhiyong Peng. Fast Dataset Search with Earth Mover's Distance. PVLDB, 15(11): 2517 - 2529, 2022.
doi:10.14778/3551793.3551811

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/yangwenzhe/DBF.

1 INTRODUCTION

Recently, many dataset search engines have been proposed to support data discovery [8, 11, 13, 19, 25, 30, 48], such as Google Dataset Search [9], Dataverse [16], and Aurum [20]. These dataset search engines typically require users to specify query conditions. However, in many cases, users may not be accustomed to the query language or do not know how to describe the data of interest [14]. To tackle this, the exemplar query has been proposed [28, 29], which directly takes an example data of interest as input to search for relevant data in a database.

There has been a growing interest in exemplar dataset search recently [11, 19, 32, 37, 47, 57, 59]. For example, in the Auctus [11] search system, users can use the bicycle usage data for one month as an exemplar dataset to retrieve relevant datasets for other months, in order to build a more robust model for prediction tasks, e.g., predicting the number of bicycle trips based on the daily temperature features. In RONIN [32], users can use an example dataset related to smart city to search for additional datasets that are joinable based on the set containment between datasets.

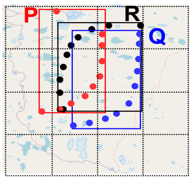
As an essential part of real-world datasets, spatial datasets have gained increasing popularity throughout the world. Hahmann and Burghardt [24] show that at least 60% of open government datasets are georeferenced, and spatial dataset search has been a core function in open government data portals [18, 26, 46]. In this paper, we focus on exemplar query for spatial dataset search, which has many important real-world applications. For example, in autonomous driving motion forecasting, a data analyst can obtain relevant driving datasets with a given query via spatial dataset search, in order to improve the forecasting accuracy (see Sec 5.3 in [12]). In tourism trip planning, a tourist can search for alternative routes that are similar to the current one, allowing more flexibility for travel [47, 49]. In addition, spatial dataset search can also be used for data imputation, by searching for relevant datasets in a database to impute the missing values in a given spatial dataset [11, 58].

In the above applications of spatial dataset search, how to effectively measure the similarity between datasets is a core challenge. One approach to measure similarity is utilizing the overlapping area of the minimum bounding rectangle (MBR) that covers all points in datasets [18, 22, 46]. Figure 1 shows an example of spatial dataset search with a query dataset Q and two candidate datasets P and R. In Figure 1(a), the red, blue, and black boxes represent the MBRs of dataset P, Q, and R, respectively. We can see that although R (compared with P) is less similar to Q, the overlapping area between R and Q is greater than that between P and Q. This indicates that the MBR-based similarity measure is not reliable.

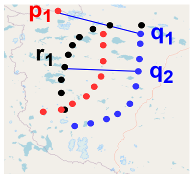
Another approach is to compute the distance among points in the spatial datasets [1, 18, 31], e.g., the Hausdorff distance [31, 36, 43] that computes the maximum nearest neighbor distance between pairs of points. However, the Hausdorff distance is sensitive to outliers. As shown in Figure 1(b), the nearest neighbor distances of most points between P and Q are small, but the nearest neighbor distance of the outlier p_1 is large. The outlier p_1 results in the Hausdorff distance between P and Q larger than that between R

Proceedings of the VLDB Endowment, Vol. 15, No. 11 ISSN 2150-8097.
doi:10.14778/3551793.3551811

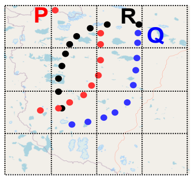
*Corresponding authors



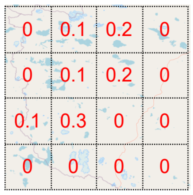
(a) MBR



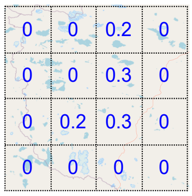
(b) Hausdorff distance



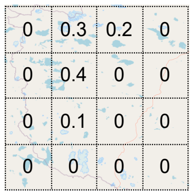
(c) Space partition



(d) P's distribution



(e) Q's distribution



(f) R's distribution

Figure 1: Given a query dataset Q and two candidate datasets P and R, we want to identify which one is more similar to Q. Although Q and R have a larger overlapping area and smaller Hausdorff distance, most of the points in P are closer to Q.

and Q. Moreover, as the Hausdorff distance has a quadratic time complexity [31] regarding the number of points in the dataset, it is slow for searching in large-scale datasets containing millions of points. In addition, as the Hausdorff distance needs to compute the nearest neighbor distance of each point in the dataset, it faces the problem of privacy restrictions in the scenario, where data owners do not allow third-parties to access their specific location information for avoiding privacy leakage [10].

Hence, it is advantageous to compress the original dataset as a distribution [10, 15, 41, 55, 56] and measure the similarity between datasets via the Earth Mover's Distance (EMD), which is one of the effective distance metrics between distributions [7, 38, 39, 41, 50]. An example is illustrated in Figure 1(c), where we divide the whole space into a grid and count the density of points in each entry of the grid for each dataset (see Figures 1(d), 1(e) and 1(f)). The EMD, also known as Wasserstein-1 distance, is defined as the minimum cost to transform one distribution into the other, where the cost is the sum of the product of the moved mass and distance. EMD is widely adopted by real-world applications, including image retrieval [4, 7, 39, 44, 51], machine learning [2], deep learning [3, 57] and natural language processing [5, 27], etc.

Compared with the overlapping area and Hausdorff distance, EMD [38, 39] is a more reliable measure to identify the similarity between two datasets. However, it has $O(n^3 \log n)$ computational complexity, where n is the number of histogram bins. Most of the existing studies utilize lower bounds of EMD to implement a scan-and-refinement (SAR) search algorithm [4–6, 27, 33, 39, 45, 51]. For top-k search, the SAR algorithm firstly computes a lower bound on the EMD between the query dataset and each of the candidate datasets, and generates a ranking for the datasets based on their lower bounds. The algorithm then computes the exact EMD between the query dataset and the top k datasets according to the ranking. For each of the remaining datasets (rank $< k$), if its lower bound on EMD is greater than the EMD of the current k-th nearest neighbor, the dataset can be pruned directly; otherwise, the EMD between the dataset and the query dataset needs to be computed.

The above studies have high filtering costs since they require the computation of a lower bound for each of the datasets in the data repository. Another approach combines lower bounds with index structure, such as B$^+$-tree [53] and quadtree [40]. These methods can prune datasets in batch and obtain the candidate datasets (i.e., datasets not pruned) by computing the lower bound between the tree nodes and the query dataset. Then, they compute the exact EMD for each candidate dataset and identify the top-k datasets with the smallest EMD distance. Although these methods can quickly obtain the candidate set, they are limited with the filtering ratio since they need to compute the exact EMD for each candidate.

We propose a Dual-Bound Filtering (DBF) framework to improve the filtering ratio, which successfully combines the index-based filtering and SAR search. Specifically, we first propose a novel Z-order histogram to model spatial dataset. During a query, we combine a new tree-structured index and pooling-based bounds to filter datasets in batch. Then, we propose a *Tighter Iterative Constrained Transfers* (TICT) bound to conduct SAR search, which can further reduce the number of candidate datasets. Overall, the contributions of this paper are summarized as follows:

- We propose DBF, a dual filtering framework composed of two-level filtering modules from coarse-grained to fine-grained to support similarity search based on spatial dataset density distribution (see Section 4).
- We propose a novel data modeling approach named Z-order histogram, which not only saves much storage space but can also be applied to different resolutions through an efficient pooling operation (see Section 5).
- To improve the search efficiency, we design an efficient tree-structured index and deduce pooling-based bounds and the TICT bound to improve the filtering ratio (see Section 6).
- Our experimental results on four spatial data repositories show the superiority of our DBF framework (see Section 7).

2 RELATED WORK

2.1 Dataset Similarity Search

The dataset similarity search can be broadly divided into textual and numeric dataset similarity searches. For the first type of search, the most commonly used similarity measures include Jaccard similarity and containment [8, 21], cosine similarity [32, 58], and set intersection size [52, 59], etc. For example, Ouellette et al. [32] used cosine similarity to measure the similarity between two semantic vectors extracted from datasets. Fernandez et al. [21] proposed a LAZO method to simultaneously estimate the Jaccard similarity and containment between two columns of data. Zhu et al. [59] proposed a JOSIE algorithm to find the joinable column in the database with the given column of a table by using the set intersection size as a similarity measure.

For numeric datasets, the similarity search is based on the distance or overlapping area between datasets. The spatial dataset is an important sub-category of the numeric dataset. The commonly used similarity measures mainly includes MBR [22] and Hausdorff distance [35], etc. For example, Nutanong et al. [31] proposed to apply the Hausdorff distance as a similarity measure between two datasets and derived upper and lower bounds by the distance between MBRs of two datasets. Adelfio et al. [1] proposed an improved lower bound of Hausdorff distance called ENHLB and an R-tree [23] index structure to accelerate the spatial similarity search. Vasconcelos et al. [46] proposed the spatial similarity based on the overlapping area of the bounding box containing all data to improve spatial search in the Brazilian open government data portal. Degbelo and Teka [18] compared the impact of utilizing Hausdorff distance and area of overlap on spatial similarity search efficiency. Hervey et al. [26] studied the state of search on open geospatial data portals and evaluated the functionality of their search engines.

However, the existing methods are not well suited for spatial dataset search. First, the above textual dataset measures are not applicable for spatial datasets, which are numeric. Second, in the existing spatial dataset similarity measures, MBR is too coarse-grained and does not consider how data distributes in the bounding boxes. Meanwhile, the Hausdorff distance is inherently sensitive to outliers and potentially leaks the privacy of individuals. To overcome these shortcomings of the previous methods, we will apply EMD to measure the spatial dataset similarity, which is robust and effective in measuring the distance between two distributions.

2.2 EMD-based Similarity Search

Due to the cubic complexity of EMD computation, it is challenging to apply EMD, especially in search scenarios that require fast responses. Numerous lower and upper bounds of EMD have been proposed to accelerate the process of EMD-based similarity search. For example, Rubner et al. [39] proposed a centroids-based lower bound of EMD. Wichterich et al. [51] deduced a dimensionality reduction-based lower bound of EMD. Tang et al. [44] proposed a dynamic lower bound in the process of exact EMD computation, which is used to terminate an EMD refinement early. Kusner et al. [27] proposed a lower bound by removing the out-flow or in-flow constraint. Uysal et al. [45] proposed an IM-Sig* lower bound by relaxing the in-flow constraint of the EMD. Atasu and Mittelholzer [5] proposed an Iterative Constrained Transfers (ICT) lower bound based on the greedy strategy. However, these methods have high filtering costs during candidate generation.

Assent et al. [4] constructed a 3D index based on the weighted lower bound, which can only handle the 3D color space in the image domain. Ruttenberg and Singh [40] proposed a new lower bound of EMD by projecting each dataset distribution onto a vector and designed a new quadtree-based index structure. Xu et al. [53] proposed lower and upper bounds utilizing the primal-dual theory and constructed a forest of B^+-trees [23] based on one of the feasible solutions of EMD. We summarize several representative EMD studies in Table 1. To sum up, the previous applications of EMD are mainly in the fields of image [39, 44, 51], document [5, 27], and video [45], but it has never been applied to the spatial dataset search. In addition, the existing methods implemented lower bound-based SAR search or index-based filtering; both are single-layer pruning and are limited by the filtering ratio. Thus, it is necessary to design an effective dual filtering framework for spatial datasets to improve the filtering ratio.

3 PRELIMINARIES

3.1 Earth Mover's Distance

Given two histograms **q** and **p**, where **q** has m bins and **p** has n bins, i.e. $\mathbf{q} = \{q_1, q_2, \dots, q_m\}$ and $\mathbf{p} = \{p_1, p_2, \dots, p_n\}$. The total density

Table 1: An overview of existing EMD studies.

Name	Complexity	Bound	Dataset	Index
$LB_{centroid}$ [39]	$O(n)$	LB	Image	No
LB_{IM} [4]	$O(n^2)$	LB	Image	3D-index
Red [51]	$--$	LB	Image	No
Normal [40]	$O(n)$	LB	Image	Quadtree
dualEMD [53]	$--$	LB, UB	Image	B^+-tree
SIA [44]	$O(n^2)$	LB,UB	Image	No
RWMD [27]	$O(n^2)$	LB	Document	No
*IM-Sig** [45]	$O(n^2 \log n)$	LB	Video	No
ICT [5]	$O(n^2 \log n)$	LB	Document	No

of **q** and **p** are normalized to one, i.e., $\sum_{i=1}^{m} q_i = \sum_{j=1}^{n} p_j = 1$.[1] We denote a flow matrix as **F**, where $f(i,j)$ indicates the flow to move from q_i to p_j, and a distance matrix **D**, where $d(i,j)$ is the ground distance between q_i and p_j. The optimization goal of the EMD problem is to find the optimal flow that minimizes the overall cost of moving **q** into **p**. The EMD can be formally defined as:

$$EMD(\mathbf{q}, \mathbf{p}) = \min \sum_{i=1}^{m} \sum_{j=1}^{n} f(i,j) d(i,j) \tag{1}$$

subject to the following constraints:

$$\forall i \in [1, m] : \quad \sum_{j=1}^{n} f(i,j) = q_i \tag{2}$$

$$\forall j \in [1, n] : \quad \sum_{i=1}^{m} f(i,j) = p_j \tag{3}$$

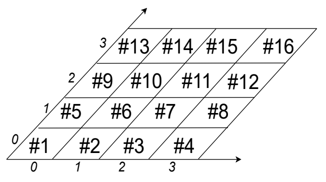
$$\forall i \in [1, m], j \in [1, n] : \quad f(i,j) \geq 0 \tag{4}$$

where the out-flow constraint (2) ensures that the total flow exiting i is equal to q_i, the in-flow constraint (3) ensures that the total flow entering j is equal to p_j, and the non-negative constraint (4) ensures that the flow is from q_i to p_j.

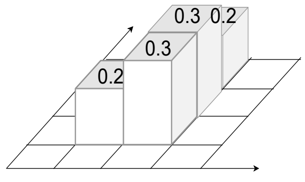
Next, we illustrate the computation process of EMD via a toy example in Figure 2, where the 2D space is divided into a 4×4 grid (namely 16 bins)[2], and the numbers 0 to 3 at the horizontal and vertical axes represent the position in each dimension, and the numbers 1 to 16 represent the ID of bins. The distributions of datasets can be represented by histograms, e.g., Figures 2(b), 2(c) and 2(d) show the histograms of the three datasets in Figure 1(c). To simplify the computation, here we use the Manhattan distance [53] as the ground distance, e.g., the distance from q_7 to q_5 is $|3-1| + |2-2| = 2$.

In Figures 3(a) and 3(b), we present the optimal flows from **q** to **p** and **q** to **r**, respectively. Based on the optimal flow in Figure 3(a), we can compute $EMD(\mathbf{q}, \mathbf{p}) = 0.1 \times 2 + 0.1 + 0.1 \times 2 + 0.1 \times 2 = 0.7$. Similarly, based on Figure 3(b), we can compute $EMD(\mathbf{q}, \mathbf{r}) = 0.1 \times 1 + 0.3 \times 2 + 0.3 \times 2 = 1.3$. As $EMD(\mathbf{q}, \mathbf{p}) < EMD(\mathbf{q}, \mathbf{r})$, the dataset P is identified as more similar to Q, which is consistent with the true result. In contrast, using the MBR or Hausdorff distance as the similarity measure would lead to the wrong result (see Figure 1).

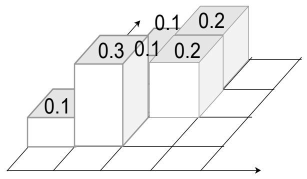
[1]Note that the EMD holds the non-negativity, symmetry and triangle inequality when the histograms have equal density [39].
[2]The entry in the grid corresponds to the bin of the histogram.



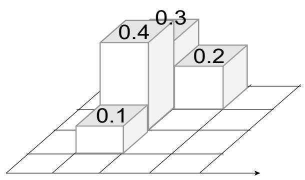
(a) The ID of entries in the grid



(b) Histogram **q** of dataset Q

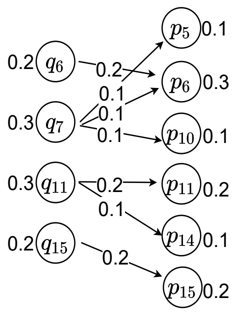


(c) Histogram **p** of dataset P



(d) Histogram **r** of dataset R

Figure 2: Examples of dataset density distributions in the form of histograms.


(a) Optimal flow from **q** to **p**

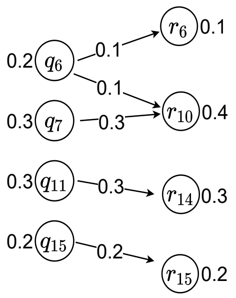

(b) Optimal flow from **q** to **r**

Figure 3: The optimal flows of EMD computation.

Table 2: Summary of notations.

Symbol	Description
D, $\mathcal{D}$	a dataset and a data repository
$\lvert D\rvert$	the number of spatial points in D
N_Q, N_P	the nodes of spatial dataset Q and P
$\mathbf{q}$, $\mathbf{p}$	the Z-order histograms of Q and P
$f(i,j)$	the density flow from q_i to p_j
$d(i,j)$	the ground distance between q_i and p_j
h, w	the height and width of the whole 2D space
μ, ν	the height and width of each entry in the grid
2^θ	the number of entries in each dimension
(x,y)	the coordinate of point
$\rho_{i,j}$	the dataset density in the i-th row j-th column entry
$\sigma_{i,j}$	the number of points falling into the i-th row j-th column entry
$\zeta(\mathbf{q},\mathbf{q}')$	the pooling cost between $\mathbf{q}$ and $\mathbf{q}'$
$\gamma(q)$	the radius of dataset node N_Q

This indicates that EMD is a more robust similarity measure than the MBR and Hausdorff distance.

3.2 Problem Definitions

Definition 1. ***(Spatial Dataset).*** *A spatial dataset D contains a set of points marked with spatial locations, i.e., $D = \{(x_1,y_1),(x_2,y_2),\ldots,(x_t,y_t)\}$, $|D| = t$ is the total number of points in this dataset.*

Definition 2. ***(Spatial Data Repository).*** *A spatial data repository $\mathcal{D}$ contains a set of spatial datasets, i.e., $\mathcal{D} = \{D_1, D_2, \ldots, D_n\}$, $|\mathcal{D}| = n$ is the scale of this repository.*

Definition 3. ***(Spatial Dataset Distribution).*** *The whole space containing all spatial datasets is divided into a $2^\theta \times 2^\theta$ grid. The coordinates and densities of all non-zero entries in the grid jointly constitute a histogram $\mathbf{q} = \{q_1, q_2, \ldots, q_m\}$, and the sum of all densities is equal to 1, i.e., $\sum_{i=1}^{m} q_i = 1$.*

Definition 4. ***(Top-k EMD Dataset Search).*** *Given a query dataset Q, a top-k EMD dataset search returns k datasets from $\mathcal{D}$, which have the minimum EMDs with Q.*

4 OVERALL FRAMEWORK

In this section, we introduce a Dual-Bound Filtering (DBF) framework to answer the top-k EMD dataset search. As shown in Figure 4, DBF is composed of four modules described as follows.

Data Modeling. For the 2D space containing all spatial datasets (see Figure 4(a)), we first divide it into a $2^\theta \times 2^\theta$ grid to preserve the spatial location information instead of partitioning each spatial dataset individually (see Figure 4(b)), where θ is called resolution. Then, we compute the density of the dataset in each entry of the grid. When storing the spatial distribution of each dataset, we propose a compact data structure called Z-order histogram to reduce storage space. More details of the Z-order histogram will be described in Section 5.1.

Index Construction. We construct a new tree index structure for all datasets (see Figure 4(c)). We first convert each dataset into a Z-order histogram with only one non-zero entry by the Z-order pooling method and obtain a pooling cost caused by density transfer. Then, we treat each dataset as a node with radius and center. Finally, we create a new tree-structured index based on all dataset nodes. (see Section 6.1 for details).

Coarse-grained Filtering. As shown in Figure 4(d), after constructing the index, we traverse the index tree from the root node one by one in a depth-first manner and compute their pooling-based lower and upper bounds (see Section 6.2). Then, we take the branch-and-bound search to prune those tree nodes whose lower bound is greater than the maximum upper bound of unfiltered nodes. After the traversal process is completed, all datasets contained in the unfiltered nodes form our first-round candidate set. More details are introduced in Section 6.3.

Fine-grained Filtering. As shown in Figure 4(e), after obtaining the candidate datasets, we propose a TICT bound based on the ICT lower bound [5] to improve the filtering ratio. We perform a scan and refinement (SAR) search over the first-round candidate set. Those candidates whose TICT bound exceeds the exact EMD of the k-th closest dataset are directly filtered out. More details are introduced in Section 6.4.

5 DATA MODELING

To apply EMD to spatial dataset similarity search, we divide the 2D spatial space into grids and represent the distribution of a spatial dataset by the densities of data points in the grid.

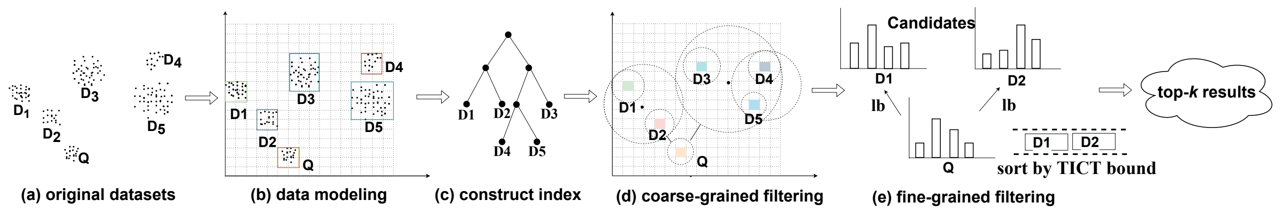


Figure 4: An overview of our proposed Dual-Bound Filtering framework.

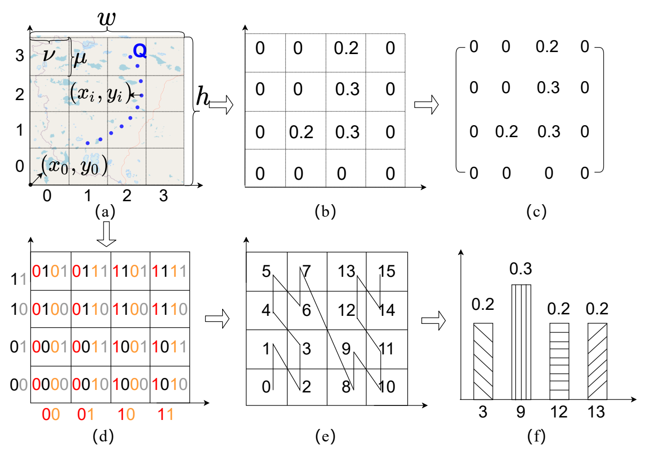
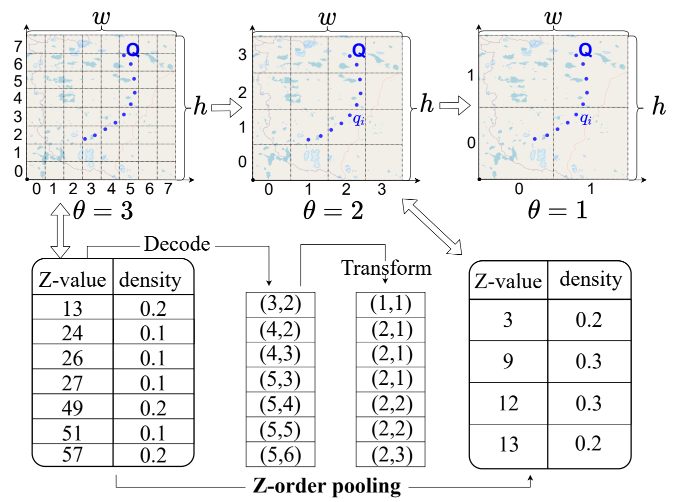


Figure 5: A comparison of matrix and Z-order histogram.



Z-value	density
13	0.2
24	0.1
26	0.1
27	0.1
49	0.2
51	0.1
57	0.2

(3,2)	(1,1)
(4,2)	(2,1)
(4,3)	(2,1)
(5,3)	(2,1)
(5,4)	(2,2)
(5,5)	(2,2)
(5,6)	(2,3)

Z-value	density
3	0.2
9	0.3
12	0.3
13	0.2

Figure 6: The diagram of Z-order pooling method.

5.1 Z-order Histogram

In the process of generating the Z-order histogram, we first initialize a 2D space based on the position (x_0, y_0), where the size of space is $h \times w$. Then, we divide the space into a $2^\theta \times 2^\theta$ grid, and hence the height and width of each entry of the grid are $\mu = \frac{h}{2^\theta}$ and $\nu = \frac{w}{2^\theta}$. As shown in Figure 5(a), the 2D space is divided into a 4×4 grid. Assuming that the position of a point in dataset Q is (x_i, y_i), then its corresponding entry coordinate is defined as:

$$X = \lfloor \frac{x_i - x_0}{\nu} \rfloor, Y = \lfloor \frac{y_i - y_0}{\mu} \rfloor, (x_i, y_i) \in D$$

The density of each entry is calculated by counting the number of points falling into each entry and dividing the counts by the total number of points in D. The density computation formula of the spatial dataset D in each entry is defined as:

$$\rho_{i,j} = \frac{\sigma_{i,j}}{|D|}, i \in [1, 2^\theta], j \in [1, 2^\theta]$$

where $\sigma_{i,j}$ denotes the number of points falling into the entry of i-th row and j-th column, and $|D|$ represents the total number of points in D.

As shown in Figure 5(b), each spatial dataset distribution can be regarded as a matrix, and hence a baseline approach to store the density distribution of a spatial dataset is to use a matrix [42]. However, in Figure 5(c), we can see that the matrix is sparse, which means most of the entries have a value of zero. Thus, much space is wasted if we use a matrix for storage. We present a compact data structure called Z-order histogram to store the one-dimensional distribution of each dataset, which saves memory by storing only the non-zero entries. The Z-order histogram is formally defined as:

Definition 5. ***(Z-order histogram).*** *A Z-order histogram of a spatial dataset $Z = \{(z_1, \rho_1), (z_2, \rho_2), \ldots, (z_m, \rho_m)\}$, where z_i and ρ_i are the Z-value and density of the i-th non-zero entry, respectively, and m is the number of non-zero entries in the grid.*

As shown in Figure 5(d), we map the coordinate of each entry in the grid to an integer named Z-value by the Z-order curve [34, 55]. Specifically, the entry coordinate is first converted to a binary representation. Then, we interlace the bits of the binary representation of coordinate to generate a combined binary string. Finally, the resulting binary string can be interpreted as an integer to identify the entry uniquely (see Figure 5(e)). After obtaining the density distribution, we use a hashmap to store the Z-value and density of all non-zero entries (see Figure 5(f)).

5.2 Z-order Pooling

When modeling the spatial dataset, if using a high resolution, the whole space is divided too fine-grained, so the number of histogram bins will increase, which also increases the time cost of EMD computation. On the contrary, if the resolution is too low, it will reduce the accuracy of the similarity search. Therefore, the choice of resolution is critical. To obtain an appropriate resolution, we divide the whole space according to the sampling distance of the entries. In addition, we also investigate how to dynamically choose an appropriate resolution based on the trade-off between query time and accuracy. The details on resolution selection are shown in the appendix of technical report [54]. As shown in Figure 6, we first obtain a fine-grained Z-order histogram under a resolution θ_1 based on the distance sampling.

Algorithm 1: `PoolingCost`(Q)

```
Input: $Q$: dataset
Output: $\zeta$: pooling cost
1 $\mathbf{q} \leftarrow$ Generate the Z-order histogram of dataset $Q$ ;
2 $q_i \leftarrow$ Find the maximum $q_i$ from $\mathbf{q} = \{q_1, q_2, \dots q_m\}$;
3 $\zeta \leftarrow 0$ ;
4 foreach $q_j \in \boldsymbol{q}$ and $j \neq i$ do
5     $f \leftarrow q_i$;
6     $d \leftarrow dist(q_i, q_j)$;
7     $\zeta \leftarrow \zeta + f \times d$;
8     $q_j \leftarrow 0$;
9 return $\zeta$;
```

However, it would be time-consuming to scan all datasets in the data repository and compute their density distributions if the resolution obtained need to be changed during the data modeling. We present a novel Z-order pooling method to obtain Z-order histograms under various resolutions specified by users without rescanning the datasets. To obtain the distribution under a lower resolution θ_2, we decode the Z-value at the resolution θ_1 into the corresponding coordinate (X_1, Y_1). We then use the formulas below to transform (X_1, Y_1) into the new coordinate (X_2, Y_2) at the resolution θ_2:

$$X_2 = \lfloor \frac{X_1}{2^{(\theta_1-\theta_2)}} \rfloor, Y_2 = \lfloor \frac{Y_1}{2^{(\theta_1-\theta_2)}} \rfloor$$

Next, we encode X_2 and Y_2 into a new Z-value and accumulate the density of the entries with the same Z-value. By doing so, we obtain a new Z-order histogram under θ_2. Compared with the repeated scanning of the dataset after the resolution is changed, the Z-order pooling method dramatically reduces the modelling time.

6 INDEX STRUCTURE AND PRUNING MECHANISMS

In this section, we design a new index structure and derive pooling-based lower and upper bounds and a TICT bound based on the ICT lower bound [5] to support our efficient dual-bound filtering framework.

6.1 Index Construction

As discussed in Section 5.2, we can decrease the resolution to generate fewer entries in the grid. For example, in Figure 6, we can generate a 2×2 or 4×4 grid from the 8×8 grid by the pooling operation. Due to the high time complexity of the EMD, it is faster to compute the exact EMD for two Z-order histograms with fewer bins. We transform the original Z-order histogram of each spatial dataset into a histogram with only one non-zero entry by using the Z-order pooling method. Thus, the exact EMD between two datasets equals the ground distance between two non-zero entries. There is a transfer cost generated in Z-order pooling, which is crucial for deriving the pooling-based lower and upper bounds.

Algorithm 1 shows the specific computation process of the Z-order pooling cost. Specifically, for a given dataset Q, we first transform the dataset Q into a Z-order histogram $\mathbf{q}$ (see Line 1). Then, we find the entry with the highest density (see Line 2). Subsequently, we transfer the densities of other entries into it and generate transfer costs (see Line 4 to Line 8).

Complexity Analysis. For Algorithm 1, the time cost of generating a Z-order histogram is linear in terms of the number of entries in a grid. In other operations (from Line 2 to Line 9), the time cost is linear in terms of the number of bins in $\mathbf{q}$, since we need to traverse all bins in $\mathbf{q}$.

After pooling all datasets in the data repository, each dataset can be treated as a node whose center c is the coordinate of the entry with the highest density before pooling operation and radius γ is the pooling cost. We further create an index structure in a top-bottom way based on these dataset nodes. Firstly, we get the root node based on all dataset nodes, whose center is the centroid of all dataset nodes. Since each dataset node has a radius, in order to cover all dataset nodes, the radius of the root node is the maximum of the ground distance from the root to the dataset node plus the pooling cost of the dataset node. Then, we select two furthest nodes among all the dataset nodes. We divide the root node into the left child node and right child node according to the ground distance of the remaining nodes to the two nodes. Finally, we repeat the above procedure until the leaf node satisfies capacity constraint or the depth of the tree satisfies maximum value.

6.2 Pooling-Based Upper and Lower Bounds

We can deduce the lower and upper bounds of $EMD(\mathbf{q}, \mathbf{p})$ by using the Z-order pooling and the triangle inequality for two histograms $\mathbf{q}$ and $\mathbf{p}$. We first prove that the pooling cost $\zeta(\mathbf{q}, \mathbf{q}')$ equals the $EMD(\mathbf{q}, \mathbf{q}')$ in the following lemma, where $\mathbf{q}'$ denotes the Z-order histogram after pooling operation.

Lemma 6.1. *The $\zeta(\boldsymbol{q}, \boldsymbol{q}')$ equals the EMD between $\boldsymbol{q}$ and $\boldsymbol{q}'$, i.e., $\zeta(\boldsymbol{q}, \boldsymbol{q}') = EMD(\boldsymbol{q}, \boldsymbol{q}')$.*

A detailed proof of Lemma 6.1 is given in the appendix of technical report [54]. Then, we propose the following lower and upper bounds based on the triangle inequality.

Lemma 6.2. *Let $\boldsymbol{q}$ and $\boldsymbol{p}$ be two Z-order histograms of equal total density. It holds:*

$$EMD(\boldsymbol{q}, \boldsymbol{p}) \leq EMD(\boldsymbol{q}', \boldsymbol{p}') + \zeta(\boldsymbol{q}, \boldsymbol{q}') + \zeta(\boldsymbol{p}, \boldsymbol{p}') \tag{5}$$

$$\begin{aligned} EMD(\boldsymbol{q}, \boldsymbol{p}) \geq \min\{&||EMD(\boldsymbol{q}', \boldsymbol{p}') - \zeta(\boldsymbol{q}, \boldsymbol{q}')| - \zeta(\boldsymbol{p}, \boldsymbol{p}')|, \\ &||EMD(\boldsymbol{q}', \boldsymbol{p}') - \zeta(\boldsymbol{p}, \boldsymbol{p}')| - \zeta(\boldsymbol{q}, \boldsymbol{q}')|\} \end{aligned} \tag{6}$$

A detailed proof of Lemma 6.2 is given in the appendix of technical report [54]. We use the density distributions of Q and P to illustrate the pooling-based lower and upper bounds. In Figure 7, we show the density transfer and the position of the entry with the highest density. Based on Algorithm 1, we can compute the pooling cost $\zeta(\mathbf{q}, \mathbf{q}') = 0.2 \times 2 + 0.3 \times 1 + 0.2 \times 1 = 0.9$ and $\zeta(\mathbf{p}, \mathbf{p}') = 0.1 \times 1 + 0.1 \times 1 + 0.1 \times 2 + 0.2 \times 2 + 0.2 \times 3 = 1.4$. Since the histogram of each dataset has only one non-zero entry after pooling operation, the EMD between $\mathbf{q}'$ and $\mathbf{p}'$ is easier to compute, which equals the ground distance between two non-zero entries of $\mathbf{q}'$ and $\mathbf{p}'$, i.e., $EMD(\mathbf{q}', \mathbf{p}') = 1$. By using Lemma 6.2, we can compute $EMD(\mathbf{q}, \mathbf{p}) \geq \min\{||1-0.9|-1.4|, ||1-1.4|-0.9|\} = \min\{1.3, 0.5\} = 0.5$, and $EMD(\mathbf{q}, \mathbf{p}) \leq 1+0.9+1.4 = 3.3$. As illustrated in Section 3.1 (see Figure 3(a)), we have computed $EMD(\mathbf{q}, \mathbf{p}) = 0.7$, which is in the range of 0.5 and 3.3.

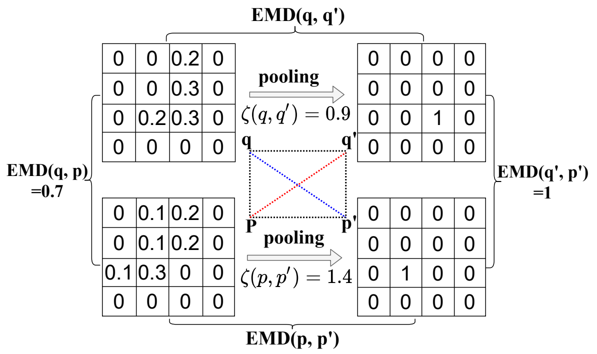


Figure 7: Pooling-based lower and upper bounds between two close datasets.

In this example, the lower bound is tight while the upper bound is quite loose. In the case that two datasets are far apart in the space (the pooling cost is much lower than the ground distance of two non-zero entities after pooling operation), the upper and lower bounds are expected to be tight. According to the unique characteristic, our pooling-based upper and lower bounds can be well applied in spatial dataset search, where it can quickly prune those datasets that are far apart in the space. In the next section, we will further introduce how to conduct pruning in batch.

6.3 Coarse-grained Batch Filtering

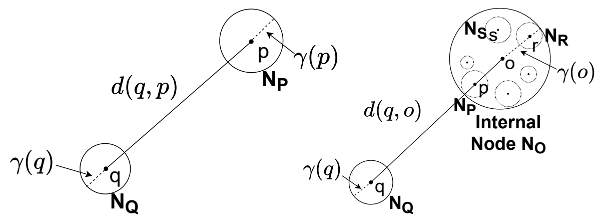


Figure 8: The lower and upper bounds between two nodes.

After constructing the index, we design corresponding pruning strategies using the pooling-based lower and upper bounds. As shown in the left of Figure 8, N_Q and N_P denote two dataset nodes generated by the pooling operation, $\gamma(q)$ and $\gamma(p)$ denote the radii of N_Q and N_P, and $d(q, p)$ denotes the ground distance between centers q and p. We have the following lemma:

Lemma 6.3. *The EMD between dataset nodes N_Q and N_P can be bounded in the range:*

$$EMD(N_Q, N_P) \le d(q,p) + \gamma(q) + \gamma(p)$$
$$EMD(N_Q, N_P) \ge \min\{||d(q,p) - \gamma(q)| - \gamma(p)|,\ ||d(q,p) - \gamma(p)| - \gamma(q)|\}$$

We can further derive the lower and upper bounds between the query node and the internal nodes of the tree for conducting filtering in batch. As shown in the right of Figure 8, N_O represents an internal tree node, which contains a set of dataset nodes $\{N_P, N_R, \dots, N_S\}$. The radii of these dataset nodes are denote as $\{\gamma(p), \gamma(r), \dots, \gamma(s)\}$. The radius of N_O equals the maximum of distance from the center o to the center of dataset node plus the radius of the dataset node, i.e., $\gamma(o) = \max\{d(o,p) + \gamma(p), d(o,r) + \gamma(r), \dots, d(o,s) + \gamma(s)\}$. Let $\gamma(o)$ denote the radius of N_O, and $d(q, o)$ denote the ground distance between q and o. We have the following lemma, and the detailed proof of this lemma is given in the appendix of technical report [54]:

Lemma 6.4. *The EMD between the dataset node N_Q and the internal node N_O can be bounded in the range:*

$$EMD(N_Q, N_O) \le d(q,o) + \gamma(q) + \gamma(o)$$
$$EMD(N_Q, N_O) \ge \min\{||d(q,o) - \gamma(q)| - \gamma(o)|,\ ||d(q,o) - \gamma(o)| - \gamma(q)|\}$$

In Algorithm 2, we display the candidate generation process of index-based batch filtering in detail. We first initialize a null priority queue to store all unfiltered leaf nodes (see Line 2). Then, we apply the `BranchAndBound` Function from the root node (see Line 5). In the pruning process, we check if the tree node N_O is a leaf node. If so, we compute the lower and upper bounds by Lemma 6.3. If the lower bound is greater than the current maximum upper bound in the priority queue, we can directly prune node N_O (see Line 14 to Line 19); otherwise, we insert node N_O to the priority queue and delete those nodes whose lower bound is greater than the upper bound of N_O (see Line 20 to Line 28).

If the node N_O is an internal node, we compute the lower and upper bounds by Lemma 6.4. Similarly, if the lower bound is greater than the current maximum upper bound in the priority queue, we can directly prune node N_O and its sub-tree (see Line 31 to Line 32); otherwise, we recursively perform `BranchAndBound` function for the left and right child nodes of N_O (see Line 33 to Line 36). After recursively traversing the tree, we obtain a priority queue containing all unfiltered leaf nodes. The dataset nodes contained in the unfiltered leaf nodes form the candidate set (see Line 6 to Line 10).

6.4 Fine-grained Filtering

To get a lower bound of EMD, Atasu and Mittelholzer [5] studied a relaxed problem, which removes the in-flow constraint and require that the flow $f(i,j) \le \min\{q_i, p_j\}$ on the basis of satisfying the out-flow constraint (2). To highlight that we use density transfer cost to measure the spatial dataset similarity, we refer to this relaxed problem as the Relaxed Density Mover's (RDM) problem. Since the out-flow constraint is equivalent to $0 \le f(i,j) \le p_j$, the RDM problem is formally defines as:

Definition 6. *Let **q** and **p** be two Z-order histograms, the RDM is formally defined as:*

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} f(i,j) d(i,j)$$

subject to:

$$\forall i \in [1,m] : \sum_{j=1}^{n} f(i,j) = q_i \tag{7}$$
$$\forall i \in [1,m], j \in [1,n], 0 \le f(i,j) \le p_j \tag{8}$$

The constraint (7) requires that the total flow existing i is equal to q_i, and the constraint (8) requires that the flow $f(i,j)$ from q_i to p_j cannot exceed the density of p_j.

For the relaxed EMD problem, Atasu and Mittelholzer [5] proposed a Iterative Constrained Transfers (ICT) algorithm to find the minimum-cost (optimal) flow. The key idea is to find the optimal

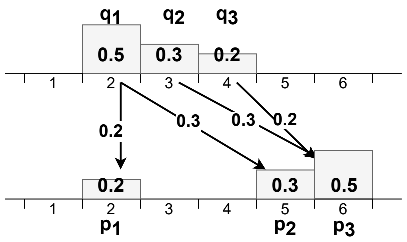


Figure 9: EMD flow.

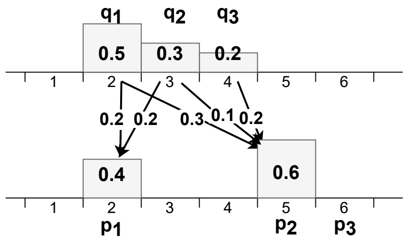


Figure 10: ICT flow.

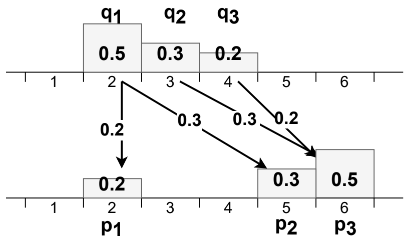


Figure 11: TICT flow.

Algorithm 2: CandidatesGen($Q, \mathcal{D}$)

```
Input: $Q$: query dataset, $\mathcal{D}$: dataset repository
Output: $C$: candidate set
1  $T \leftarrow$ Create the tree for $\mathcal{D}$ ;
2  $PQ \leftarrow$ Initialize a priority queue;
3  $maxLB, maxUB \leftarrow \infty$ ;
4  $N_Q \leftarrow$Generate the dataset node for $Q$;
5  BranchAndBound($N_Q, T.root, PQ, maxLB, maxUB$);
6  while $PQ.nonEmpty()$ do
7      $N_O \leftarrow PQ.Dequeue()$;
8      $List \leftarrow$ All dataset nodes included in $N_O$;
9      foreach $l \in List$ do
10         $C.add(l)$;
11 return $C$;
12 Function BranchAndBound($N_Q, N_O, PQ, LB, UB$):
       Input: $N_Q$:query node, $N_O$: tree node, $PQ$: priority queue,
              $UB$: the upper bound, $LB$: the lower bound
13     $maxUB, maxLB \leftarrow UB, LB$;
14     if $N_O$ is a leaf node then
15         $N_O.lb, N_O.ub \leftarrow$ Compute the lower bound and upper
             bound between $N_Q$ and $N_O$ by Lemma 6.3;
16         if $PQ.isEmpty()$ then
17             $PQ.insert(N_O)$;
18         if $N_O.lb \geq maxUB$ then
19             Pruning node $N_O$;
20         else
21             while $N_O.ub \leq maxLB$ do
22                 $PQ.Dequeue()$;
23                 if $PQ.nonEmpty()$ then
24                     $maxLB \leftarrow PQ.head().lb$ ;
25                     $maxUB \leftarrow PQ.head().ub$ ;
26                 else
27                     Break;
28             $PQ.insert(N_O)$;
29     else
30         $N_O.lb, N_O.ub \leftarrow$ Compute the lower bound and upper
             bound between $N_Q$ and $N_O$ by Lemma 6.4;
31         if $N_O.lb \geq maxUB$ then
32             Pruning node $N_O$;
33         else
34             $N_L, N_R \leftarrow$The left and right child nodes of $N_O$;
35             BranchAndBound($N_Q, N_L, PQ, maxLB, maxUB$);
36             BranchAndBound($N_Q, N_R, PQ, maxLB, maxUB$);
```

flow iteratively for each $q_i \in \mathbf{q}$, i.e., moving all density of q_i to the nearest p_j, under the flow constraints (7) and (8).

For the sake of illustration of ICT flow and TICT flow, $R(i)$ denotes the remaining density of q_i after the density transfer from q_i to p_j, $D(j)$ denotes the desired density of p_j. We initialize $\forall i \in [1, m], j \in [1, n]$: $R(i) = q_i$ and $D(j) = p_j$. Next, we present the formal definition of the ICT flow.

Definition 7. *For $\forall i \in [1, m]$, the ICT flow is distributed in ascending order of $d(i, j)$ as:*

$$f(i,j) = \begin{cases} p_j, & if\ R(i) > p_j \\ R(i), & else \end{cases}$$

The remaining density $R(i)$ is defined as:

$$R(i) = \begin{cases} R(i) - f(i,j), & if\ R(i) > p_j \\ 0, & else \end{cases}$$

The ICT flow ensures that the density of $\mathbf{q}$ is fully transferred to its corresponding nearest neighbor p_j in $\mathbf{p}$. We use a toy example to illustrate the difference between EMD optimal flow and ICT flow. In Figure 9, the numbers 1 to 6 at the bottom of the horizontal line denote the position in one dimensional feature space, and the ground distance between position i and j is simply defined as $|i - j|$, i.e., $d(2, 1) = |3 - 2| = 1$. The density of each bin is represented by the numbers in the bin. Figure 9 illustrates a minimum cost flow between $\mathbf{q}$ and $\mathbf{p}$ under the EMD constraints, and their EMD distance is $0.2 \times 0 + 0.3 \times 3 + 0.3 \times 3 + 0.2 \times 2 = 2.2$.

Figure 10 shows the ICT flow between two histograms $\mathbf{q}$ and $\mathbf{p}$. Specifically, the ICT algorithm first ranks the $d(1, j)$ in ascending order according to their ground distance, i.e., $d(1, 1) \leq d(1, 2) \leq d(1, 3)$, then transfers the $\min(q_1, p_1)$ density from q_1 to p_1. At this time, the remaining density of q_1 is 0.3, which is transferred to p_2. When the remaining density of q_1 is 0, the density of q_2 and q_3 is transferred sequentially according to the same transfer rules. In this way, the flow cost is $0.2 \times 0 + 0.3 \times 3 + 0.2 \times 1 + 0.1 \times 2 + 0.2 \times 1 = 1.5$.

However, the lower bound generated by the ICT algorithm is still not tight enough. In Figure 10, we can see that the histogram generated by the ICT algorithm is significantly different from the target histogram $\mathbf{p}$. To obtain a tighter "bound", we further propose a tighter ICT (TICT) algorithm. Next, we present the formal definition of the TICT flow.

Definition 8. *For $\forall i \in [1, m]$, the TICT flow is distributed in ascending order of $d(i, j)$ as:*

$$f(i,j) = \begin{cases} D(j), & if\ R(i) > D(j), \\ R(i), & else \end{cases}$$

After the density transfers, the desired density $D(j)$ is defined as:

$$D(j) = \begin{cases} 0, & if\ R(i) > D(j) \\ p_j, & else \end{cases}$$

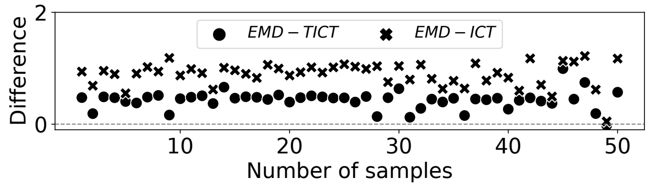


Figure 12: The diagram of differences between EMD and TICT, and between EMD and ICT in a set of samples.

The remaining density $R(i)$ is defined as:

$$R(i) = \begin{cases} R(i) - D(j), & if\ R(i) > D(j) \\ 0, & else \end{cases}$$

Algorithms 3 shows the process of the TICT algorithm. For each q_i, we first compute the distance between q_i and p_j ($j = [1, 2, \ldots, n]$), then construct a min-heap over n distance values in ascending order (see Line 3 to Line 6). Secondly, we find the indices (i, j) of the smallest distance from the min-heap and transfer the density from q_i to p_j under the capacity constraints, until the remaining density of q_i equals 0 (see Line 8 to Line 18). Note that after the density of q_j is satisfied, we change its density to 0, which prevents continual transfer density to those bins that already satisfy the target density (see Line 16).

Complexity Analysis. For $\forall q_i \in \mathbf{q}$, we first compute the ground distance between q_i and all p_j ($j \in [1, n]$). Then we construct the min-heap according to the $d(i, j)$, so the time complexity is $O(n \log n)$. As $\mathbf{q} = \{q_1, q_2, \ldots, q_m\}$, the above step needs to be executed m times. Thus the overall time complexity of TICT is $O(mn \log n)$.

Next, we prove that the TICT bound must be greater than or equal to the ICT lower bound. In the RDM problem, the ICT algorithm iteratively computes the minimum flow cost on the premise of satisfying all RDM constraints by a greedy strategy, i.e., the ICT algorithm can obtain the minimum value of the RDM problem. Since the flow generated by the TICT is a feasible solution for the RDM problem, the TICT bound must be greater than or equal to the ICT lower bound.

Figure 11 illustrates the TICT flow generated by the TICT algorithm. After ranking $d(1, j)$ in ascending order according to their ground distance, the TICT algorithm first transfers $\min(q_1, p_1) = 0.2$ density from q_1 to p_1 and $\min(R_1, p_2) = 0.3$ density from q_1 to p_2. Since the target density of p_1 and p_2 has been satisfied, other bins cannot transfer density to p_1 and p_2. Thus, the density of q_2 is transferred to p_3, even though p_3 is not the nearest neighbor. The flow cost of TICT is $0.2 \times 0 + 0.3 \times 3 + 0.3 \times 3 + 0.2 \times 2 = 2.2$.

It is important to note that in theory the TICT is not a valid lower bound for EMD. However, we observe in our experiments that the TICT is almost always smaller than the EMD and is tighter than the ICT [5]. In Figure 12, we randomly select a set of samples and plot the differences between EMD and TICT and between EMD and ICT. The plot shows that the TICT is tighter than the ICT. In the one case where TICT is not a valid lower bound, the value of TICT is still very close to EMD.

Algorithm 4 shows the process of the top-k EMD search in detail. We first generate the candidate set C by Algorithm 2 (see Line 2). Then, we initialize a null priority queue PQ and an infinite filtering threshold τ. For all datasets in the candidate set, we compute the TICT bounds and insert them into PQ in ascending order (see Line 5 to Line 8). Next, we traverse the dataset in the priority queue. When the length of the result list is less than k, we directly compute the exact EMD and update the threshold (see Line 11 to Line 14). Subsequently, if the TICT bound of the candidate dataset is less

Algorithm 3: TICT($\mathbf{q}, \mathbf{p}$)

```
Input: q: Z-order histogram of the dataset $Q$, p: Z-order histogram
       of the dataset $P$
Output: B: TICT bound
1  $B \leftarrow 0$;
2  foreach $q_i \in q$ do
3      $H \leftarrow$ Initialize a min-heap;
4      foreach $p_j \in p$ do
5          $d(i,j) \leftarrow$ Compute the distance between $q_i$ and $p_j$;
6          $H.insert(d(i,j))$;
7      $d(i,j) \leftarrow H.pop()$;
8      while $q_i > 0$ do
9          if $q_i \leq p_j$ then
10             $f \leftarrow q_i$;
11             $q_i \leftarrow 0$;
12             $B \leftarrow cost + f \times d(i,j)$;
13         else
14             $f \leftarrow p_j$;
15             $q_i \leftarrow q_i - f$;
16             $p_j \leftarrow 0$;
17             $B \leftarrow cost + f \times d(i,j)$ ;
18         $d(i,j) \leftarrow H.pop()$;
19 return $B$;
```

Algorithm 4: TopKSearch($Q, k, \mathcal{D}$)

```
Input: $Q$: query dataset, $k$: number of results, $\mathcal{D}$: data repository
Output: $L$: result list
1  q $\leftarrow$ Generate the Z-order histogram of $Q$;
2  $C \leftarrow$ CandidatesGen($Q, \mathcal{D}$);
3  $PQ \leftarrow$ Initialize a priority queue;
4  $\tau \leftarrow \infty$;
5  foreach $c \in C$ do
6      c $\leftarrow$ Generate the Z-order histogram of $c$;
7      $B \leftarrow TICT(\mathbf{q}, \mathbf{c})$;
8      $PQ.insert(\mathbf{c}, B)$;
9  while $PQ.nonEmpty()$ do
10     $(\mathbf{c}, B) \leftarrow PQ.Dequeue()$;
11     if $L.size() < k$ then
12         $e \leftarrow$ Compute the exact EMD between q and c;
13         $L.insert(\mathbf{c})$;
14         $\tau \leftarrow \min(\tau, e)$;
15     else
16         if $B < \tau$ then
17             $e \leftarrow$ Compute the exact EMD between q and c;
18             if $e < \tau$ then
19                 $L.pop()$;
20                 $L.insert(c)$;
21         else
22             Break;
23 return $L$;
```

Table 3: Details of four spatial data repositories.

Data repository	Storage (GB)	Number of datasets	Number of points	Coordinates range
`Argoverse`	16.86	205,942	[157, 3394]	[(-102.43, 552.15), (4798.03, 4099.93)]
`Trackable`	4.48	66,380	[1, 2,223,200]	$[(-179°98', -78°20'), (179°99', 74°63')]$
`Identifiable`	19.64	235,483	[1, 3,043,935]	$[(-180°00', -90°00'), (179°99', 90°00')]$
`Public`	29.43	546,193	[1, 3,747,735]	$[(-180°00', -84°14'), (179°99', 90°00')]$

Table 4: Parameter settings.

Parameter	Settings
k: number of results	{5, 8, <u>10</u>, 13, 15, 20}
s: number of datasets	{0.2, 0.4, 0.6, 0.8, <u>1</u>} $\times|\mathcal{D}|$
θ for `Argoverse`	{5, 6, <u>7</u>, 8, 9}
θ for others	{10, 11, 12, <u>13</u>, 14, 15 }

than the filtering threshold τ, the dataset needs to be verified (see Line 16 to Line 20); otherwise, the dataset and subsequent candidate datasets are filtered without computing the exact EMD (see Line 22).

7 EXPERIMENTS

We first introduce the experimental settings. Then, we evaluate the efficiency of our proposed DBF framework compared with three state-of-the-art methods, which will be elaborated on later. The evaluation mainly covers the following aspects: 1) the performance of our data modeling approach; 2) the validity of our proposed TICT bound; and 3) the efficiency of the DBF framework under various parameters.

7.1 Settings

7.1.1 Datasets. We evaluate our DBF framework on four real-world spatial data repositories. `Argoverse`[3] is a widely used self-driving dataset. The other three are public GPS datasets we collected from OpenStreetMap[4], which records real geographic datasets from the world.

We show the details of these four spatial data repositories in Table 3, which covers the storage capacity, the number of datasets, the number of points, and the range of coordinates. Figure 13 shows the distribution of dataset scale in four data repositories. Since the number of points in datasets ranges from one to several million, we only present the points distribution of 90% of the datasets in `Trackable`, `Identifiable`, and `Public`. In addition, we show the heatmaps of four data repositories in the appendix of our technical repore [54].

7.1.2 Implementation. We implement all algorithms in Java 1.8 and run all experiments on a 10-core Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHZ processor, with 376G memory. The implementation code and guidelines can be obtained on the Github[5].

7.1.3 Query Generation. We randomly select 10 datasets from each data repository as query datasets to search the remaining datasets and report the average running time of these 10 queries. To prove the validity of our approach in data modeling, index structure, and top-k search, we vary several key parameters to observe the performance of our method compared with several state-of-the-art methods. These parameters include the number of results (k), resolution (θ), and the scale of datasets (s). The coordinates of points in `Argoverse` are fixed coordinate values relative to a single city, while in the other three datasets, the range of points is worldwide. Thus, we select different resolution ranges for `Argoverse` and the other three spatial datasets. We summarize the parameter settings in Table 4, and the default value for each parameter is underlined.

7.2 Efficiency of the DBF Framework

Spatial Data Storage and Compression. In the process of spatial data modeling, we store these Z-order histograms in memory after converting the spatial datasets to Z-order histograms. Thus, we first test the memory space change caused by altering the resolution θ. The left of Figure 14 shows that the required storage space gradually decreases as the resolution decreases. This is because when we use a lower resolution, the whole space is divided coarse-grained, which results in a reduction in the number of bins of Z-order histogram generated per dataset.

We then compare the running time of Z-order histograms generated by the Z-order pooling method and scanning method. The experimental results are shown in the right of Figure 14. The first column for each data repository shows the running time of data modeling by scanning all datasets for the first time. As the resolution decreases, we compare the running time of modeling by using the Z-order pooling method or rescanning all datasets. We observe that our Z-order pooling method achieves significant speed-up by order of magnitude. This is because the Z-order pooling method only needs to scan all datasets once, which reduces the time of repeatedly scanning all datasets in the data repository.

Efficiency of the TICT Bound. To verify the efficiency of our proposed TICT bound, we compare the TICT bound with three state-of-the-art lower bounds proposed over the past few years in terms of both accuracy and running time:

- ICT [5]: The ICT algorithm has been introduced in Section 6.4, whose key idea is to iteratively compute the minimum transfer cost for each bin of histogram $\mathbf{q}$.
- IM-Sig* [45]: The IM-Sig* constructs a $m \times n$ distance matrix for two histograms $\mathbf{q}$ and $\mathbf{p}$, and extracts the smallest distance from the matrix to transfer the density until all density in $\mathbf{p}$ is transferred to $\mathbf{q}$.
- RWMD [27]: The RWMD provides a lower bound of EMD by computing the dot-product between p_i ($i = [1, 2, \ldots, m]$) and the minimum value of i-th row of distance matrix.

We use the classic simplex algorithm [17] to compute the exact EMD. However, due to the high complexity of EMD, it is not feasible to perform the pairwise EMD computation for all datasets. Thus,

[3]https://www.argoverse.org
[4]https://www.openstreetmap.org/traces
[5]https://github.com/yangwenzhe/DBF/

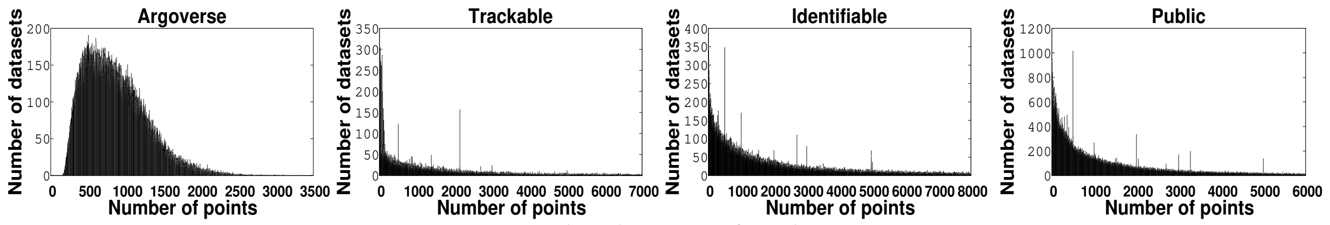


Figure 13: Dataset distributions in four data repositories.

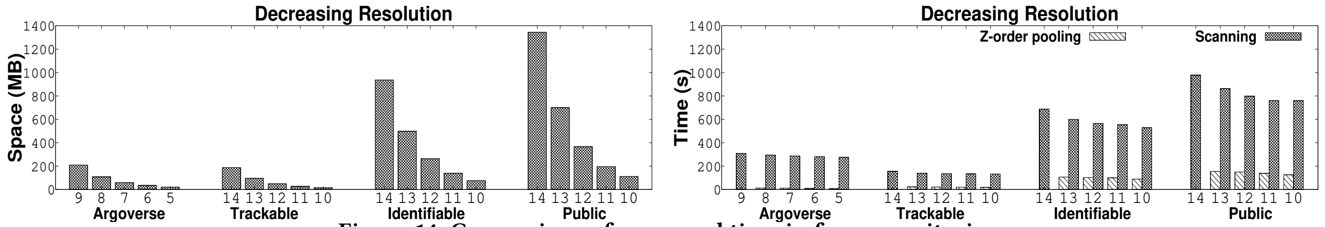


Figure 14: Comparison of space and time in four repositories.

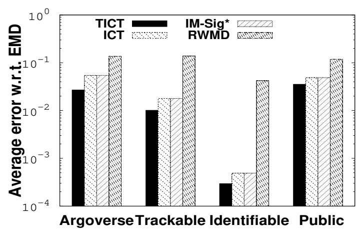


Figure 15: The average error comparison of several bounds.

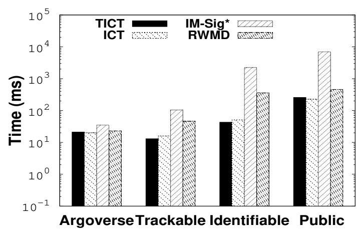


Figure 16: The running time comparison of several bounds.

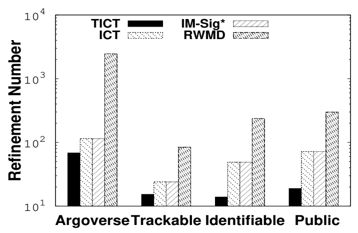


Figure 17: The refinement number comparison of several bounds.

we randomly select 100 pairs of datasets in four data repositories to compare the average error between several bounds and exact EMD. We use the Euclidean distance as the ground distance. Figure 15 shows the average error of the bounds computed by the four algorithms in four data repositories. We observe that our TICT achieved the smallest average error. The ICT and IM-Sig* have the same average error, and the RWMD is rather loose.

Next, Figure 16 presents the running time of the four methods in four data repositories. We observe that our method is faster than IM-Sig* and RWMD. Although the running time of TICT is not always faster than ICT, TICT is closer to the exact EMD. Finally, we use the four bounds to compare the number of datasets that need validation in the refinement stage. In Figure 17, we can see that compared with the other three lower bounds, our method achieves stronger filtering power. The main reason is that the TICT has the smallest average error relative to the exact EMD, which helps us filter out more candidate datasets.

Efficiency of the Index Structure. We compare the running time on different stages of DBF framework and the SAR search based on TICT, ICT, and IM-Sig*, respectively. We do not show the RWMD results in the subsequent figures. The main reason is that RWMD is too loose relative to exact EMD, which results in the overall search time of RWMD being far more than the other baselines.

We intuitively show each stage's average running time, which contains the index construction stage, filtering stage, and refinement stage. In Figure 18, we can observe that the index construction is very fast, which can be completed in less than one second. We can also see that the filtering of the DBF framework is faster than the other three methods by one order of magnitude. Such a significant improvement shows that using the pooling-based index structure to prune datasets in batch is essential in the coarse-grained filtering stage. In addition, as shown in Figure 18, the refinement is the most time-consuming stage, which increases one to two orders of magnitude compared with the other three stages. However, the refinement time of our DBF framework is still less than the other three methods. Since it has considerable filtering power and faster filtering, DBF achieves the best top-*k* search performance.

Efficiency of the Top-*k* Search. From the results in Figure 19, we can see that our DBF framework consistently outperforms the other three methods. The reason is that our framework can prune datasets in batch with the help of our pooling-based index structure, which avoids pairwise bound computation in the filtering stage. In addition, we can see that the filtering based on the TICT without index has better performance than that based on ICT and IM-Sig*. This is because the TICT is closer to the exact EMD, resulting in more datasets being filtered out in the filtering stage. Thus, the search process could terminate faster than that of ICT and IM-Sig*.

Top-*k* Search Efficiency under Different Resolutions. In the data modeling stage, we divide the whole space into a $2^{\theta} \times 2^{\theta}$ grid and generate the Z-order histogram for each dataset. Thus, the number of bins in each Z-order histogram depends on the resolution θ. We increase θ to investigate the search performance of the DBF framework compared with the other three methods. From the results in Figure 20, we observe that the search time of the four methods increases gradually as the resolution increases. The reason is that each dataset is divided into finer granularity with the increase of resolution, increasing the number of histogram bins. Another finding is that our DBF framework is still faster than the

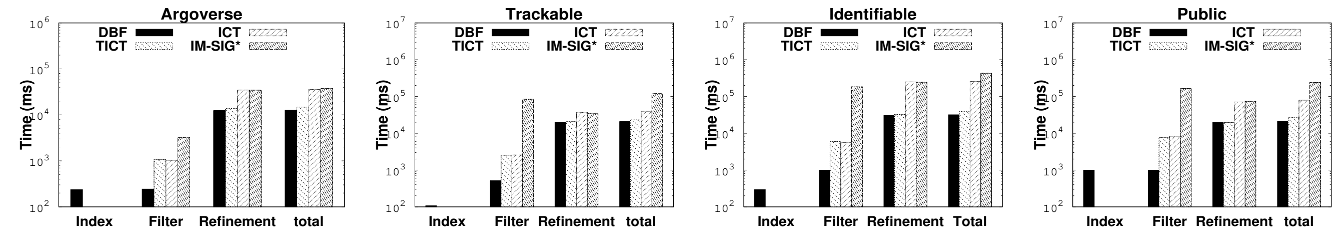


Figure 18: The running time comparison of different stages.

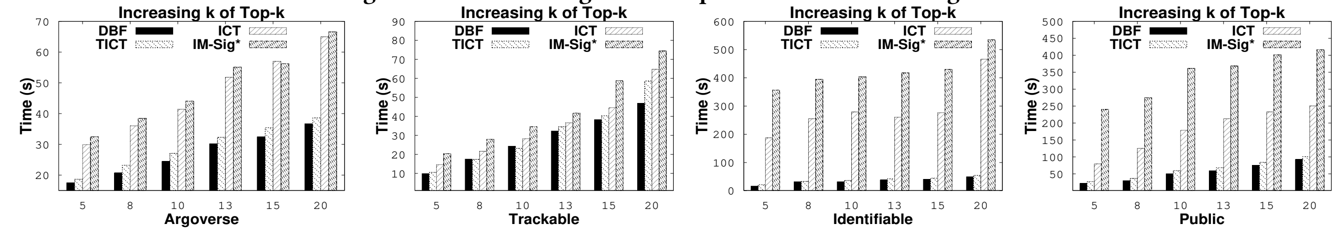


Figure 19: Effect of the top-k search with the increase of k.

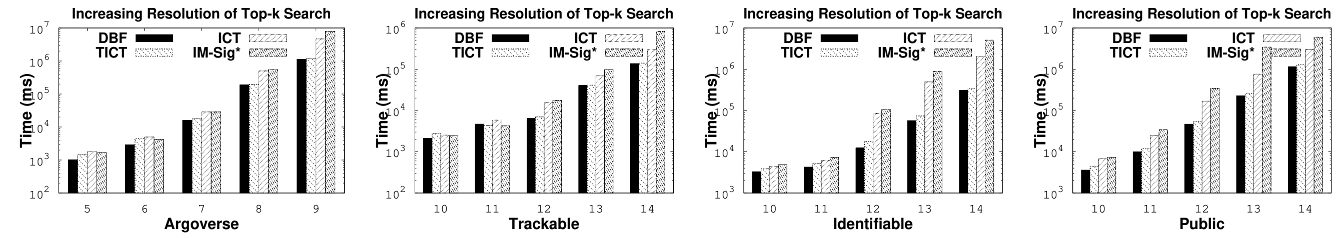


Figure 20: Effect of the top-k search with the increase of θ.

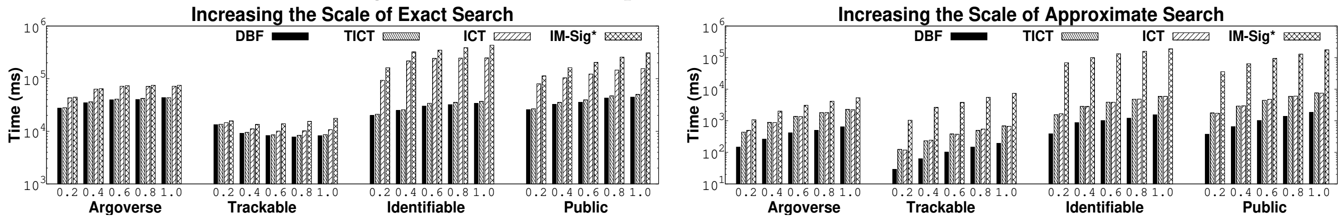


Figure 21: Effect of the top-k search with increasing dataset scale.

other three methods under different resolutions, which proves that our DBF framework is robust to different resolutions.

Scalability. Finally, we test the scalability of our framework. We vary the dataset size of four data repositories from 20% to 100% and report the average search time. The experimental results of the exact search in four data repositories are shown in the left of Figure 21. In addition, we use the bound as approximate EMD to quickly obtain the search results, suitable for the situation requiring high query speed. The right of Figure 21 shows the performance of the approximate search. In Figure 21, we can see that no matter how the size of the dataset increases, our method can maintain a relatively faster search speed than other methods.

Additional Experiments. We conduct more experiments to verify the efficiency of our DBF framework. First, we conduct experiments on the higher-dimensional data repository to show that our method is also applicable in higher-dimensional space. Second, we conduct the cross repository search experiments to compare the running time of four methods. Third, to support interactive search, we use the TICT bound as an approximate EMD and compare its running time and accuracy with other approximate methods. Finally, we investigate the effect of data skew on running time and accuracy. Due to the space limit, we place these experimental results in the appendix of technical report [54].

8 CONCLUSIONS

This paper mainly studied the EMD-based spatial dataset search. We presented DBF, a dual filtering framework to accelerate the similarity search. First, we showed how to represent the spatial datasets by Z-order histograms. Then, we designed a Z-order pooling method to organize those datasets as nodes in a tree-structured index. Next, we deduced pooling-based lower and upper bounds and designed pruning strategies combined with index structure to filter datasets in batch. Subsequently, we proposed a TICT bound based on the ICT lower bound to conduct SAR search, further improving the filtering ratio. The experimental results showed that our framework is more efficient than the state-of-the-art methods. In future, we will explore the spatial dataset search to return the subset of the spatial dataset that is the most similar to the given query dataset.

ACKNOWLEDGMENTS

This work was partially supported by the National Natural Science Foundation of China (No. U1811263).

REFERENCES

[1] Marco D. Adelfio, Sarana Nutanong, and Hanan Samet. Similarity search on a large collection of point sets. In *SIGSPATIAL*, pages 132–141, 2011.

[2] David L. Applegate, Tamraparni Dasu, Shankar Krishnan, and Simon Urbanek. Unsupervised clustering of multidimensional distributions using earth mover distance. In *SIGKDD*, pages 636–644, 2011.

[3] Pattaramanee Arsomngern, Cheng Long, Supasorn Suwajanakorn, and Sarana Nutanong. Self-supervised deep metric learning for pointsets. In *ICDE*, pages 2171–2176, 2021.

[4] Ira Assent, Andrea Wenning, and Thomas Seidl. Approximation techniques for indexing the earth mover's distance in multimedia databases. In *ICDE*, pages 11–23, 2006.

[5] Kubilay Atasu and Thomas Mittelholzer. Linear-complexity data-parallel earth mover's distance approximations. In *ICML*, pages 364–373, 2019.

[6] Kubilay Atasu, Thomas P. Parnell, Celestine Dünner, Manolis Sifalakis, Haralampos Pozidis, Vasileios Vasileiadis, Michail Vlachos, Cesar Berrospi, and Abdel Labbi. Linear-complexity relaxed word mover's distance with GPU acceleration. In *BigData*, pages 889–896, 2017.

[7] Arturs Backurs, Yihe Dong, Piotr Indyk, Ilya P. Razenshteyn, and Tal Wagner. Scalable nearest neighbor search for optimal transport. In *ICML*, volume 119, pages 497–506, 2020.

[8] Alex Bogatu, Alvaro A. A. Fernandes, Norman W. Paton, and Nikolaos Konstantinou. Dataset discovery in data lakes. In *ICDE*, pages 709–720, 2020.

[9] Dan Brickley, Matthew Burgess, and Natasha F. Noy. Google dataset search: Building a search engine for datasets in an open web ecosystem. In *WWW*, pages 1365–1375, 2019.

[10] Chu Cao and Mo Li. Generating mobility trajectories with retained data utility. In *KDD*, pages 2610–2620, 2021.

[11] Sonia Castelo, Rémi Rampin, Aécio S. R. Santos, Aline Bessa, Fernando Chirigati, and Juliana Freire. Auctus: A dataset search engine for data discovery and augmentation. *PVLDB*, 14(12):2791–2794, 2021.

[12] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. In *CVPR*, pages 8748–8757, 2019.

[13] Zhiyu Chen, Haiyan Jia, Jeff Heflin, and Brian D. Davison. Leveraging schema labels to enhance dataset search. In *ECIR*, volume 12035, pages 267–280, 2020.

[14] Zhiyu Chen, Mohamed Trabelsi, Jeff Heflin, Yinan Xu, and Brian D. Davison. Table search using a deep contextualized language model. In *SIGIR*, pages 589–598, 2020.

[15] Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, Entong Shen, and Ting Yu. Differentially private spatial decompositions. In *ICDE*, pages 20–31, 2012.

[16] Mercè Crosas. The dataverse network®: An open-source application for sharing, discovering and preserving data. *D Lib Mag.*, 17(1/2), 2011.

[17] George B. Dantzig. Linear programming. *Oper. Res.*, 50(1):42–47, 2002.

[18] Auriol Degbelo and Brhane Bahrishum Teka. Spatial search strategies for open government data: A systematic comparison. In *GIR*, pages 1–10, 2019.

[19] Yuyang Dong, Kunihiro Takeoka, Chuan Xiao, and Masafumi Oyamada. Efficient joinable table discovery in data lakes: A high-dimensional similarity-based approach. In *ICDE*, pages 456–467, 2021.

[20] Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. Aurum: A data discovery system. In *ICDE*, pages 1001–1012, 2018.

[21] Raul Castro Fernandez, Jisoo Min, Demitri Nava, and Samuel Madden. Lazo: A cardinality-based method for coupled estimation of jaccard similarity and containment. In *ICDE*, pages 1190–1201, 2019.

[22] Patricia Frontiera, Ray R. Larson, and John Radke. A comparison of geometric approaches to assessing spatial similarity for GIR. *Int. J. Geogr. Inf. Sci.*, 22(3): 337–360, 2008.

[23] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. *ACM SIGMOD Record*, 14(2):47–57, 1984.

[24] Stefan Hahmann and Dirk Burghardt. How much information is geospatially referenced? networks and cognition. *Int. J. Geogr. Inf. Sci.*, 27(6):1171–1189, 2013.

[25] Alon Y. Halevy, Flip Korn, Natalya Fridman Noy, Christopher Olston, Neoklis Polyzotis, Sudip Roy, and Steven Euijong Whang. Goods: Organizing google's datasets. In *SIGMOD*, pages 795–806, 2016.

[26] Thomas Hervey, Sara Lafia, and Werner Kuhn. Search facets and ranking in geospatial dataset search. In *GIScience*, pages 1–15, 2021.

[27] Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. From word embeddings to document distances. In *ICML*, pages 957–966, 2015.

[28] Davide Mottin, Matteo Lissandrini, Yannis Velegrakis, and Themis Palpanas. Exemplar queries: Give me an example of what you need. *PVLDB*, 7(5):365–376, 2014.

[29] Davide Mottin, Matteo Lissandrini, Yannis Velegrakis, and Themis Palpanas. Exemplar queries: a new way of searching. *VLDBJ*, 25(6):741–765, 2016.

[30] Fatemeh Nargesian, Ken Q. Pu, Erkang Zhu, Bahar Ghadiri Bashardoost, and Renée J. Miller. Organizing data lakes for navigation. In *SIGMOD*, pages 1939–1950, 2020.

[31] Sarana Nutanong, Edwin H. Jacox, and Hanan Samet. An incremental Hausdorff distance calculation algorithm. *PVLDB*, 4(8):506–517, 2011.

[32] Paul Ouellette, Aidan Sciortino, Fatemeh Nargesian, Bahar Ghadiri Bashardoost, Erkang Zhu, Ken Q Pu, and Renée J Miller. Ronin: data lake exploration. *PVLDB*, 14(12):2863–2866, 2021.

[33] Ofir Pele and Michael Werman. A linear time histogram metric for improved SIFT matching. In *ECCV*, pages 495–508, 2008.

[34] Jinglin Peng, Hongzhi Wang, Jianzhong Li, and Hong Gao. Set-based similarity search for time series. In *SIGMOD*, pages 2039–2052, 2016.

[35] Jan Ramon and Maurice Bruynooghe. A polynomial time computable metric between point sets. *Acta Informatica*, 37(10):765–780, 2001.

[36] Walter Renteria-Agualimpia, Francisco J. Lopez-Pellicer, Javier Lacasta, F. Javier Zarazaga-Soria, and Pedro R. Muro-Medrano. Improving the geospatial consistency of digital libraries metadata. *J. Inf. Sci.*, 42(4):507–523, 2016.

[37] El Kindi Rezig, Anshul Bhandari, Anna Fariha, Benjamin Price, Allan Vanterpool, Vijay Gadepally, and Michael Stonebraker. Dice: data discovery by example. *PVLDB*, 14(12):2819–2822, 2021.

[38] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. A metric for distributions with applications to image databases. In *ICCV*, pages 59–66, 1998.

[39] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vis.*, 40(2):99–121, 2000.

[40] Brian E Ruttenberg and Ambuj K Singh. Indexing the earth mover's distance using normal distributions. *PVLDB*, 5(3):205–216, 2011.

[41] Jie Shao, Zi Huang, Heng Shen, Jialie Shen, and Xiaofang Zhou. Distribution-based similarity measures for multi-dimensional point set retrieval applications. In *MM*, pages 429–438, 2008.

[42] Pyrrhos Stathis, Stamatis Vassiliadis, and Sorin Cotofana. A hierarchical sparse matrix storage format for vector processors. In *IPDPS*, pages 61–69, 2003.

[43] Abdel Aziz Taha and Allan Hanbury. An efficient algorithm for calculating the exact hausdorff distance. *TPAMI*, 37(11):2153–2163, 2015.

[44] Yu Tang, Leong Hou U, Yilun Cai, Nikos Mamoulis, and Reynold Cheng. Earth mover's distance based similarity search at scale. *PVLDB*, 7(4):313–324, 2013.

[45] Merih Seran Uysal, Christian Beecks, Daniel Sabinasz, Jochen Schmücking, and Thomas Seidl. Efficient query processing using the earth mover's distance in video databases. In *EDBT*, pages 389–400, 2016.

[46] Padf Vasconcelos, Wds Alencar, Vhds Ribeiro, N. F. Rodrigues, and Fdg Andrade. Enabling spatial queries in open government data portals. In *EGOVIS*, pages 64–79, 2017.

[47] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, Timos Sellis, Mark Sanderson, and Xiaolin Qin. Answering top-k exemplar trajectory queries. In *ICDE*, pages 597–608, 2017.

[48] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, Zizhe Xie, Qizhi Liu, and Xiaolin Qin. Torch: A search engine for trajectory data. In *SIGIR*, pages 535–544, 2018.

[49] Sheng Wang, Zhifeng Bao, Shixun Huang, and Rui Zhang. A unified processing paradigm for interactive location-based web search. In *WSDM*, page 601–609, 2018.

[50] Ting Wang, Shicong Meng, and Jiang Bian. Indexing earth mover's distance over network metrics. *IEEE Trans. Knowl. Data Eng.*, 27(6):1588–1601, 2015.

[51] Marc Wichterich, Ira Assent, Philipp Kranen, and Thomas Seidl. Efficient emd-based similarity search in multimedia databases via flexible dimensionality reduction. In *SIGMOD*, pages 199–212, 2008.

[52] Chuan Xiao, Wei Wang, Xuemin Lin, and Haichuan Shang. Top-k set similarity joins. In *ICDE*, pages 916–927, 2009.

[53] Jia Xu, Zhenjie Zhang, Anthony K. H. Tung, and Ge Yu. Efficient and effective similarity search over probabilistic data based on earth mover's distance. *VLDBJ*, 21(4):535–559, 2012.

[54] Wenzhe Yang, Sheng Wang, Yuan Sun, and Zhiyong Peng. Fast dataset search with earth mover's distance (technical report). 2022. URL http://shengwang.site/papers/22VLDB-tr.pdf.

[55] Eleni Tzirita Zacharatou, Darius Šidlauskas, Farhan Tauheed, Thomas Heinis, and Anastasia Ailamaki. Efficient bundled spatial range queries. In *SIGSPATIAL*, pages 139–148, 2019.

[56] Sepanta Zeighami, Ritesh Ahuja, Gabriel Ghinita, and Cyrus Shahabi. A neural database for diferentially private spatial range queries. *PVLDB*, 15(5):1066–1078, 2021.

[57] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *CVPR*, pages 12200–12210, 2020.

[58] Yi Zhang and Zachary G Ives. Finding related tables in data lakes for interactive data science. In *SIGMOD*, pages 1951–1966, 2020.

[59] Erkang Zhu, Dong Deng, Fatemeh Nargesian, and Renée J Miller. Josie: Overlap set similarity search for finding joinable tables in data lakes. In *SIGMOD*, pages 847–864, 2019.