Indexing Earth Mover's Distance over Network Metrics

Ting Wang Shicong Meng Jiang Bian

Abstract—The Earth Mover's Distance (EMD) is a well-known distance metric for data represented as probability distributions over a predefined feature space. Supporting EMD-based similarity search has attracted intensive research effort. Despite the plethora of literature, most existing solutions are optimized for L^p feature spaces (e.g., Euclidean space); while in a spectrum of applications, the relationships between features are better captured using networks. In this paper, we study the problem of answering k-nearest neighbor (k-NN) queries under network-based EMD metrics (NEMD). We propose OASIS, a new access method which leverages the network structure of feature space and enables efficient NEMD-based similarity search. Specifically, OASIS employs three novel techniques: (i) *Range Oracle*, a scalable model to estimate the range of k-th nearest neighbor under NEMD, (ii) *Boundary Index*, a structure that efficiently fetches candidates within given range, and (iii) *Network Compression Hierarchy*, an incremental filtering mechanism that effectively prunes false positive candidates to save unnecessary computation. Through extensive experiments using both synthetic and real datasets, we confirmed that OASIS significantly outperforms the state-of-the-art methods in query processing cost.

Index Terms—Earth Mover's Distance, Network Metrics, Similarity Search, Delimit and Filter

1 INTRODUCTION

With continued advance in data acquisition technologies (e.g., geographic tracking [1], sensor network [2], social media [3]), recent years have witnessed a new deluge of probabilistic data, which is represented as probability distributions over a predefined feature space. Due to the complexity of measuring distance between distributions, supporting similarity search in this context represents new challenges for data management research.

One of the most popular similarity metrics for probabilistic data is the Earth Mover's Distance (EMD). It intuitively captures the minimum cost required to transform one distribution into another and is robust against outliers and slight probability shifting. These desired properties have made EMD increasingly popular in a range of applications, such as multimedia retrieval [4], [5], [6], clustering comparison [7], and shape matching [8]. The quality of similarity measure, however, comes at the cost of computational efficiency. Although solvable using linear programming methods (e.g., [9]), the empirical time complexity of computing EMD is typically cubic in the number of features, which is prohibitively expensive for large-scale, high-dimensional data.

A substantial body of work has been dedicated to addressing the efficiency issues of EMD-based similarity search [4], [5], [6], [10], [11], [12], [13], [14]. In most existing solutions, a de facto, and sometimes crucial [10], [13], assumption is that the features are embedded in certain L^p spaces and the ground distance between two features is measured by their L^p distance (e.g., Euclidean or Manhattan distance). Nevertheless, in a spectrum of applications, it could be extremely hard to find a proper L^p embedding for the feature space. For example,

- In transport engineering, the geographic patterns of traffics are frequently modeled as distributions over underlying road networks [15].
- In natural language processing, recent studies have introduced ways to represent documents as distributions over graphical structures of words [16].
- In recommender systems (e.g., Netflix), users' preferences are often represented as distributions over items (e.g., movies), which are interconnected via their semantic relationships (e.g., genres) [17].

In the cases above, networks provide much more intuitive representations of the underlying feature spaces. Unfortunately, such network representations may violate many critical properties of L^p spaces (e.g., triangular inequality), thereby rendering many existing access methods suboptimal or even malfunctioning.

Motivated by this, we study the problem of supporting similarity search under network-based EMD metrics (NEMD). We propose Oracle-Assisted SImilarity Search (OASIS), a new access method which exploits the network structure of feature space and enables efficient k-nearest neighbor (k-NN) query processing under NEMD.

Conventionally, processing k-NN queries follows the *enumerate-and-filter* paradigm [12], [13]. At each iteration, a *getnext*() function fetches an unseen object, which is then either pruned by filters or verified by exact EMD solvers. A range threshold of the best-so-far top-k candidates is maintained to prune unpromising objects. However, as this threshold is derived from objects seen so far and no particular enumeration order is specified, in the worst case, the entire database has to be exhausted, which incurs significant scalability issues.

As illustrated in Fig. 1, OASIS overcomes this drawback by adopting a novel *delimit-and-filter* paradigm. It first determines the range of k-th NN, then fetches candidates within this range, prunes false candidates, and finally

- *T. Wang is with IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 10598. E-mail: tingwang@us.ibm.com.*
- *S. Meng is with Facebook, Inc., Menlo Park, CA, 94025. E-mail: shicong@fb.com.*
- *J. Bian is with Microsoft Research Asia, Beijing, P.R. China, 100080. E-mail: jibian@microsoft.com.*

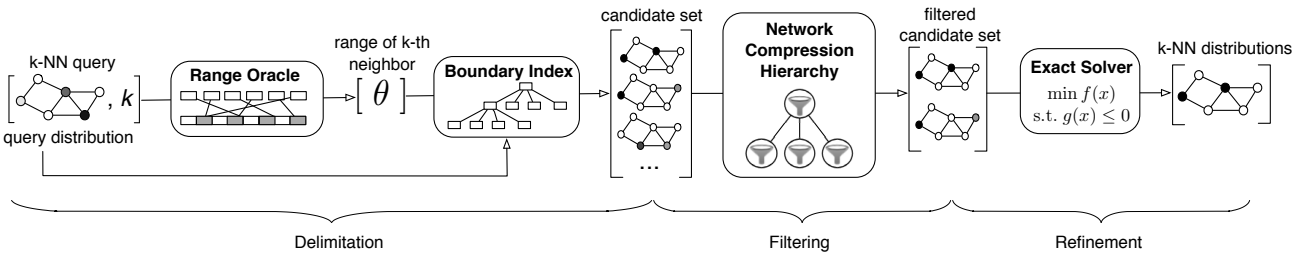

Fig. 1: Flowchart of k-NN query processing in OASIS, following the delimit-filter-refine paradigm.

feeds candidates to exact NEMD computation to identify true results. To realize this, OASIS integrates three novel techniques, *Range Oracle* (RO), *Boundary Index* (BI), and *Network Compression Hierarchy* (NCH). Intuitively, RO is a model that accurately delimits the range of k-th NN under NEMD; BI is an indexing structure that efficiently fetches candidate within range suggested by RO; while NCH is an incremental filter that effectively prunes false positive candidates. More specifically,

Range Oracle: We show that when the network structure is simplified to a tree, the restricted NEMD (TEMD) is equivalent to distance in a transformed L^1 space. More interestingly, for a given query, the range of its k-th NN under TEMD gives tight estimation for the range of its counterpart under NEMD. RO leverages this key observation and extends locality sensitive hashing (LSH) [18] to perform scalable range estimation.

Boundary Index: It maps high dimensional data to low dimensional domains while preserving their locality under NEMD. For each query, a pair of lower and upper bounds is derived for each domain, guaranteeing all the candidates residing in this interval. Compared with the classical bipartite flow formulation, BI leverages the network structure of feature space, thereby leading to much tighter bounds.

Network Compression Hierarchy: It provides a hierarchal, reduced-dimensional representation of data, which exploits both the network structure of feature space and the distribution skewness of data objects. Each level of the hierarchy offers unique trade-off between computational cost and pruning power.

Contributions

To our best knowledge, this work represents the first attempt on indexing NEMD to facilitate answering k-NN queries for large-scale datasets. Our contributions can be summarized as below:

- We highlight and articulate the problem of indexing network-based EMD metrics for similarity search;
- We propose a new delimit-and-filter paradigm, which addresses several drawbacks of the conventional enumerate-and-filter framework.
- We present a new access method, OASIS, which fulfills this paradigm, leverages the network structure of feature space, and enables efficient k-NN query processing;
- We prove the superiority of OASIS over alternative techniques with extensive experiments on both real and synthetic datasets.

Roadmap

The remainder of the paper will proceed as follows. In § 2 we survey relevant literature. In § 3 we formalize the problem of indexing NEMD for similarity search and motivate the design of OASIS. In § 4, § 5, and § 6, we detail the three core components of OASIS, Range Oracle, Boundary Index, and Network Compression Hierarchy, respectively. The proposed solution is empirically evaluated in § 7. The paper is concluded in § 8.

2 Related Work

The Earth Mover's Distance (EMD) is a popular distance metric for data represented as probability distributions over a predefined feature space. Generally, the solutions to the EMD are obtainable using methods from linear programming (e.g., transportation simplex), with empirically observed complexity of $O(n^3)$, with n being the number of features [9], which is considered expensive for many large-scale settings.

A plethora of work has been proposed to speed up the computation of EMD for large-scale, high-dimensional data [4], [5], [6], [10], [19], [20], [11], [12], [13]. Among them the first line of work (e.g., [19], [11]) attempted to approximate EMD using sketching methods (e.g., linear projection sketching [21]), with the best known results as a logarithmic approximation with sketches of poly-logarithmic size. Despite their theoretical bounds of the approximation accuracy, it is unclear how to apply the results to building indexing structures for query processing, e.g., range and k-NN queries. Another line of work explored various lower bounds of EMD so as to efficiently prune false positive cases in image retrieval using a "scan-and-refine" strategy [4], [5], [6]. However, without effective indexing support, these methods still suffer the scalability issue.

Realizing this issue, recent work [12], [13] has focused on building effective indexing structures for query processing. Xu et al. [12] introduced a lower bound B$^+$-tree indexing structure (TBI) by exploiting the dual solution to the transportation problem. Based on a feasible solution to the EMD from fixed data samples, their method creates a set of B$^+$ trees, which are used at query time to eliminate false positive candidates. The pruning power

of the found feasible solutions dictates the performance of TBI. Following a similar framework, Ruttenberg and Singh [13] proposed an indexing structure by projecting each distribution to a set of vectors and approximating the transformed distribution using a normal distribution. Unfortunately, relying on the projection bound [10], this method is only applicable for L^p feature spaces. Moreover, these lower bound-based indexing methods still suffer the scalability issue in processing k-NN queries: due to the a priori unknown range of k-th nearest neighbor, an "enumerate-and-prune" strategy is employed; the search terminates only when the entire database is either verified or pruned. In an orthogonal direction, Tang et al. [14] scaled up the refinement phase by (i) adopting an efficient min-flow algorithm for EMD computation, (ii) terminating an EMD refinement early via maintaining a dynamic distance bound, and (iii) optimizing execution order for candidates that have passed all the filters. All these techniques can well complement the indexing and filtering method proposed in this work.

Despite the simplicity of L^p metrics, in varied settings, it is natural to model feature spaces as networks, which allow more general distance definitions. Unfortunately, many existing solutions become suboptimal or even malfunctioning in this context as they rely heavily on some critical properties of L^p metrics. Efficiently estimating and indexing network-based EMD metrics (NEMD) remains one open problem, even in very restricted settings (e.g., subgraphs [22], circles [23], $\Delta \times \Delta$ grids [19]). To our best knowledge, this work represents the first attempt on indexing NEMD to facilitate similarity search for large-scale, high-dimensional data.

3 PRELIMINARIES

In this section, we introduce a set of fundamental concepts used throughout the paper, formalize the problem of NEMD-based similarity search, and then motivate the design of OASIS.

3.1 Classical EMD

The classical Earth Mover's Distance (EMD), as defined in [9], measures the minimum cost of transforming one histogram into another. This formalization is easily generalized to discrete distributions [13].

Specifically, given two distributions $o = (o_1, o_2, \ldots, o_n)$ and $q = (q_1, q_2, \ldots, q_n)$ over a predefined feature space $\mathcal{V} = (v_1, v_2, \ldots, v_n)$, one transforms o into q by moving "mass" from o_i to q_j for every pair of (i, j), such that the difference of two distributions is minimized. The cost of this transformation is defined as: $\sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} \cdot d_{ij}$, where f_{ij} represents the "flow" from o_i to q_j and d_{ij} denotes the cost of moving one unit of mass from o_i to q_j, which is typically defined by the distance between features v_i and v_j. The EMD corresponds to an optimal transformation, which minimizes the total cost:

$$\mathsf{EMD}(o, q) = \min f_{ij} \cdot d_{ij}$$
$$\text{s.t.} \begin{cases} \forall i, j: & f_{ij} \geq 0 \\ \forall i: & \sum_j f_{ij} = o_i \\ \forall j: & \sum_i f_{ij} = q_j \end{cases}$$

where the constraints ensure that all flows from o to q are nonnegative and neither overdraw o or overflow q. For simplicity, we assume o and q have equal total mass.

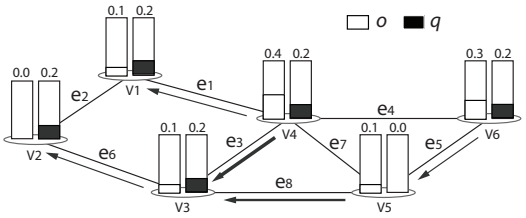

Fig. 2: Formulation of NEMD as network flows with the thickness of arrows indicating their size.

3.2 EMD over Network Metrics

A de facto assumption in literature is that features are embedded in certain L^p spaces. Nevertheless, in a range of applications (cf. § 1), the feature spaces are perhaps more naturally modeled using network structures.

In this paper, we consider the setting as follows. The feature space is modeled as an undirected network $G = (\mathcal{V}, \mathcal{E})$, with node set[1] $\mathcal{V}$ and edge set $\mathcal{E}$ respectively representing features and their relationships; The distance d_{ij} of nodes v_i and v_j is measured by their shortest path distance in G (extension to other metrics is discussed in Appendix); w_{ij} denotes the length of edge $(v_i, v_j) \in \mathcal{E}$; $\mathcal{N}_v^G$ denotes the neighbors of node $v \in \mathcal{V}$ in G. Further, a distribution o over G is defined as a discrete set of probability masses over $\mathcal{V}$: $(o_1, o_2, \ldots, o_n)$, with o_i being its mass at v_i. For simplicity, we assume (i) $\sum_{i=1}^{n} o_i = 1$ and (ii) G is connected; Otherwise each connected component can be treated separately. We entitle the EMD over such network metrics as NEMD.

Example 1: In Fig. 2, the feature space is represented by network G, which consists of node set $\{v_i\}_{i=1}^{6}$ and edge set $\{e_i\}_{i=1}^{8}$. The distribution o is encoded as a vector $o = [0.1, 0.0, 0.1, 0.4, 0.1, 0.3]$.

We are interested in the following problem: devising indexing structures for a large collection of distributions over network G, such that one can efficiently answer k-NN query (q, k), which finds the k nearest neighbors of query distribution q, under the NEMD metrics.

3.3 Design Rationale

A straightforward solution would be: (i) precomputing the pairwise distance for all the features in $\mathcal{V}$, (ii) transforming the problem to the classical EMD setting, and then (iii) applying existing NEMD indexing methods to processing k-NN queries (e.g., [12], [13]).

1. Below we use the terms "node" and "feature" interchangeably.

Unfortunately, this solution suffers several drawbacks. First, many existing methods (e.g., [4], [13]) rely heavily on some critical properties of L^p metrics and thus may be inapplicable for the network setting. Second, given a network of $|\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges, the general EMD formulation defines one variable over each pair of nodes (totally $|\mathcal{V}|^2$ variables); while as we will introduce in § 4, a network-oriented formulation defines variable over edges (totally $|\mathcal{E}|$ variables). For real datasets, typically $|\mathcal{E}| \ll |\mathcal{V}|^2$, thus the general EMD formulation incurs much higher computational cost. Finally, when processing k-NN queries, most existing indexing methods (e.g, [13], [12]) follow an enumerate-and-filter paradigm. At each iteration, a *getnext*() is invoked to fetch the next unseen data object, which is then either pruned by filters or verified by exact EMD computation. In the worst case, the search process has to exhaust the entire database, which may cause serious scalability issues.

The design objectives of OASIS are to avoid the aforementioned drawbacks of existing solutions. Specifically, OASIS possesses the following properties:

- *Removal of L^p metrics assumption*. It dose not rely on any L^p metrics-specific properties (e.g., the projection bound [10]).
- *Awareness of network*. It fully exploits the network structure of feature space to minimize the query processing cost.
- *Delimitation of search space*. It quickly narrows down the search scope of candidates to avoid enumerating the entire database.

In the following sections, we show in detail how OASIS fulfills these design objectives.

4 RANGE ORACLE

In this section, we elaborate the first core component of the delimit-and-filter framework, Range Oracle (RO). For given query q, RO accurately estimates the range of k-th NN of q under NEMD. In the following, all the proofs are referred to Appendix.

4.1 Network Flow-Based Formulation

Instead of following the conventional bipartite-flow formulation of EMD, we directly work on the flows over the edges of network G, which essentially move masses from nodes with higher o density to ones with higher q density and finally balance the two distributions. This scenario is illustrated in Fig. 2.

Note that such flows are directional, i.e., for given edge $(v_i, v_j) \in \mathcal{E}$, $f_{ij} = -f_{ji}$. The computation of NEMD is then formulated as:

$$\mathsf{NEMD}(o,q) = \min \sum_{(v_i,v_j)\in\mathcal{E}} w_{ij}|f_{ij}| \tag{1}$$

$$\text{s.t.} \begin{cases} \forall v_i \in \mathcal{V}: & \sum_{v_j \in \mathcal{N}^G_{v_i}} f_{ij} = o_i - q_i \\ \forall (v_i,v_j) \in \mathcal{E}: & f_{ij} = -f_{ji} \\ \forall (v_i,v_j) \in \mathcal{E}: & f_{ij} \in \mathbb{R} \end{cases}$$

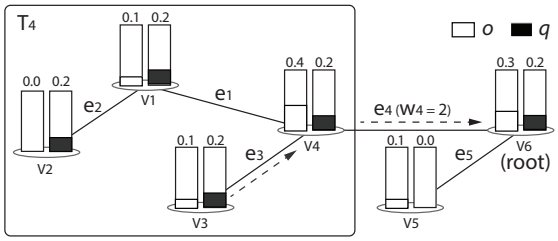

Fig. 3: Computation of NEMD over a tree.

This formulation essentially states that NEMD is the aggregated cost over all edges (flow size × edge length), conditional on that the flows must balance the density of o and q at all the nodes. Note that the total number of variables here is $|\mathcal{E}|$, which is typically much smaller than $|\mathcal{V}| \times |\mathcal{V}|$ as in classical bipartite-flow formulation.

4.2 When G is a tree

Next we show how to compute NEMD when G is a tree T. We may randomly pick a node from $\mathcal{V}$ as the root of T and specify a partial order over the rest nodes.

Definition 1 (Precedence): Given nodes $v_i, v_j \in \mathcal{V}$, we say v_i precedes v_j, denoted by $v_i \prec v_j$ (or equivalently $v_j \succ v_i$), if v_j is on the path of v_i to the root.

Example 2: In Fig. 3, with v_6 being the root, $v_3 \prec v_4$.

Deleting edge $(v_i, v_j) \in \mathcal{E}$ with $v_i \prec v_j$ partitions T into two subtrees. Denote by T_i the subtree rooted at v_i with $\mathcal{V}_{T_i}$ and $\mathcal{E}_{T_i}$ as its nodes and edges. Lemma 1 lays the foundation of computing NEMD for T.

Lemma 1: For each edge $(v_i, v_j) \in \mathcal{E}$ with $v_i \prec v_j$, the minimum possible flow over (v_i, v_j), f^*_{ij}, is given by:

$$f^*_{ij} = \sum_{v_k \in \mathcal{V}_{T_i}} (o_k - q_k) \tag{2}$$

Thus we have $\mathsf{NEMD}(o,q) = \sum_{(v_i,v_j)\in\mathcal{E}} w_{ij}|f^*_{ij}|$.

Example 3: In Fig. 3, the minimum flow over e_4 is: $f^*_{4,6}$ = $w_4 \times |(o_1 - q_1) + (o_2 - q_2) + (o_3 - q_3) + (o_4 - q_4)|$ = 0.4.

For simplicity, we now introduce the matrix formulation of Lemma 1. Let w_i be the length of the outgoing (towards the root) edge of v_i (which is defined as 0 for the root). We define an $n \times n$ matrix P, which encodes the precedence relationships:

$$[P]_{ij} = \begin{cases} w_i & v_j \preceq v_i \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Example 4: For the tree shown in Fig. 3, the precedence matrix P is given by:

$$P = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} \begin{array}{c} \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\ \begin{bmatrix} w_1 & w_1 & 0 & 0 & 0 & 0 \\ 0 & w_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & w_3 & 0 & 0 & 0 \\ w_4 & w_4 & w_4 & w_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

We have the following result.

Lemma 2: In the special case that $G = T$, given the precedence matrix P, we have $\mathsf{NEMD}(o,q) = ||P \cdot (o - q)||_1$, where $||\cdot||_1$ represents L^1-norm.

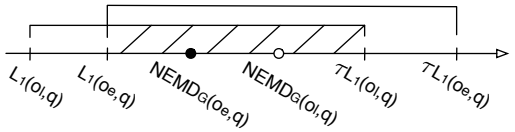

Fig. 4: Relationships of the nearest neighbors in NEMD space and that in transformed L_1-norm space.

4.3 Upper and Lower Bounds

Below we derive the bounds of NEMD for general network G. We assume a spanning tree T is predefined and will discuss shortly how to select proper spanning trees to improve the quality of range estimation. We consider two cases: (i) the exact case, wherein the flows are allowed over all the edges of G; and (ii) the restricted case, wherein the flows are only allowed over the edges of T. We use NEMD and TEMD to distinguish them. From the definitions, we first have:

Lemma 3: Given network G and its spanning tree T, NEMD is upper bounded by TEMD, that is, $\forall o, q$: $\mathsf{NEMD}(o, q) \leq \mathsf{TEMD}(o, q)$.

Intuitively this is explained by that the edges in $G \setminus T$ may "shortcut" the paths in T and lead to flows with lower cost.

Next we establish the lower bound of NEMD using the concept of *stretch*.

Definition 2 (Stretch): A spanning tree T of network G has stretch τ if the distance between every pair of nodes in T is at most τ times of their distance in G.

Thus, given a flow over a path in G, we can map this flow to a counterpart in T with cost at most τ times of that in G. This leads to a lower bound of NEMD: $\mathsf{NEMD}(o, q) \geq \mathsf{TEMD}(o, q)/\tau$. Incorporating Lemma 2, we derive the following relationships:

$$||\hat{o} - \hat{q}||_1 \leq \mathsf{NEMD}(o, q) \leq \tau \cdot ||\hat{o} - \hat{q}||_1 \tag{4}$$

where $\hat{o} = P \cdot o/\tau$ and $\hat{q} = P \cdot q/\tau$. This suggests that NEMD is well preserved in this transformed L^1 space. More interestingly, it also sheds light on the relationship between the nearest neighbors of q in the NEMD and L^1-norm spaces. Let $\mathsf{L1}(o, q) \triangleq ||\hat{o} - \hat{q}||_1$. We have:

Theorem 1: Given query q, let o_e and o_l be its k-th NN in the NEMD and transformed L^1 spaces respectively. We have: $\mathsf{L1}(o_l, q) \leq \mathsf{NEMD}(o_e, q) \leq \tau \cdot \mathsf{L1}(o_l, q)$.

As illustrated in Fig. 4, Theorem 1 entails two key implications. First, it provides a lower bound for NEMD to prune false positive results. Second, the range of o_l in the transformed L^1 space serves as an estimate of its counterpart o_e under NEMD. This estimate is often fairly tight due to the low stretch observed in real data (cf. § 7).

However, due to the hardness of indexing L^1 distance [24], computing $\mathsf{L1}(o_l, q)$ is expensive, especially for high-dimensional data. Locality sensitive hashing (LSH) [18], [25], [26], [27] is the only known practical method that handles k-NN queries at scale for high-dimensional data. We propose a novel use of LSH: efficiently estimating the range of k-th NN without identifying it. Next we show how to build RO by leveraging LSH as the building brick.

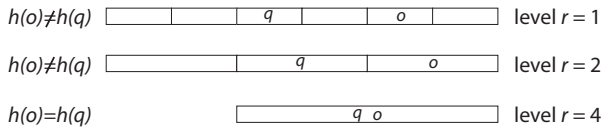

Fig. 5: Illustration of collision-counting hashing.

4.4 Basic Construction

We first briefly review the concept of LSH. We assume all the distributions have been projected to the transformed L^1 space and will refer to them as *data points*.

Intuitively an LSH function $\mathsf{h}(\cdot)$ maps data point o to an integer $\mathsf{h}(o)$, referred to as its "bucket ID" ("bid"). The probability that two points have a same bid ("collision") follows a decreasing function of their distance. Therefore, we can estimate whether two points o and q are within given distance r by counting their collisions w.r.t. a set of hash functions randomly sampled from an LSH family. If the collision number is above a certain threshold, o and q are considered within distance r.

In particular, we consider a multi-level LSH family:

$$\mathsf{h}^r_{\vec{a},b}(o) = \left\lfloor \frac{\vec{a} \cdot \vec{o} + b}{r} \right\rfloor \tag{5}$$

Here $\vec{o}$ is the vector representation of o; $\vec{a}$ is a vector with each element drawn from a p-stable distribution [18] (e.g., Cauchy distribution for L^1 distance); $\vec{a} \cdot \vec{o}$ denotes their inner product; b is uniformly drawn from $[0, c^{\log_c \lceil gt \rceil}]$, with c, g, and t respectively being the approximation ratio, the number of coordinates, and the largest coordinate value; finally r is an integer power of c with $r \leq c^{\log_c \lceil gt \rceil}$, referred to as the "level" of LSH.

Interestingly, from a set of level-1 (base) LSH functions, we can dynamically construct a set of level-r LSH functions [25], [26]. Concretely, a level-r bucket of bid x under $\mathsf{h}^r_{\vec{a},b}(\cdot)$ corresponds to the union of r consecutive level-1 buckets of bids xr, $xr + 1$, ..., $xr + (r - 1)$ under $\mathsf{h}^1_{\vec{a},b}(\cdot)$. That is, using the level-1 bid of data point o, $\mathsf{h}^1_{\vec{a},b}(o)$, we can infer its level-r bid, $\mathsf{h}^r_{\vec{a},b}(o)$, for any $r = 1, c, c^2, \ldots$, with little cost, as illustrated in Fig. 5.

Thus, RO indexes data points in the database using a base of m LSH functions[2]: $\mathcal{H}^1 = \{\mathsf{h}^1_1, \mathsf{h}^1_2, \ldots, \mathsf{h}^1_m\}$. For each function, we build a table wherein each bucket contains the set of points with the corresponding bid. From these materialized hash tables, we can easily infer the level-r bids of given points.

To estimate the range of k-th NN of query point q, we first compute the bid of q w.r.t. each base LSH function, then gradually increase the level $r = 1, c, c^2, \ldots$, and check at each level whether there are sufficient points with enough number of collisions with q. We report the minimum level satisfying this condition as the estimate. This procedure is sketched in Algorithm 1. The critical question here is how to accurately estimate the number of neighbors μ_r at each level r. However, deriving this estimation model is non-trivial; as a probabilistic dimensionality reduction method, LSH may incur both

2. We omit the reference of $\vec{a}$ and b when the context is clear.

false positive and negative errors. It is thus necessary to account for such two-sided errors in the estimation. Next we present our estimation model of RO.

Algorithm 1: Range Oracle

Input: query data point q, k of k-NN query, stretch τ
Output: range of k-th NN of q
Notations: μ_r: number of neighbors within r; l_i: number of collisions between data point o_i and q
for *level* $r = 1, c, c^2, \ldots$ **do**
  for *each level-r LSH* $\mathsf{h}_i^r(\cdot)$ **do**
    identify the bucket B of bid $\mathsf{h}_i^r(q)$;
    update l_j for each point o_j in B;
  estimate μ_r according to Eqn.(10);
  if $\mu_r \geq \mu_r^*$ **then** break;
return $r \cdot \tau$ as the estimate;

4.5 Estimation Model

We start with characterizing the cornerstone of LSH, the collision probability function (CPF), which quantifies for two points with given distance their probability of colliding under a function randomly selected from a given LSH family. Concretely, a level-r LSH function as defined in Eqn.(5) follows the CPF as:

$$\mathsf{CPF}(d) = \int_0^r \frac{1}{d} f\left(\frac{t}{d}\right)\left(1 - \frac{t}{r}\right) \mathrm{d}t \tag{6}$$

where d is the distance between two points and $f(\cdot)$ is the Cauchy distribution[3]. As shown in Fig. 6, the CPF is strictly monotonic; therefore for any $d > 0$, there is a bijective mapping between d and $\mathsf{CPF}(d)$. For simplicity of discussion, we will focus on the collision probability of point o and query q rather than their distance.

Denote by ϵ_m^l the event that out of m functions, o and q collide under l of them. The likelihood of ϵ_m^l given that o and q have collision probability s is given by:

$$\mathsf{Pr}(\epsilon_m^l | s) = \binom{m}{l} s^l (1-s)^{m-l} \tag{7}$$

Thus, using Bayes' theorem, the posterior probability that o and q have collision probability s conditional on that ϵ_m^l actually happens is given by:

$$\mathsf{Pr}(s|\epsilon_m^l) = \frac{\mathsf{Pr}(\epsilon_m^l|s)\mathsf{Pr}(s)}{\int_{s'=0}^1 \mathsf{Pr}(\epsilon_m^l|s')\mathsf{Pr}(s')}$$

where $\mathsf{Pr}(s)$ represents the prior probability that o and q have collision probability s.

Assuming data points in the database have collision counts $l_1, l_2, \ldots, l_n$ with q, we are interested in estimating the number of points within distance r to q, that is, those points with collision probability $s \geq \mathsf{CPF}(r) = p_1$ with q. The expected number of such points is given by:

$$\begin{aligned} \mu_r &= \sum_{i=1}^n \left[1 \cdot \mathsf{Pr}(s \geq p_1 | \epsilon_m^{l_i}) + 0 \cdot \mathsf{Pr}(s < p_1 | \epsilon_m^{l_i})\right] \\ &= \sum_{i=1}^n \int_{s=p_1}^1 \mathsf{Pr}(s|\epsilon_m^{l_i}) \mathrm{d}s \end{aligned} \tag{8}$$

3. $f(x) = \frac{1}{\pi}\frac{1}{1+x^2}$

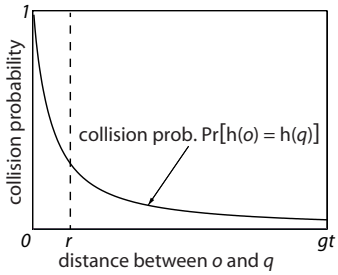

Fig. 6: Collision probability of LSH w.r.t. the distance of data points o and q.

where the linearity of expectation is applied.

The likelihood function $\mathsf{Pr}(\epsilon_m^l|s)$ (Eqn.(7)) is a binomial distribution; following the common practice of Bayesian inference, we choose Beta distribution as the prior distribution $\mathsf{Pr}(s)$: $\mathsf{Pr}(s) = s^{\alpha-1}(1-s)^{\beta-1}/B(\alpha,\beta)$, where $\alpha, \beta > 0$ are hyper-parameters and $B(\alpha,\beta)$ is the beta function to ensure that the total probability integrates to 1. We will discuss the setting of α and β shortly.

Instantiating $\mathsf{Pr}(s)$ with the Beta distribution, we obtain the closed form of $\mathsf{Pr}(s|\epsilon_m^l)$:

$$\mathsf{Pr}(s|\epsilon_m^l) = \frac{s^{l+\alpha-1}(1-s)^{m-l+\beta-1}}{B(l+\alpha, m-l+\beta)} \tag{9}$$

Furthermore, from Eqn.(8), we obtain:

$$\mu_r = \sum_{i=1}^n I_{1-p_1}(m - l_i + \beta, l_i + \alpha) \tag{10}$$

where $I_{1-p_1}(m-l_i+\beta, l_i+\alpha)$ is a regularized incomplete beta function, i.e., the cumulative distribution function of the Beta distribution.

Finally, it is desired to ensure that μ_r is above a proper threshold such that the actual number of points within distance r to q is above k. We have the following theorem which provides this threshold.

Theorem 2: Denote by n_r the actual number of points within distance r to q. Given $0 < \rho < 1$, if $\mu_r \geq \mu_r^*$, then $\mathsf{Pr}(n_r \geq k) \geq (1-\rho)$, where

$$\mu_r^* = k - \ln\rho + \sqrt{\ln^2\rho - 2k\ln\rho} \tag{11}$$

In the extremely rare case that the range suggested by OASIS contains insufficient number of points, OASIS incrementally enlarges the search range to guarantee the correctness of query results (details in § 5).

4.6 Implementation

Next we detail the efficient implementation of RO, including: finding spanning tree for feature space, counting collisions in estimation model, estimating number of neighbors, and setting parameters.

Spanning tree. As indicated in Eqn.(4), to achieve the tightest range estimation, we need to find the spanning tree with the minimum stretch w.r.t. the given network, which is referred to as the *minimum max-stretch spanning tree* problem. However, this problem is NP-hard even to approximate for unweighted planar graphs [28]. Instead,

we use minimum spanning tree (MST) as an alternative. A MST features the lowest overall length among all the spanning trees. We enumerate all the MSTs and select the one with the minimum stretch. This MST often features fairly low stretch as verified in § 7.

Counting collisions. At each level we need to count the collisions between points in the database and query q. Recall that a level-r bucket of bid x corresponds to r consecutive level-1 buckets of bids $xr, xr+1, \ldots, xr+(r-1)$. Consider a specific LSH function $\mathsf{h}^1_{\vec{a},b}(\cdot)$ which maps q to a level-1 bucket $\mathsf{h}^1_{\vec{a},b}(q)$. When increasing the level $r = 1, c, c^2, \ldots$, we can identify the set of points that collide with q under a level-cr function $\mathsf{h}^{cr}_{\vec{a},b}(\cdot)$ by expanding the bucket $\mathsf{h}^r_{\vec{a},b}(q)$ obtained in the previous round towards both "left" and "right". Specifically, the set of buckets to check have level-1 bids as:

$$\text{left:} \quad \lfloor \frac{\mathsf{h}^1_{\vec{a},b}(q)}{cr} \rfloor \cdot cr \le \text{bid} < \lfloor \frac{\mathsf{h}^1_{\vec{a},b}(q)}{r} \rfloor \cdot r$$

$$\text{right:} \quad \lfloor \frac{\mathsf{h}^1_{\vec{a},b}(q)}{r} \rfloor \cdot r + r - 1 < \text{bid} \le \lfloor \frac{\mathsf{h}^1_{\vec{a},b}(q)}{cr} \rfloor \cdot cr + cr - 1$$

A useful property of this scheme is that when checking level-cr buckets, one can skip the level-r buckets that have been checked in previous rounds.

Lookup table for μ_r. At each round, rather than strictly following Eqn.(10) to compute μ_r after all the collision counts $\{l_1, l_2, \ldots, l_n\}$ have been collected, we build a lookup table and update μ_r as we count the collisions. Concretely, it is noticed that for fixed m, α, and β, Eqn.(10) only depends on $\{l_1, l_2, \ldots, l_n\}$. Therefore, we precompute and store $I_{1-p_1}(m-l+\beta, l+\alpha)$ for each $0 \le l \le m$. When the collision count for point o_i increases from l_i to l'_i, we look up the table and update μ_r with $\Delta = I_{1-p_1}(m-l'_i+\beta, l'_i+\alpha) - I_{1-p_1}(m-l_i+\beta, l_i+\alpha)$.

Multiple approximation ratios. The default approximation ratio is $c = 2$, i.e., $r = 1, 2, 2^2, \ldots$. We may build additional indices to handle levels of $r = 1+\varepsilon, 2\times(1+\varepsilon), 2^2\times(1+\varepsilon), \ldots$ for any $0 \le \varepsilon \le 1$, which leads to tighter range estimation. For this purpose, we only need to divide all coordinates in the database by $(1+\varepsilon)$ and build the index on this transformed dataset. In our implementation, we build two sets of indices, with respective $\varepsilon = 0$ and 0.5.

Setting of α, β, and m. The simplest setting for α, β is non-informative uniform: $\alpha = \beta = 1$. With access to points uniformly sampled from the database, we may set α, β using models such as [27] to better capture statistical properties of the dataset.

However, it is observed in our empirical evaluation that the influence of α, β is fairly marginal; μ_r (cf. Eqn.(10)) estimated using the non-informative setting and that based on database sampling typically differ less than 5%, which is consistent with the observations in [27].

As indicated in Eqn.(9), m is essentially the sample size of the posterior Beta distribution. We apply the worst outcome criteria (WOC) [29], which guarantees desired posterior coverage over anticipated datasets. Specifically, we set the minimum sample size as $m^* = z^2_{\gamma/2}/p_1^2 - \alpha - \beta$, where $z_{\gamma/2}$ is the upper $\gamma/2$ endpoint of the standard normal distribution. In our implementation, we set the confidence level as 99%, i.e., $\gamma = 0.005$.

5 Boundary Index

In this section, we detail Boundary Index (BI), which is the core component following RO in the delimit-and-filter framework. For given query q and range θ of k-th NN of q as suggested by RO, BI efficiently identifies the set of candidates within NEMD θ to q.

5.1 Basic Construction

Similar to TBI scheme [12], BI exploits the dual form of the NEMD formulation to derive the lower and upper bounds of NEMD. Recall the formulation of NEMD in Eqn.(1). We derive its dual form by introducing Lagrange multipliers $\{\lambda_i\}$ over node set $\mathcal{V}$ of network G:

$$\max_{\{\lambda_i\}} \sum_{v_i \in \mathcal{V}} (o_i - q_i)\lambda_i$$
$$\text{s.t.} \begin{cases} \forall (v_i, v_j) \in \mathcal{E}: & |\lambda_i - \lambda_j| \le w_{ij} \\ \forall v_i \in \mathcal{V}: & \lambda_i \in \mathbb{R} \end{cases}$$

Following the duality theorem [30], a feasible solution $\{\lambda_i\}$ for this dual problem must satisfy the condition of

$$\sum_i (o_i - q_i)\lambda_i \le \mathsf{NEMD}(o, q) \tag{12}$$

which leads to the result below.

Lemma 4: Given a feasible solution $\{\lambda_i\}$ of the dual problem, the following inequality must hold:

$$\begin{cases} \sum_i \lambda_i o_i \le \mathsf{NEMD}(o,q) + \sum_i \lambda_i q_i \\ \sum_i \lambda_i o_i \ge \sum_i \lambda_i q_i - \mathsf{NEMD}(o,q) \end{cases} \tag{13}$$

This inequality implies a simple yet effective indexing scheme. As the left sides of the inequality only depends on o, we can use $\sum_i \lambda_i o_i$, denoted by $\phi(o)$, as the index value of o. As $\phi(o)$ is a scalar value, various off-the-shelf indexing structures (e.g., B$^+$-tree) can be readily used. We entitle this index structure as Boundary Index (BI).

Now, given query distribution q and range θ of k-th NN of q as suggested by RO, BI efficiently identifies the set of candidate data points, which must have index values within the interval of $[\phi(q)-\theta, \phi(q)+\theta]$. It is noteworthy that compared with TBI [12], BI exploits the NEMD-specific formulation, leading to much tighter intervals for candidates, i.e., constantly 2θ, which in turn imply higher pruning power.

5.2 Implementation

Next we detail the efficient implementation of BI and propose multi-fold optimizations.

Multiple indices. Similar to TBI scheme [12], multiple indices based on different feasible solutions can be employed to improve pruning power. The BI structure is implemented as a base of B$^+$-tree, $\{BT^j\}_{j=1}^n$, which

Algorithm 2: k-NN Query Processing

```
Input: query (q, k), suggested range θ, collision counts
Output: list of k-NN of q
// transformed k-NN query
list L ← ∅;
for each B+-tree BT^j do
    compute index value φ^j(q) of q;
while |L| < k do
    for each B+-tree BT^j do
        R^l ← points within [φ^j(q) − θ, φ^j(q) + θ] on BT^j;
    R ← R^1 ∩ R^2 ∩ ... ∩ R^n \ L;
    // prioritization
    sort R in descending order of collision counts;
    for each point o in R do
        // prune using filters
        prune o_i using TEMD and NCH filters (cf. § 6);
        if o is not pruned then
            compute d = NEMD(o, q);
            add (o, d) to L;
            if |L| > k then
                delete the point with largest NEMD;
                θ' ← largest NEMD in L;
                // improve estimated range
                if θ' < θ then θ ← θ';
    if |L| < k then θ ← 2θ;
return L;
```

index data points in the database. Each tree BT^j corresponds to one specific feasible solution $\{\lambda_i^j\}$. Specifically, with $\{\lambda_i^j\}$, each point o is projected to its index value $\phi^j(o)$ using the mapping: $\phi^j(o) = \sum_i \lambda_i^j o_i$. We employ a sampling scheme to generate the set of feasible solutions. In particular, each time we randomly select two points from the database and solve for them the dual form problem using the Simplex method [31].

Given range query (q, θ), we first map q to its index value $\phi^j(q)$ w.r.t. each feasible solution $\{\lambda_i^j\}$. We then construct a set of sub-queries which, on each B$^+$-tree BT^j, search for points within $[\phi^j(q)-\theta, \phi^j(q)+\theta]$. Denote by $\mathcal{R}^j$ the set of points found on BT^j. The true results must be contained in the intersection of the results of these queries, $\mathcal{R}^1 \cap \mathcal{R}^2 \cap \ldots \cap \mathcal{R}^n$.

Pruning. BI performs multi-fold filtering before resorting to exact EMD computation. Compared with existing work (e.g. [12]), BI employs two novel filters. The first one is TEMD, which exploits the lower bound of NEMD (cf. Eqn.(4)). Given query q and range θ, TEMD prunes point o if $\mathsf{TEMD}(o, q) > \tau \cdot \theta$. The second one is Network Compression Hierarchy (NCH), which leverages the network structure of feature space (details in § 6). As the computation of TEMD has linear complexity, this filter is applied ahead of NCH.

Prioritization. In addition to suggesting the range of k-th NN, RO also provides a list of collision counts for data points in the database. Recall that the number of collisions between two points implies their proximity. BI intelligently uses this list to prioritize the execution of NEMD computation, with the hope of further improving the estimated range θ during the filtering process, which can benefit the pruning of remaining points.

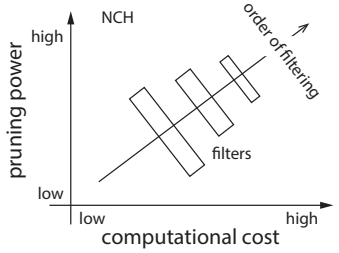

Fig. 7: Illustration of incremental filtering by NCH.

Putting everything together, Algorithm 2 sketches the process of answering k-NN queries by OASIS, which realizes the delimit-and-filter framework.

6 Network Compression Hierarchy

Next we elaborate Network Compression Hierarchy (NCH), the third core component of the delimit-and-filter framework. Taking candidates generated by BI as input, NCH effectively prunes false positive results.

While a plethora of filters have been proposed in literature, including: dimension reduced EMD (R-EMD [6]), lower bound of independent minimization ($\mathrm{LB_{IM}}$ [4]), and upper bound of feasible solution ($\mathrm{UB_P}$ [12]), which significantly reduce overall workload; none of them are optimized for the setting of network feature space. To our best knowledge, (NCH) is the first filter that exploits both the network structure of feature space and the distribution skewness of data points.

6.1 Overview of NCH

Intuitively NCH is a hierarchy of reduced dimensional representations of data points, for which their NEMD is efficiently computable. Different levels of NCH correspond to varying reduced dimensionality, thereby offering unique trade-offs between computational cost and pruning power. Given a candidate point, one may start with the level featuring the lowest computational cost and incrementally increase the pruning power if necessary. This scheme is illustrated in Fig. 7.

Specifically, NCH achieves dimensionality reduction by "compressing" the network structure of feature space, in which neighboring nodes are merged together.

Definition 3 (Compression): Let G and G' be original and compressed networks. The quantity $|G'|/|G|$ is called the *compression ratio*. We specify G' as follows.

- When merging n_i nodes $\{v_{i_x}\}_{x=1}^{n_i}$ of G into node v_i' of G' (denoted by $\{v_{i_x}\}_{x=1}^{n_i} \mapsto v_i'$), the mass of o at v_i' is the sum of masses at $\{v_{i_x}\}_{x=1}^{n_i}$: $o_i' = \sum_{x=1}^{n_i} o_{i_x}$.
- The length of edge $(v_i', v_j') \in G'$ ($\{v_{i_x}\}_{x=1}^{n_i} \mapsto v_i'$ and $\{v_{j_y}\}_{y=1}^{n_j} \mapsto v_j'$) is the minimum pairwise distance between $\{v_{i_x}\}_{x=1}^{n_i}$ and $\{v_{j_y}\}_{y=1}^{n_j}$: $w_{ij}' = \min_{i_x, j_y} w_{i_x j_y}$.

We realize the compression by a sequence of merging operations. At each step, two nodes are merged together. Concretely, consider merging v_i and v_j into v_u. Clearly, this operation only affects the nodes and edges adjacent

to v_i or v_j. Let G and G' be networks before and after this step and $\mathcal{N}_i$, $\mathcal{N}_j$, and $\mathcal{N}_{i,j}$ respectively represent the set of nodes adjacent to v_i, v_j, and both v_i and v_j in G. Then we have $o'_u = o_i + o_j$. Further, if $v_k \in \mathcal{N}_i \setminus \mathcal{N}_{i,j}$ (or $\mathcal{N}_j \setminus \mathcal{N}_{i,j}$), $w'_{uk} = w_{ik}$ (or $w'_{uk} = w_{jk}$ resp.); if $v_k \in \mathcal{N}_{i,j}$, we define $w'_{uk} = \min(w_{ik}, w_{jk})$.

Example 5: With compression ratio = $5/6$, the network in Fig. 2 is compressed into Fig. 8 by merging v_3 and v_4 into v_7. The length of edge e_9 is given by the minimum length of e_7 and e_8 in Fig. 2.

We have the next theorem which guarantees the correctness of filtering using this compressed network.

Theorem 3: Given original and compressed networks G and G', where G' is constructed following Definition 3, then we have $\mathsf{NEMD}_G(o,q) \geq \mathsf{NEMD}_{G'}(o,q)$ for any distributions o and q.

This property ensures that for query q and range θ, it is safe to prune all candidates o with $\mathsf{NEMD}_{G'}(o,q) > \theta$.

6.2 Optimal Compression

Among all compressed networks with fixed compression ratio κ, denoted by $\mathcal{G}_{G,\kappa} = \{G' | G \mapsto G', |G'|/|G| = \kappa\}$, we are particularly interested in the one with the highest pruning power. Intuitively, a compressed network has higher pruning power if it better preserves NEMD for given data points. Formally,

Definition 4 (Optimal Compression): Given compressed networks $G', G'' \in \mathcal{G}_{G,\kappa}$, we say G' is superior to G'' w.r.t. points o and q if $\mathsf{NEMD}_{G'}(o,q) > \mathsf{NEMD}_{G''}(o,q)$. The optimal compressed network G^* is the one with the largest $\mathsf{NEMD}_{G^*}(o,q)$, i.e., $G^* = \arg\max_{G' \in \mathcal{G}_{G,\kappa}} \mathsf{NEMD}_{G'}(o,q)$.

Note that this optimality is data dependent. We thus incorporate knowledge of the underlying database. Recall that NEMD is essentially the weighted sum of a set of flows (cf. Eqn.(1)). We sample a set of points $\mathcal{S}$ from the database and calculate their aggregated flows to guide the process of deriving the compressed network. Specifically, for points $o, o' \in \mathcal{S}$, denote by $f^*_{ij}(o,o')$ the flow over edge (v_i, v_j) as dictated by $\mathsf{NEMD}(o,o')$. The non-directional, aggregated flow of $\mathcal{S}$ over (v_i, v_j) is defined as $\widehat{f_{ij}} = \sum_{o,o' \in \mathcal{S}} |f^*_{ij}(o,o')|$. The aggregated NEMD w.r.t. $\mathcal{S}$ is then given by $\mathsf{AEMD}_G(\mathcal{S}) = \sum_{(v_i,v_j)} \widehat{f_{ij}} w_{ij}$. We intend to identify $G^* \in \mathcal{G}_{G,\kappa}$ that maximizes $\mathsf{AEMD}_{G^*}(\mathcal{S})$.

Finding the exact optimal compressed network requires computing $\mathsf{AEMD}_{G'}(\mathcal{S})$ for every network $G' \in \mathcal{G}_{G,\kappa}$, which is infeasible for large-scale networks. Instead, we propose an effective greedy algorithm that iteratively identifies high-quality compressed networks.

Recall that network compression consists of a sequence of merging steps. W.l.o.g., consider the step of merging nodes v_i, v_j into v_u. Let G and G' be networks before and after this step and $\widehat{f}$ and $\widehat{f}'$ be their aggregated flows. We can approximate $\widehat{f}'$ as follows. For each node $v_k \in \mathcal{N}_{i,j}$, the aggregated flow over new edge (v_k, v_u) is given as: $\widehat{f'_{ku}} = \widehat{f_{ki}} + \widehat{f_{kj}}$, while the aggregated flows over all other edges remain unchanged.

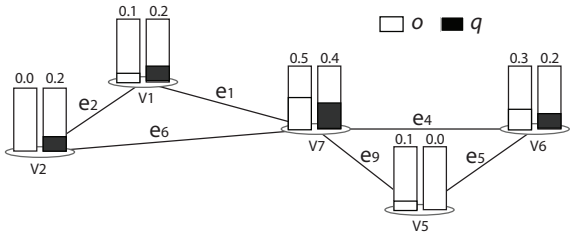

Fig. 8: Compression of network feature space.

Then following Definition 3, the difference between $\mathsf{AEMD}_G(\mathcal{S})$ and $\mathsf{AEMD}_{G'}(\mathcal{S})$ is approximated by:

$$\Delta^G_{i,j} = w_{ij} \cdot \widehat{f_{ij}} - \sum_{v_k \in \mathcal{N}^G_{i,j}} \min(w_{ki}, w_{kj}) \cdot (\widehat{f_{ki}} + \widehat{f_{kj}}) + \sum_{v_k \in \mathcal{N}^G_{i,j}} (w_{ki} \cdot \widehat{f_{ki}} + w_{kj} \cdot \widehat{f_{kj}}) \quad (14)$$

Here the first term corresponds to deleting edge (v_i, v_j); the rest terms correspond to replacing (v_i, v_k) and (v_j, v_k) with (v_u, v_k). The optimal node pair (v_i, v_j) to be merged should be the one which minimizes $\Delta^G_{i,j}$.

Algorithm 3: Network Compression

Input: original network G, compression ratio κ, sample points $\mathcal{S}$
Output: compressed network $G' \in \mathcal{G}_{G,\kappa}$
Notations: L: stack of edges in order of delta AEMD
`// initialization`
$G' \leftarrow G$, $L \leftarrow \emptyset$;
for *each edge* $(v_i, v_j) \in G'$ **do**
  compute $\Delta^{G'}_{i,j}$ following Eqn.(14);
  insert (v_i, v_j) into L in ascending order of $\Delta^{G'}_{i,j}$;
while $|G'| > \kappa|G|$ **do**
  `// edge with minimum delta AEMD`
  pop top edge (v_i, v_j) from L;
  `// merge` v_i, v_j `as` v_u
  for $v_k \in \mathcal{N}_{i,j}$ **do**
    $\widehat{f_{ku}} \leftarrow \widehat{f_{ki}} + \widehat{f_{kj}}$;
    $w_{ku} \leftarrow \min(w_{ki}, w_{kj})$;
  update G' and L by merging v_{i^*} and v_{j^*} as v_u;
return G' as compressed network;

Algorithm 3 sketches how to derive the compressed network. Starting from original network G, it iteratively merges the optimal node pair until the specified compression ratio is reached. Given compressed network G', we can further compress G' to generate more compressed networks. Alternatively, we can first compress original network G to a highly compressed network G' and incrementally "decompress" G' to form a hierarchy of networks. Note that we only materialize the mapping from G to each compressed network in this hierarchy. When performing filtering using compressed network G', we project both candidate point o and query point q to the space of G' and calculate $\mathsf{NEMD}_{G'}(o,q)$.

7 Empirical Evaluation

In this section we empirically evaluate the performance of OASIS against existing methods as benchmarks. Specif-

	# point	# dimension	NCH ratio	NCH height
DBLP	28K	20	N/A	N/A
Cabspotting	200K	70	N/A	N/A
IDPS	1.24M	117	0.6	2
Small-world	1M	320	0.75	3

TABLE 1: Summary of real and synthetic datasets (data cardinality, data dimensionality, NCH compression ratio, and NCH hierarchy height).

ically, § 7.1 describes the datasets and queries, the assessment metrics, and the techniques to be evaluated. § 7.2 and § 7.3 evaluate in detail the three core components of OASIS, namely, Range Oracle, Boundary Index, and Network Compression Hierarchy. § 7.4 demonstrates that OASIS significantly outperforms the state-of-the-art techniques in processing k-NN queries and also explores the strengths and limitations of OASIS.

7.1 Experimental Setting

Datasets and Queries

We use four datasets in the evaluation including three real datasets: *Cabspotting*[4], *IDPS*, and *DBLP* [32], and a synthetic dataset: *Small-world*.

Cabspotting. This dataset contains the GPS trajectories of approximately 500 taxis collected over a period of 30 days in the San Francisco Bay Area. We map the traces to the underlying road network and randomly select 70 road junctions on the highways as "checkpoints". For an interval of every 10 seconds, we count the number of taxis passing these checkpoints and generate the traffic distributions using these counts. The distance between two checkpoints is measured by their distance over the road network in miles.

IDPS. This dataset contains the alert events raised by an Intrusion Detection and Prevention System deployed in a large enterprise network over a period of 2 years. For totally 117 distinct alert types, we construct a semantic network similar to [16]: for each alert type, we extract the words used in its specification; the distance of two alert types is computed based on their word sets[5] and an edge is added if either one appears in the other's k-NN (k = 10 in our implementation). We generate the dataset by computing the distribution of alerts over these alert types for an interval of every 1 hour.

DBLP. This dataset contains a subset of DBLP records that belong to four relevant areas: database, data mining, information retrieval, and artificial intelligence. It includes the publication records of 28,702 authors in 20 top conferences from these four areas. The words used in the paper abstracts from these conferences are also collected. The derivation of the "conference network" is similar to the *IDPS* dataset. Each data point corresponds to the publication distribution of a particular author in these conferences. The dataset thus contains 28,702 data points, each represented as a 20-dimensional vector.

Small-world. We use the Barabási-Albert model [33] to generate a random network of 320 nodes, wherein all edges are of uniform length. Over this network, we generate 1,000,000 data points, with each coordinate value randomly sampled from the power law distribution (with $a = 2$ in our implementation): $\mathsf{Pr}(x) \propto x^{-a}$. The coordinate values of each data point are then normalized to ensure that they sum up to 1.

The statistics of datasets are summarized in Table 1. Furthermore, for the *IDPS* and *Small-world* datasets, we apply NCH with compression ratio and hierarchy height listed in Table 1. For each dataset, we randomly select 100 data points to form the query set.

Assessment Metrics and Alternative Techniques

We compare the performance of alternative techniques by measuring their query processing time, I/O cost, and number of exact EMD computation (refinement). Moreover, for OASIS we also evaluate its extra overhead in terms of preprocessing time and index size.

To our best knowledge, TBI [12] is the state-of-the-art solution to process EMD-based range and k-NN queries, which is nearly two times faster than the traditional scan-and-refine methods (e.g. [6]). Although improved over TBI, Normal Index [13] relies on the assumption of L^p feature spaces, which makes it inapplicable for the setting of NEMD. We therefore mainly evaluate OASIS against the TBI-based methods.

For each method we implement two variants: a basic version only using the B$^+$-tree index and an enhanced version applying a set of filters (NCH for OASIS and R-EMD [6], $\mathrm{LB_{IM}}$ [4], and $\mathrm{UB_P}$ [12] for TBI) before performing exact EMD computation. We use subscriptions $-b$ and $-e$ to differentiate basic and enhanced variants. All methods are implemented in C++ and compiled using GCC 4.2. All experiments are performed on a Linux box running Intel i7 2.6GHz CPU with 16GB RAM.

7.2 Experiments on BI and NCH

The first set of experiments evaluates the BI and NCH components of OASIS. Concretely, we compare the performance of OASIS and TBI in handling range queries. Fig. 9 depicts the average query processing time and the number of EMD refinement used by different methods under varying similarity threshold θ. It is observed that OASIS-b significantly outperforms TBI-b across all the datasets. As both methods only rely on B$^+$-tree indices (i.e., without the help of filters), this indicates that the indexing structure of OASIS is more effective. This is explained by that TBI uses the classical bipartite-flow formulation of EMD, while OASIS adopts a formulation specific to NEMD, which leads to tighter bounds for candidates (cf. § 5). The application of filters considerably reduces the required cost of full EMD computation.

4. http://cabspotting.org

5. For alert types t_1 and t_2, denote by $N_{t_i}(w)$ the count of word w in the specification of alert type t_i ($i = 1, 2$). The distance of t_1 and t_2 is defined as: $d(t_1, t_2) = 1 - \frac{\sum_w N_{t_1}(w) \times N_{t_2}(w)}{\sqrt{\sum_w N^2_{t_1}(w)} \times \sqrt{\sum_w N^2_{t_2}(w)}}$.

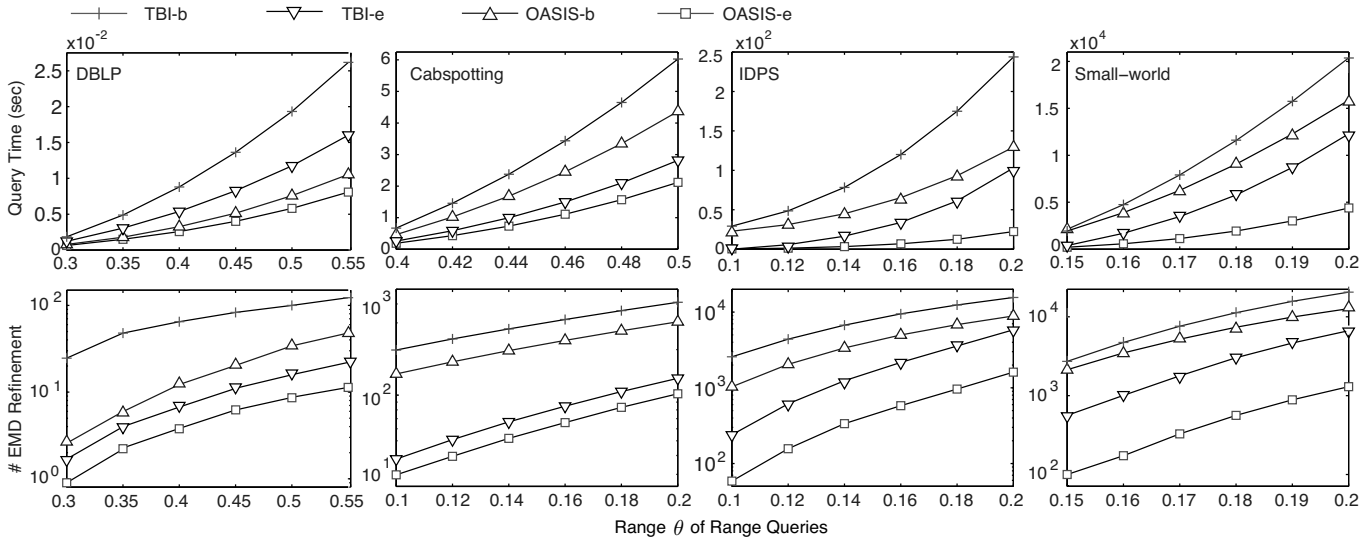

Fig. 9: Average query processing time and number of EMD refinement under varying similarity threshold θ.

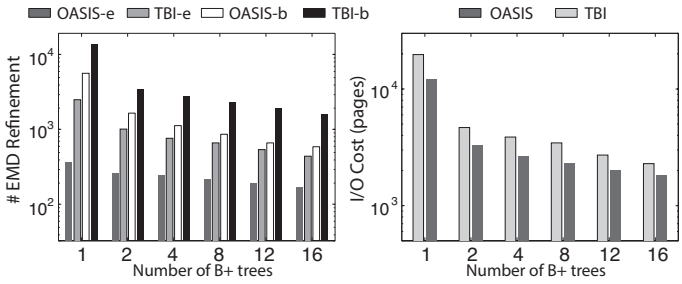

Fig. 10: Average number of EMD refinement and I/O cost under varying number of B$^+$-tree.

Across all the cases, OASIS-e and TBI-e save orders-of-magnitude of EMD refinement over their basic variants. However, for *IDPS* and *Small-world* datasets over which NCH is applied, OASIS-e outperforms TBI-e by 4 to 5 times in terms of both query processing time and EMD refinement cost. This is attributed to that NCH effectively exploits both the network structure of feature space and the distribution skewness of data points (cf. § 6). Meanwhile, it is noticed that the relative cost of pruning itself can be expensive, especially in the case of low dimensional data and multiple filters. For example, for *DBLP* dataset (20 dimensions), while OASIS-b performs more full EMD computation than TBI-e, it still uses less query processing time than the latter.

One possible way to reduce the pruning cost is to employ multiple B$^+$-tree in conjunction, as the true results must be contained in the intersection of t candidate sets of all the trees (cf. § 5.2). Further, this may also reduce the I/O cost as less data points need to be loaded for pruning or full EMD computation. Fig. 10 shows how the EMD computation cost and the I/O cost decrease with the number of B$^+$-tree for *IDPS* dataset (with θ fixed as 0.2). We have the following observations. (i) For both OASIS and TBI, using multiple B$^+$-tree significantly reduces the number of candidates; the decreasing rate becomes flatter with the growing number of trees. (ii) The use of filters to a certain extent "dilutes" the impact of B$^+$-tree on the EMD refinement cost. (iii) Compared with alternative methods, OASIS-e is much less sensitive to the number of B$^+$-tree. Thus, we can conclude that using OASIS-e, we are able to use only a small number of trees (e.g., 2 in our implementation) to achieve both low pruning cost and low EMD refinement cost.

7.3 Experiments on RO

As OASIS is built upon the unique component of Range Oracle (RO), we separately assess its efficacy. More concretely, we examine the quality of range suggested by RO. Let $r_k^*(q)$ and $r_k(q)$ be the actual NEMD of k-th NN of query q and that suggested by RO. We measure the quality of range estimation using the rank-k ratio: $R_k(q) = r_k(q)/r_k^*(q)$. It is noteworthy however that as RO operates in the transformed L^1 space and essentially predicts the range of k-th NN in this space (denoted by $r'_k(q)$), thus the ratio $r'_k(q)/r_k^*(q)$ is the ideal ratio that RO is possibly to achieve.

	k	4	8	16	32	64
DBLP	RO	1.9589	1.9140	1.8228	1.8113	1.7650
	Ideal	1.6455	1.6401	1.6148	1.5480	1.5320
Cabspotting	RO	1.3346	1.2895	1.2838	1.2605	1.2590
	Ideal	1.2542	1.2485	1.2471	1.2434	1.2404
IDPS	RO	2.7633	2.4752	2.4591	2.2484	2.0750
	Ideal	2.0635	2.0441	2.0397	2.0208	2.0120
Small-world	RO	2.0346	1.9455	1.9448	1.8947	1.8558
	Ideal	1.7496	1.7394	1.7262	1.7253	1.7196

TABLE 2: Quality of k-NN ranges suggested by RO.

Table 2 compares the ideal ratios and that given by RO under varying k for the four datasets. It is clear that across all the cases, RO achieves ratios fairly close to optimal ones. For example, the relative estimation error is less than 15% for $k = 64$. Note that theoretically it is possible to further improve the estimation accuracy by using more LSH functions and different approximation ratios (cf. § 4.5), which however incurs additional I/O and storage cost. As will be shown shortly, the ratios shown in Table 2 suffice to significantly boost the efficiency of answering k-NN queries.

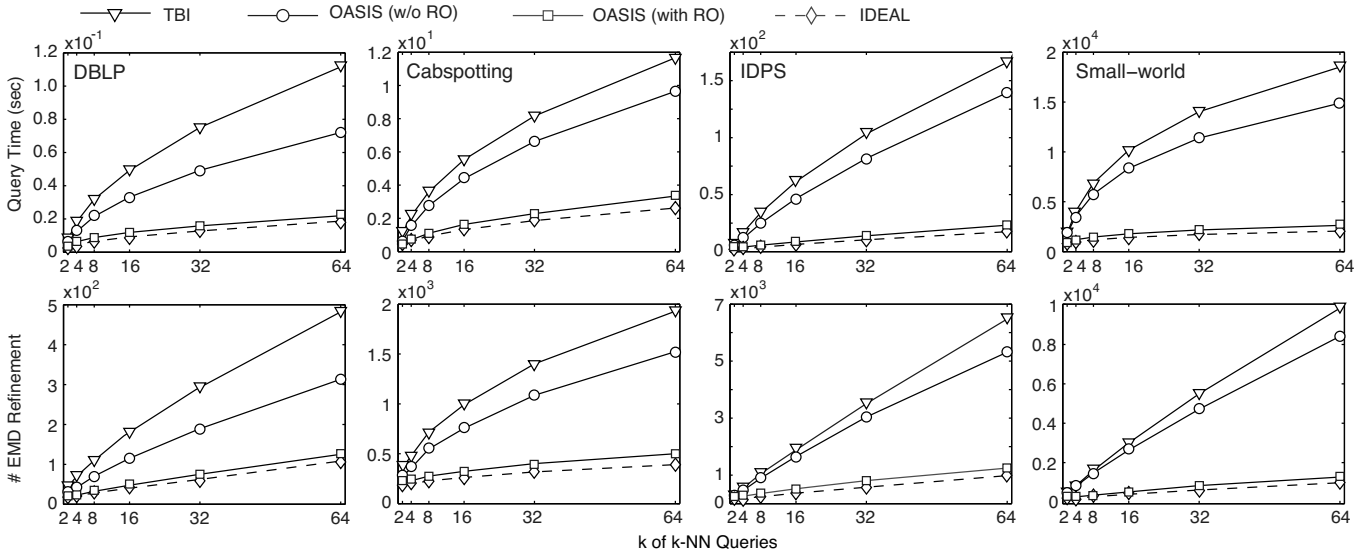

Fig. 11: Average query processing time and number of EMD computation under varying k of k-NN queries.

	m	Proc. Time (min)	Index Size (MB)
DBLP	128	0.55	35.24
Cabspotting	256	3.35	104.98
IDPS	398	27.24	501.93
Small-world	398	125.05	5629.24

TABLE 3: Preprocessing time and index size of RO.

We also measure the overhead of RO, which mainly consists of the time cost of precomputing the hash values of data points and the space cost of RO indexing structure. Table 3 summarizes the number of hash functions, the preprocessing time, and the index size of RO for each dataset. It is observed that RO incurs reasonable costs across all four datasets. Also note that the preprocessing time is a one-time cost.

7.4 Experiments on k-NN Query Processing

Having evaluated the performance of individual components of OASIS, next we measure their overall effectiveness in processing k-NN queries.

More specifically, we implement four competing methods: (1) TBI-e, (2) OASIS-e (without RO), which uses the scheme in [12], (3) OASIS-e (with RO), and (4) the ideal method, which is an OASIS-e implementation but with access to ground truth of the range of k-th NN. Fig. 11 shows the average query processing time and the number of EMD refinement used by the four methods as k of k-NN queries varies from 2 to 64. As expected, with the help of RO, OASIS saves a large number of EMD refinement, which is especially evident when handling high-dimensional data. For example, for *IDPS* dataset (117 dimensions), OASIS with RO is about 2 times faster than the variant without RO. For all four datasets, with the help of RO, OASIS achieves performance close to the ideal method. This is explained by that (i) the range estimation of k-th NN by RO is fairly tight (cf. Table 2) and (ii) RO not only suggests the range of k-th NN but also helps prioritize NEMD computation for data points more likely to be proximate to query points, which may further improve the suggested range and benefit the pruning (cf. § 5.2).

In the last set of experiments, we explore the strengths and limitations of OASIS. Concretely, we intend to understand for what type of datasets OASIS works best (or worst). To answer this question, we generate a set of synthetic datasets by varying the following four metrics: (i) data cardinality, (ii) data dimensionality, (iii) network density, and (iv) distribution skewness. In particular, let $|\mathcal{V}|$ and $|\mathcal{E}|$ denote the number of nodes and edges of feature space; the network density is measured by ratio $\frac{2|\mathcal{E}|}{|\mathcal{V}|(|\mathcal{V}|-1)}$. The distribution skewness is controlled by parameter a of the power law formula [33], with smaller a implying larger skewness.

Fig. 12 illustrates the impact of these metrics on the average query processing time by alternative methods. In each set of experiments, we vary one metric while keeping the rest fixed. The default setting is: data cardinality = 1M, dimensionality = 320, network density = 0.1, $a = 2$, and $k = 32$ for k-NN queries. We can obtain the following observations. First, all four methods scale well with data cardinality. Second, in comparison, data dimensionality has larger influence on the performance of all four methods, mainly due to its impact on the complexity of EMD refinement. Nevertheless, following the delimit-and-filter paradigm, OASIS (with RO) scales gracefully by saving on the number of EMD refinement. Third, the performance of OASIS gradually degrades as network density increases. This is explained by: optimized for network flow-based EMD formulation with variables defined over network edges (cf. § 4), OASIS works best for sparse networks (with smaller number of edges). Finally, the query processing time of all four methods decreases as the distribution skewness grows. This may be intuitively explained by: the probability mass of a skewed distribution is dominated by a few nodes (features); distributions are either fairly proximate to or well separated from each other. It is thus easier

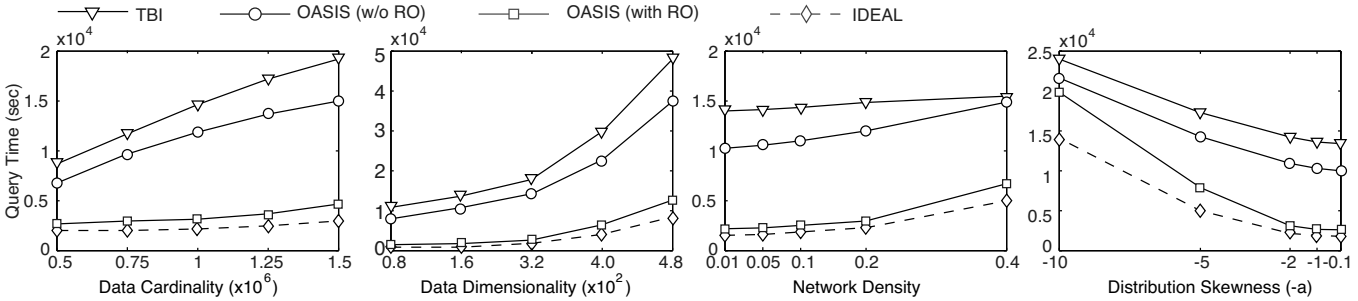

Fig. 12: Impact of various characteristics of datasets on average query processing time by alternative methods.

to narrow down to a small number of candidates when processing k-NN queries. Moreover, for OASIS, distribution skewness may also benefit the range estimation of RO as distant distributions under NEMD tend to be well separable in transformed L^1 space as well.

8 CONCLUSIONS

Indexing network-based Earth Mover's Distance metrics (NEMD) to support similarity search benefits a spectrum of applications. This work presents OASIS, a new access method which leverages the network structure of feature space and enables efficient k-NN query processing under the NEMD metrics. We argue that the proposed method carries both theoretical and practical significance. In theory, it stands for a new delimit-and-filter paradigm, which overcomes several drawbacks of the conventional enumerate-and-filter framework; in practice, it considerably outperforms the state-of-the-art indexing methods for NEMD-based similarity search.

APPENDIX

Proofs of Lemmas and Theorems

(Lemma 1): (Sketch) Consider edge (v_i, v_j) with $v_i \prec v_j$. For each pair of nodes in T_i, their shortest path in T comprises only edges in T_i. Therefore the minimum cost flow on (v_i, v_j) is simply the flow that just balances $\{o_i\}$ and $\{q_i\}$ for all nodes in $\mathcal{V}_{T_i}$, which is given by Eqn.(2). The NEMD between o and q is thus the summation of such flows over all the edges. □

(Lemma 2): Note that the i-th row of P, dented by $[P]_{i,\cdot}$, encodes the precedence relationship between v_i and the rest nodes in the subtree T_i. Therefore we have: $[P]_{i,\cdot}(o-q) = w_i \sum_{v_j \in \mathcal{V}_{T_i}} (o_j - q_j)$, which is equivalent to Eqn.(2). The L^1-norm then sums up over all nodes. □

(Theorem 1): W.l.o.g., consider the case that $k = 1$, i.e., NN. Following that o_e and o_l are respectively the NN of q in the NEMD and transformed L^1-norm spaces, we have that (i) $\mathsf{NEMD}(o_e, q) \leq \mathsf{NEMD}(o_l, q)$ and (ii) $\mathsf{L1}(o_l, q) \leq \mathsf{L1}(o_e, q)$.

Now, applying the inequality in Eqn.(4), we obtain:

$$\mathsf{NEMD}(o_e, q) \geq \mathsf{L1}(o_e, q) \geq \mathsf{L1}(o_l, q)$$
$$\mathsf{NEMD}(o_e, q) \leq \mathsf{NEMD}(o_l, q) \leq \tau \mathsf{L1}(o_l, q)$$

This scenario is illustrated in Fig. 4. It is straightforward to generalize this conclusion to the case of k-th NN with $k > 1$. □

(Theorem 2): We introduce a set of Bernoulli variables $\{b_i\}_{i=1}^n$ over the data points $\{o_i\}_{i=1}^n$:

$$b_i = \begin{cases} 1 & o_i \text{ is within distance } r \text{ to } q \\ 0 & \text{otherwise} \end{cases}$$

which is conditional on that o_i has collision number l_i.

Following Eqn.(9), we have $\Pr(b_i = 1) = I_{1-p_1}(m - l_i + \beta, l_i + \alpha)$. Clearly, for given $\{l_i\}_{i=1}^n$, $\{b_i\}_{i=1}^n$ is a set of independent random variables.

Observe that $n_r = \sum_{i=1}^n b_i$ and $\mu_r = \mathbb{E}[n_r]$. According to the Chernoff bound, for $0 < \delta < 1$, $\Pr(n_r < (1-\delta)\mu_r) \leq \exp\left(-\delta^2 \mu_r / 2\right)$. Let $(1-\delta)\mu_r^* = k$ and $\exp\left(-\delta^2 \mu_r^*/2\right) = \rho$, we obtain: $\mu_r^* = k - \ln\rho + \sqrt{\ln^2 \rho - 2k \ln \rho}$. Thus, for any $\mu_r \geq \mu_r^*$, we have $\Pr(n_r \geq k) \geq 1 - \rho$. □

(Lemma 4): We use the symmetry of both NEMD and its dual form, i.e., (i) $\mathsf{NEMD}(o, q) = \mathsf{NEMD}(q, o)$, and (ii) if $\{\lambda\}$ is feasible for $\mathsf{NEMD}(o, q)$, it must also be feasible for $\mathsf{NEMD}(q, o)$. Therefore, $|\sum_i (o_i - q_i)\lambda_i| \leq \mathsf{NEMD}(o, q)$, which leads to the bounds in Lemma 4 trivially. □

(Theorem 3): Let f_{ij} be the flow over edge (v_i, v_j) in G. We say that the set of flows $\{f_{ij}\}_{i,j}$ is valid if it balances the density of o and q at every node. Further, let $\{f_{ij}^*\}_{i,j}$ denote the set of flows corresponding to $\mathsf{NEMD}_G(o, q)$, which is by definition valid w.r.t. G.

For any two nodes v_i' and v_j' in compressed network G' (with $\{v_{i_x}\}_{x=1}^{n_i} \mapsto v_i'$ and $\{v_{j_y}\}_{y=1}^{n_j} \mapsto v_j'$), we define the flow f_{ij}' over edge (v_i', v_j') as the aggregation of optimal flows between $\{v_{i_x}\}_{x=1}^{n_i}$ and $\{v_{j_y}\}_{y=1}^{n_j}$: $f_{ij}' = \sum_{i_x, j_y} f_{i_x j_y}^*$. It is verifiable that the set of flows $\{f_{ij}'\}_{i,j}$ is valid w.r.t. G'. To see this, we sum up the flows relevant to v_i':

$$\sum_{v_j' \in \mathcal{N}_i^{G'}} f_{ij}' = \sum_{v_{i_x}} \sum_{v_{j_y} \in \mathcal{N}_{i_x}^G} f_{i_x j_y}^* = \sum_{v_{i_x}} (q_{i_x} - o_{i_x}) = q_i' - o_i'$$

That is, $\{f_{ij}'\}$ balance the density of o and q at every node of G'. Following Eqn.(1) and Definition 3, we have:

$$\begin{aligned} \mathsf{NEMD}_{G'}(o, q) &\leq \sum_{(v_i', v_j') \in G'} |f_{ij}'| w_{ij}' \\ &= \sum_{(v_i', v_j') \in G'} |\sum_{i_x, j_y} f_{i_x j_y}^*| \min_{i_x, j_y} w_{i_x j_y} \\ &\leq \sum_{(v_i', v_j') \in G'} \sum_{i_x, j_y} |f_{i_x j_y}^*| w_{i_x j_y} \\ &= \sum_{(v_{i_x}, v_{j_y}) \in G} |f_{i_x j_y}^*| w_{i_x j_y} \\ &= \mathsf{NEMD}_G(o, q) \end{aligned}$$

□

Extension to Other Network Distance Metrics

One potential solution to extend the proposed method to support random walk-based distance metrics (e.g., Personalized PageRank, SimRank, Commute Time, Hitting Time) can be as follows. (1) In RO, embed random walk metrics into Euclidean space [34] and then follow [10] to bound NEMD as L^1 distance; (2) In BI, use the bounds of general EMD [12]; and (3) In NCH, compress original network into smaller ones, such that random walk distance over original network is tightly bounded by its counterpart over compressed networks. After this, the "delimit-and-filter" framework is directly applicable.

REFERENCES

[1] C. S. Jensen and S. Pakalnis, "Trax: Real-world tracking of moving objects," in *VLDB*, 2007.

[2] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "The design of an acquisitional query processor for sensor networks," in *SIGMOD*, 2003.

[3] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan, "A data-based approach to social influence maximization," *Proc. VLDB Endow.*, vol. 5, no. 1, pp. 73–84.

[4] I. Assent, A. Wenning, and T. Seidl, "Approximation techniques for indexing the earth mover's distance in multimedia databases," in *ICDE*, 2006.

[5] I. Assent, M. Wichterich, T. Meisen, and T. Seidl, "Efficient similarity search using the earth mover's distance for large multimedia databases," in *ICDE*, 2008.

[6] M. Wichterich, I. Assent, P. Kranen, and T. Seidl, "Efficient emd-based similarity search in multimedia databases via flexible dimensionality reduction," in *SIGMOD*, 2008.

[7] D. Zhou, J. Li, and H. Zha, "A new mallows distance based metric for comparing clusterings," in *ICML*, 2005.

[8] S. Cabello, P. Giannopoulos, C. Knauer, and G. Rote, "Matching point sets with respect to the earth mover's distance," *Comput. Geom. Theory Appl.*, vol. 39, no. 2, pp. 118–133.

[9] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *Int. J. Comput. Vision*, vol. 40, no. 2, pp. 99–121.

[10] S. Cohen and L. Guibas, "The earth mover"s distance: Lower bounds and invariance under translation," Stanford University, Tech. Rep., 1997.

[11] E. Verbin and Q. Zhang, "Rademacher-sketch: A dimensionality-reducing embedding for sum-product norms, with an application to earth-mover distance," in *ICALP*, 2012.

[12] J. Xu, Z. Zhang, A. K. H. Tung, and G. Yu, "Efficient and effective similarity search over probabilistic data based on earth mover's distance," *Proc. VLDB Endow.*, vol. 3, no. 1-2, pp. 758–769.

[13] B. E. Ruttenberg and A. K. Singh, "Indexing the earth mover's distance using normal distributions," *Proc. VLDB Endow.*, vol. 5, no. 3, pp. 205–216.

[14] Y. Tang, L. Hou U, Y. Cai, N. Mamoulis, and R. Cheng, "Earth mover's distance based similarity search at scale," *Proc. VLDB Endow.*, vol. 7, no. 1-2, pp. 758–769.

[15] R. Steinbach, P. Edwards, and C. Grundy, "The road most travelled: the geographic distribution of road traffic injuries in england," *International Journal of Health Geographics*, vol. 12, no. 30.

[16] Q. Mei, D. Zhang, and C. Zhai, "A general optimization framework for smoothing language models on graph structures," in *SIGIR*, 2008.

[17] T. Wang, M. Srivatsa, D. Agrawal, and L. Liu, "Microscopic social influence," in *SDM*, 2012.

[18] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *SCG*, 2004.

[19] A. Andoni, K. D. Ba, P. Indyk, and D. P. Woodruff, "Efficient sketches for earth-mover distance, with applications." in *FOCS*, 2009.

[20] M.-H. Jang, S.-W. Kim, C. Faloutsos, and S. Park, "A linear-time approximation of the earth mover's distance," in *CIKM*, 2011.

[21] G. Cormode, M. Garofalakis, P. J. Haas, and C. Jermaine, "Synopses for massive data: Samples, histograms, wavelets, sketches," *Found. Trends databases*, vol. 4, no. 1-3, pp. 1–294.

[22] A. McGregor and D. Stubbs, "Sketching earth-mover distance on graph metrics," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, ser. Lecture Notes in Computer Science, vol. 8096, pp. 274–286.

[23] J. Brody, H. Liang, and X. Sun, "Space-efficient approximation scheme for circular earth mover distance," in *LATIN*, 2012.

[24] B. Brinkman and M. Charikar, "On the impossibility of dimension reduction in l1," *J. ACM*, vol. 52, no. 5, pp. 766–788.

[25] Y. Tao, K. Yi, C. Sheng, and P. Kalnis, "Quality and efficiency in high dimensional nearest neighbor search," in *SIGMOD*, 2009.

[26] J. Gan, J. Feng, Q. Fang, and W. Ng, "Locality-sensitive hashing scheme based on dynamic collision counting," in *SIGMOD*, 2012.

[27] V. Satuluri and S. Parthasarathy, "Bayesian locality sensitive hashing for fast similarity search," *Proc. VLDB Endow.*, vol. 5, no. 5, pp. 430–441.

[28] S. P. Fekete and J. Kremer, "Tree spanners in planar graphs," *Discrete Appl. Math.*, vol. 108, no. 1, pp. 85–103.

[29] C. E. M'Lan, L. Joseph, and D. B. Wolfson, "Bayesian sample size determination for binomial proportions," *Bayesian Analysis*, vol. 3, no. 2, pp. 269–296.

[30] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[31] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Inc., 1982.

[32] H. Deng, J. Han, B. Zhao, Y. Yu, and C. X. Lin, "Probabilistic topic models with biased propagation on heterogeneous information networks," in *KDD*, 2011.

[33] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 5439, pp. 509–512.

[34] X. Zhao, A. Chang, A. D. Sarma, H. Zheng, and B. Y. Zhao, "On the embeddability of random walk distances," *Proc. VLDB Endow.*, vol. 6, no. 14, pp. 1690–1701.

Ting Wang is a Research Staff Member at IBM Thomas J. Watson Research Center. Prior to joining IBM, he received his Ph.D. degree from Georgia Institute of Technology and B.S. degree from Zhejiang University, both in Computer Science. His research interests span both theoretical foundations and real-world applications of large-scale data mining and management. His current work focuses on data analytics for privacy and security.

Shicong Meng obtained his PhD in Computer Science from Georgia Tech. He also worked at IBM TJ Watson Research Center as a research staff member. His research focuses on performance, scalability and security issues in large-scale distributed systems. He has worked on cloud datacenter monitoring, cross-datacenter transactional systems and in-memory high performance key-value stores. He currently works at Facebook, Inc.

Jiang Bian is a Researcher at Microsoft Research. Prior to that, he was a scientist at Yahoo! Labs. He obtained his PhD degree in Computer Science from Georgia Institute of Technology, Atlanta GA, in 2010. His research interests include machine learning, information retrieval, data mining, social network analysis, and computational advertising. More details of his research and background can be found at https://sites.google.com/site/jiangbianhome/.