Earth Mover's Distance based Similarity Search at Scale

Yu Tang[†], Leong Hou U[‡], Yilun Cai[†], Nikos Mamoulis[†], Reynold Cheng[†]

[†]The University of Hong Kong [‡]University of Macau

[†]{ytang, ylcai, nikos, ckcheng}@cs.hku.hk [‡]ryanlhu@umac.mo

ABSTRACT

Earth Mover's Distance (EMD), as a similarity measure, has received a lot of attention in the fields of multimedia and probabilistic databases, computer vision, image retrieval, machine learning, etc. EMD on multidimensional histograms provides better distinguishability between the objects approximated by the histograms (e.g., images), compared to classic measures like Euclidean distance. Despite its usefulness, EMD has a high computational cost; therefore, a number of effective filtering methods have been proposed, to reduce the pairs of histograms for which the exact EMD has to be computed, during similarity search. Still, EMD calculations in the refinement step remain the bottleneck of the whole similarity search process. In this paper, we focus on optimizing the refinement phase of EMD-based similarity search by (i) adapting an efficient min-cost flow algorithm (SIA) for EMD computation, (ii) proposing a dynamic distance bound, which can be used to terminate an EMD refinement early, and (iii) proposing a dynamic refinement order for the candidates which, paired with a concurrent EMD refinement strategy, reduces the amount of needless computations. Our proposed techniques are orthogonal to and can be easily integrated with the state-of-the-art filtering techniques, reducing the cost of EMD-based similarity queries by orders of magnitude.

1. INTRODUCTION

Given two histograms (e.g., probability distributions), their *Earth Mover's Distance* (EMD) is defined as the minimum amount of work to transform one histogram into the other. EMD is robust to outliers and small shifts of values among histogram bins [20], improving the quality of similarity search in different domain areas, such as computer vision [19, 21], machine learning [6, 9], information retrieval [23, 24], probabilistic [25, 32] and multimedia databases [5, 30]. Typically, the EMD between two histograms is modeled and solved as a linear optimization problem, the *min-cost flow problem*, which requires super-cubic time. The high computational cost of EMD restricts its applicability to datasets of low-scale. For example, in computer vision applications, the quality of results is typically compromised by the use of low-granularity histograms, to render EMD-based similarity retrieval feasible [22, 30].

Proceedings of the VLDB Endowment, Vol. 7, No. 4

EMD-based similarity search has been extensively studied in [5, 25, 30, 32]. Given a query histogram $\mathbf{q}$ and a database of histogram objects $\mathcal{D}$, the objective is to find the k nearest neighbors of $\mathbf{q}$ in $\mathcal{D}$. All these works adopt the *filter-and-refinement* framework; to evaluate a query, unpromising objects (or object groups) are filtered out, by utilizing various effective EMD lower bound estimations, based on centroids and projections [7], dimensionality reduction [30], primal-dual space [32], normal distributions [25], etc. Actual EMD calculations[1] are applied only between $\mathbf{q}$ and all objects that survive the filter step. Thus, the primary focus of previous research has been in tightening the lower bounds such that more objects can be pruned at the filter step. For instance, [25] demonstrate that the projection-based lower bound can be up to 90% of the actual EMD. However, the effectiveness of a lower bound largely depends on various factors, such as the dimensionality and granularity of histograms, the data distribution, and the parameters of the similarity query (e.g., k). In particular, for large-scale datasets (e.g., 1M cardinality and/or 1K histogram dimensionality), the current state-of-the-art solution [25] is not feasible, due to the extreme cost of the refinement step. For instance, based on the experiments in [25], it may take 10 minutes[2] to complete one k-NN query on a dataset with 1M objects even when 99% of objects are filtered out.

In view of this, we focus on optimizing the *refinement phase* of EMD-based similarity search. Calculating the EMD between two object histograms is equivalent to finding the *minimum-cost flow* (MCF) in a bipartite network, where each vertex indicates a histogram bin and edges connect bins from different object histograms. Techniques from operations research [1], such as network simplex, primal-dual, successive shortest path, and cost-scaling can be used to solve MCF. However, these solutions do not scale well with the number of histogram bins since their computations rely on a complete bipartite network. For example, consider two histograms having 1K bins each, and the corresponding flow graph (bipartite network) with 1M (1K $\times$ 1K) edges in total. Constructing and using this graph for solving MCF requires high computational resources. To alleviate this problem, we adapt a *simplified graph incremental algorithm* (SIA), originally proposed for assignment-joins in spatial databases [29], which incrementally constructs the flow graph during the flow calculations based on the edge weights. Our adaptation significantly reduces the EMD computation time.

Min-cost flow algorithms, such as SIA, only aim at efficiently evaluating a single EMD computation but they do not exploit the execution plan of EMD-based similarity queries. In other words, by integrating SIA into the filter-and-refinement framework as a black-box module, the number of EMD calculations is not affected,

[1]By *EMD calculation* we refer to the entire run of an algorithmic process that computes the EMD between two histograms.
[2]A linear estimation derived from the IRMA experiment in [25].

and every calculation is still conducted at its *entirety*. In our study, we observe that it is possible to incrementally derive and tighten a *running lower bound* for the EMD during the SIA calculation. Based on this, we introduce a *progressive bounding* (PB) technique, which can terminate the SIA calculations early for histograms that are no longer promising to the similarity query. In addition, we propose a *dynamic refinement ordering* (DRO) technique, which concurrently handles and dynamically re-orders multiple EMD calculations. These two techniques greatly reduce the computations at the refinement phase of EMD-based similarity search, boosting the search performance.

PB and DRO can be seamlessly integrated with any existing (and future) filtering techniques. We show by experimentation that our techniques can compute EMD-based similarity queries one to two orders of magnitude faster, compared to the current state-of-the-art [25]. To the best of our knowledge, ours is the first study on this subject that considers datasets of million-scale on the object cardinality and thousand-scale on the histogram dimensionality.

The rest of the paper is organized as follows. Section 2 formally defines EMD, presents a min-cost flow algorithm for its computation, and discusses the standard filter-and-refinement framework used for EMD-based similarity queries. Section 3 describes SIA, an optimized implementation of the successive shortest path MCF algorithm. Section 4 presents our progressive bounding and dynamic refinement ordering techniques. Section 5 includes an extensive experimental evaluation which demonstrates the effectiveness of our techniques. Related work is presented in Section 6. Finally, Section 7 concludes the paper with a discussion about future work.

2. PRELIMINARIES

The *Earth Mover's Distance* (EMD), first introduced by the computer vision community in [23,24], is a distance function that measures the dissimilarity of two histograms (e.g., probability or feature distributions). Given two histograms $\mathbf{q} = (q_1, \ldots, q_n)$ and $\mathbf{p} = (p_1, \ldots, p_n)$, each having n bins, a *flow matrix* $\mathbf{F}$, where $f_{i,j}$ indicates flow (i.e., earth) to move from q_i to p_j, and a *cost matrix* $\mathbf{C}$, where $c_{i,j}$ models cost of moving flow from the i-th bin to the j-th bin, we can define the total cost of moving unit flow according to $\mathbf{F}$ and $\mathbf{C}$ between $\mathbf{q}$ and $\mathbf{p}$ as

$$d(\mathbf{q}, \mathbf{p}) = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{i,j} c_{i,j} \quad (1)$$

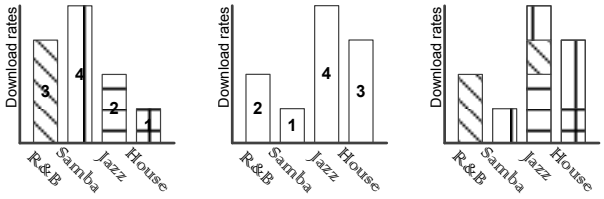
The cost matrix (a.k.a., ground distance) $\mathbf{C}$ can be designed by domain experts and/or derived from a mathematical formula [25, 32]. Intuitively, $c_{i,i} = 0$ and the larger the distance between i and j in the bin space, the larger $c_{i,j}$ is.[3] Assuming that $\mathbf{q}$ and $\mathbf{p}$ are *normalized* such that $\sum_{i=1}^{n} q_i = \sum_{i=1}^{n} p_i$, the EMD between $\mathbf{q}$ and $\mathbf{p}$ is formally defined as follows:

$$\begin{aligned} emd(\mathbf{q}, \mathbf{p}) &= \min_{\mathbf{F}} d(\mathbf{q}, \mathbf{p}), \\ \text{such that} \quad & \forall i, j \in [1, n] : f_{i,j} \geq 0, \\ & \forall i \in [1, n] : \sum_{j=1}^{n} f_{i,j} = q_i, \\ \text{and} \quad & \forall j \in [1, n] : \sum_{i=1}^{n} f_{i,j} = p_j \end{aligned} \quad (2)$$

[3]The motivating example of [32] partitions a 2-dimensional feature space (humidity and temperature) into 4×4 cells based on the range of domain values. The cost $c_{i,j}$ between bins i and j is represented by their Euclidean distance of the corresponding cells.

$emd(\mathbf{q}, \mathbf{p})$ is the minimum cost needed to transform $\mathbf{q}$ to $\mathbf{p}$; to do so, we distribute the flow (i.e., earth) from each bin q_i to a set of initially empty bins for $\mathbf{p}$, such that the resulting histogram will be equal to $\mathbf{p}$. As moving earth $f_{i,j}$ from q_i to the j-th bin of $\mathbf{p}$ bears a cost $f_{i,j}c_{i,j}$, the objective is to find the flow distribution that results in the minimum total cost. Note that $emd(\mathbf{q}, \mathbf{p})$ is equal to $emd(\mathbf{p}, \mathbf{q})$ when the cost matrix $\mathbf{C}$ is symmetric.

We demonstrate the calculation and applicability of EMD via a real example from web data analysis. Figure 1(a) and 1(b) illustrate the download rates of four music genres by two customers, $\mathbf{q}$ and $\mathbf{p}$, in an online store. The rates are normalized such that all values of each histogram sum to 10. To calculate the distance $emd(\mathbf{q}, \mathbf{p})$ between the two customers, we should identify the minimum work to transform genre distribution $\mathbf{q}$ to distribution $\mathbf{p}$. Assume that the cost matrix $\mathbf{C}$ of the music genres is as shown in Table 1, where indices 1, 2, 3, 4 denote the four music genres (i.e., *R&B*, *Samba*, *Jazz*, and *House*, respectively). Figure 1(c) illustrates the best transformation of $\mathbf{q}$ to $\mathbf{p}$ in terms of the total cost among all feasible transformations. For instance, there are 3 units in $\mathbf{q}$'s *R&B* genre. In the transformation, $f_{1,1} = 2$ units are moved to $\mathbf{p}$'s *R&B* (with cost $2 \cdot c_{1,1} = 0$) and $f_{1,3} = 1$ unit is moved to $\mathbf{p}$'s *Jazz* (with cost $1 \cdot c_{1,3} = 0.1$). Thus, based on the best transformation, $emd(\mathbf{q}, \mathbf{p})$ is $0.1 + 2.4 + 0 + 0 = 2.5$, providing a quantitative measure for the dissimilarity between these two customers. This example demonstrates an application of EMD to viral marketing analysis, which enables enterprises to derive similarities between customers in order to promote their products.

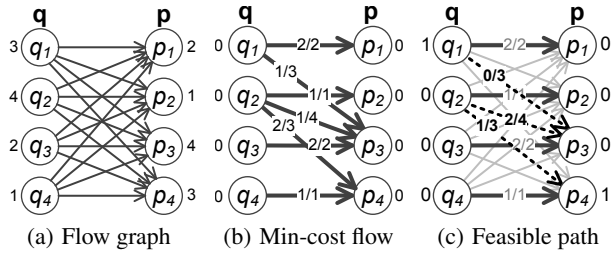


(a) $\mathbf{q}$'s distribution (b) $\mathbf{p}$'s distribution (c) $emd(\mathbf{q}, \mathbf{p})$

Figure 1: A concrete example of online music library analysis

Table 1: Cost matrix C of 4 music genres

	p_1	p_2	p_3	p_4
q_1	0	0.9	0.1	0.7
q_2	0.9	0	0.6	0.9
q_3	0.1	0.6	0	0.3
q_4	0.7	0.9	0.3	0



(a) Flow graph (b) Min-cost flow (c) Feasible path

Figure 2: The flow network of the music example

2.1 Computing the EMD

EMD can be computed using linear programming [12] and network flow algorithms [1]. We now explain how EMD computation can be modeled and solved as a network flow problem. We first construct a bipartite flow network (see Figure 2(a) for the example of Figure 1), where the vertices are derived from the histogram bins (e.g., music genres) and the edges connect the bins between the two histograms. Each edge carries a cost according to the corresponding cell of the cost matrix. The *flow capacity* of each vertex corresponds to the value of the corresponding histogram bin. For instance, the flow capacity of vertex q_1 (i.e., *R&B* of $\mathbf{q}$) in Figure 2(a) is set to 3 according to Figure 1(a). Finding the *minimum-cost flow* in this bipartite graph is equivalent to finding the EMD from $\mathbf{q}$ to $\mathbf{p}$. Each vertex of $\mathbf{q}$ should send total flow equal to its capacity and each vertex of $\mathbf{p}$ should receive total flow equal to its capacity. The minimum-cost flow is shown in Figure 2(b). On each edge $e(q_i, p_j)$, the label $f_{i,j}/cap_{i,j}$ shows the flow $f_{i,j}$ sent from the origin vertex q_i and the *capacity* $cap_{i,j}$ of that edge (i.e., the maximum flow which could possibly be sent from q_i to p_j). The edge capacity $cap_{i,j}$ is the minimum capacity of the two end-vertices; e.g., the capacity of $e(q_2, p_4)$ is 3 $(= \min\{cap_{q_2}, cap_{p_4}\} = \min\{4, 3\})$.

The *successive shortest path* (SSP) algorithm [1] is the most representative algorithm in the category of the network flow based solutions. SSP iteratively computes and *augments* paths that (i) start from a vertex q_i which has remaining flow capacity, (ii) terminate to a vertex p_i which also has remaining flow capacity, and (iii) nodes from $\mathbf{q}$ and $\mathbf{p}$ are alternated in these paths. A valid path should include *feasible* edges only. Given a flow graph, an edge is *feasible* if there is remaining flow capacity on the edge. When augmenting a flow $f_{i,j}$ on an edge $e(q_i, p_j)$, we subtract $f_{i,j}$ from the capacity of $e(q_i, p_j)$ and add $f_{i,j}$ to the flow capacity of $e(p_j, q_i)$. In our running example, initially, no flow has been augmented on any edge (i.e., the $f_{i,j}$ labels of all edges are set to 0); thus, edge $e(q_2, p_4)$ (illustrated in Figure 3(a)) is feasible since the remaining flow capacity from q_2 to p_4 is 3 (illustrated by the number on the dashed line). If we augment 1 unit of flow on $e(q_2, p_4)$, we subtract 1 from the capacity of $e(q_2, p_4)$ and add 1 to the capacity of $e(p_4, q_2)$ (as shown in Figure 3(b)). Note that $e(p_4, q_2)$ is not a physical edge in G, as there are only directed edges from $\mathbf{q}$ to $\mathbf{p}$ but not vice-versa. However, during path computation, SSP traverses also reverse edges provided that they are feasible. A formal definition of feasible edges is given below. The capacity of a non-physical edge (i.e., reverse edge) $e(p_j, q_i)$ always equals to the flow $f_{i,j}$ currently on edge $e(q_i, p_j)$.

DEFINITION 1 (FEASIBLE EDGE). *Given a flow graph, a physical edge $e(q_i, p_j)$ is feasible if $f_{i,j} < cap_{i,j}$; a non-physical (reverse) edge $e(p_j, q_i)$ is feasible if $f_{i,j} > 0$.*

(a) Before augmentation (b) After augmentation

Figure 3: Augmenting 1 flow on edge $e(q_s, p_h)$

Besides, the *cost* $c_f(u, v)$ of an edge $e(u, v)$ is determined by its physical existence in the flow graph and the cost matrix $\mathbf{C}$:

DEFINITION 2 (COST OF FEASIBLE EDGE). *The cost of a physical edge $e(q_i, p_j)$ is $c_f(q_i, p_j) = c_{i,j}$, while the cost of a non-physical (reverse) edge $e(p_j, q_i)$ is $c_f(p_j, q_i) = -c_{i,j}$.*

For instance, the reverse edge $e(p_4, q_2)$ in Figure 3(b) is feasible. Its cost is $c_f(p_4, q_2) = -c_{2,4} = -0.9$ since $e(p_4, q_2)$ is not a physical edge. To calculate the min-cost flow (i.e., EMD), SSP iteratively searches for the *feasible* path having the *minimum cost*. As discussed above, a feasible path starts from a vertex in $\mathbf{q}$, which still has positive flow capacity, ends at a vertex in $\mathbf{p}$ with positive flow capacity and includes only feasible edges. For instance, there is a feasible path highlighted by three dashed lines in Figure 2(c); the path starts at q_1, passes p_3 and q_2, and finally reaches p_4. The cost of this path is $c_f(q_1, p_3) + c_f(p_3, q_2) + c_f(q_2, p_4) = 0.1 + (-0.6) + 0.9 = 0.4$. SSP selects the feasible path with the *lowest cost* and *augments* the maximum possible flow along the path. The augmented flow is determined by the minimum of the following quantities: (i) the remaining flow capacity at the source node, (ii) the remaining flow capacity at the destination, (iii) the minimum remaining capacity of all edges on the path. For example, Figure 2(b) shows the result of augmenting the path shown by the three dashed lines in Figure 2(c). The augmentation adds 1 flow unit to all physical edges on the path (i.e., $e(q_1, p_3)$ and $e(q_2, p_4)$), subtracts 1 flow unit from the edges, for which the reverse edge is on the path (i.e., $e(q_2, p_3)$), and updates the capacities of path edges, q_1, and p_4.

Computing the EMD, using SSP requires $O(F|E|log|V|)$ time, where F is the total number of flows we need to augment and $O(|E|log|V|)$ is the cost of a shortest path search on a bipartite graph with $|V|$ vertices and $|E|$ edges. After each path augmentation, the *changes in the graph* render the subsequent shortest path search *independent* from the previous one, therefore, a large number of shortest path computations should be applied. As we shall see in Section 3, we can greatly reduce the cost of SSP by a method which builds and searches the flow graph incrementally.

2.2 Filter-and-Refinement Framework

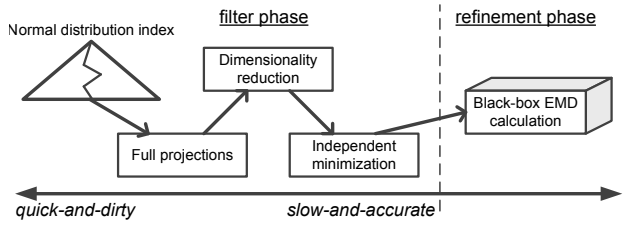


Figure 4: Filter-and-refinement framework

Given a collection $\mathcal{D}$ of histograms and a *query* histogram $\mathbf{q}$, a k-nearest neighbor (k-NN) query, finds a subset $\mathcal{S}$ of $\mathcal{D}$ containing k histograms, such that $\forall \mathbf{p} \in \mathcal{S}, \forall \mathbf{p}' \in \mathcal{D} \setminus \mathcal{S}, emd(\mathbf{q}, \mathbf{p}) \leq emd(\mathbf{q}, \mathbf{p}')$. The k-NN query is the most popular similarity search type, as the number of results is controlled by k and there is no requirement for setting a similarity threshold prior to search. In previous studies [5, 7, 25, 30], k-NN queries are evaluated based on a filter-and-refinement framework. The EMD $emd(\mathbf{q}, \mathbf{p})$ between the query histogram $\mathbf{q}$ and every histogram $\mathbf{p} \in \mathcal{D}$ is estimated with the help of lower bound filtering techniques, such as the *normal distribution index* [25], the *full projection lower bound* [7], the *reduced dimension lower bound* [30], and *independent minimization* [5]. In general, these filters are applied in an order starting from *quick-and-dirty* ones to *slow-and-accurate* ones, as shown in Figure 4. For histograms $\mathbf{p}$ that cannot be pruned by the filters, the actual $emd(\mathbf{q}, \mathbf{p})$ is calculated by a black-box computation module, such as SSP or transportation simplex [12].

Algorithm 1 FILTER-AND-REFINEMENT FOR k-NN

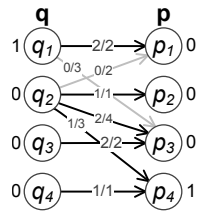
H: heap, θ: pruning threshold
Algorithm k-NN(Query $\mathbf{q}$, Index I, Filters Δ)
1: $\theta := \infty$; $H := \emptyset$
2: **while** $I.getnext(\mathbf{q}, \theta, \langle \mathbf{p}, lb_{\mathbf{p}} \rangle)$ **do**
3: **for** $\delta_i \in \Delta$ **do** ▷ Filter phase
4: $lb_{\mathbf{p}} := \max\{lb_{\mathbf{p}}, \delta_i(\mathbf{q}, \mathbf{p})\}$
5: **if** $lb_{\mathbf{p}} \geq \theta$ **then** break loop
6: **if** $lb_{\mathbf{p}} < \theta$ **then** ▷ Refinement phase
7: **if** $emd(\mathbf{q}, \mathbf{p}) < \theta$ **then**
8: update H to include $\langle \mathbf{p}, emd(\mathbf{q}, \mathbf{p}) \rangle$
9: $\theta :=$ k-th EMD value in H
10: **return** H

Algorithm 1 is a pseudocode of the filter-and-refinement framework used in previous work. Since histograms correspond to objects (e.g., images), we will use the terms objects and histograms interchangeably. First, the framework accesses objects from an index I, such as the *normal distribution index* [25] or the *TBI index* [32]. These indexes provide a $getnext$ function which returns at each call an unseen object $\mathbf{p}$ having lower bound $lb_{\mathbf{p}}$ of $emd(\mathbf{q}, \mathbf{p})$ smaller than a given threshold θ (line 2). In k-NN search, θ is the k-th largest EMD computed so far (i.e., the distance of the current k-th NN of $\mathbf{q}$ in $\mathcal{D}$). At each iteration (lines 2–9), the framework accesses a histogram $\mathbf{p} \in \mathcal{D}$ using the $getnext$ function and attempts to tighten its lower bound $lb_{\mathbf{p}}$ by applying a set Δ of progressively more expensive and accurate lower bound estimation techniques (see Figure 4). If any of the computed lower bounds is not smaller than the pruning threshold θ (line 5), then $\mathbf{p}$ is *filtered*, i.e., the exact $emd(\mathbf{q}, \mathbf{p})$ needs not to be computed. Otherwise, $emd(\mathbf{q}, \mathbf{p})$ is essentially computed by a black-box algorithm (line 7). During search, Algorithm 1 maintains a heap H with the k histograms having the lowest EMD so far and the pruning threshold θ (lines 8–9). The k-NN candidates are confirmed as the result, when no more objects are returned by the $getnext$ function (line 2), i.e., all unseen objects do not satisfy the distance threshold θ.

3. SCALING UP SSP

Computing EMD by SSP requires having the complete bipartite graph between $\mathbf{q}$ and $\mathbf{p}$, which is quadratic to the number of histogram bins. In our previous work, we proposed an optimized version of SSP, *simplified graph incremental algorithm* (SIA) [29], to scale up the computation of *spatial matching* problems. We now show how SIA can be adopted to scale up the computation of EMD, for histograms with a large number of bins. Different from SSP, SIA incrementally constructs a *partial* flow graph G' by inserting edges from the complete bipartite graph G to G'. The incremental graph construction significantly reduces the search cost since the size of the partial graph G' is typically much smaller than the complete graph G. We use the running example (cf. Figure 2) to demonstrate the superiority of SIA in Figure 5. Recall that the min-cost feasible path starting at q_1 (cf. Figure 2(c)) is $q_1 \rightarrow p_3 \rightarrow q_2 \rightarrow p_4$ in the complete graph G. Suppose a partial graph G' is constructed based on the seen values in the partial cost matrix (Figure 5(b)), the same min-cost feasible path can be found in G' as well. Thereby, augmenting this path in G' is equivalent to the augmentation in G which returns the same result in Figure 2(b).

The question now turns to *how we can guarantee that the min-cost feasible paths in G and G' are equivalent*. This is done by a distance bound checking where the bound Π is derived from the edges not yet inserted into G'. As an intuition, if the edges in G' are inserted incrementally in ascending order to their costs at every vertex, it is possible that the min-cost path in G' is cheaper than all



(a) Partial graph, G'

	p_1	p_2	p_3	p_4
q_1	0	-	0.1	-
q_2	0.9	0	0.6	0.9
q_3	-	-	0	-
q_4	-	-	-	0

(b) The cost matrix of G'

Figure 5: SIA on a partial flow graph

feasible paths which include edges not in G'. In this case, the path is augmented; otherwise SIA expands G' by inserting more edges until the best feasible path in G' has lower cost than the threshold Π[4]; the expansion of G' can only reduce the cost of the feasible flow and increase Π, as edges are inserted to G' in increasing order of their costs.

Algorithm 2 SIA BASED EMD CALCULATION

Π: distance bound, G': running subgraph
Algorithm emd_{SIA}(Histograms $\mathbf{q}$, $\mathbf{p}$)
1: $\Pi := 0$; $G' := \emptyset$
2: **while** $\exists$ feasible q_i **do**
3: $sp :=$Dijkstra(q_i, G')
4: **while** $sp.cost > \Pi$ or sp doesn't reach any feasible p_j **do**
5: insert min-cost edge $e(q_l, p_m) \in G - G'$ into G'
6: update distance bound Π
7: $sp :=$Dijkstra(q_i, G')
8: augment sp
9: **return** total augmenting cost

Algorithm 2 is a pseudocode of SIA for EMD calculations. At each iteration, SIA searches the min-cost feasible path sp using Dijkstra's shortest path algorithm [1] in G' from any feasible vertex (line 3), i.e., a vertex of $\mathbf{q}$ with remaining capacity.[5] If the cost of sp does not exceed the distance bound Π (line 4), then it must be a valid min-cost feasible path in the entire graph G. We augment the flow of sp if sp is valid (line 8). Otherwise, G' is essentially expanded by adding more edges from G (line 5) and the distance bound Π is updated accordingly (line 6).

We demonstrate the functionality of SIA by the example of Figure 6. Suppose that G' contains only 6 edges and there are 9 flow capacities sent already. According to the flow capacities, q_1 is the only feasible node in $\mathbf{q}$ but there is no feasible path currently from q_1 to any node of $\mathbf{p}$ (see Figure 6(a)). Subsequently, we insert a new edge, $e(q_1, p_3)$, into G' and now there is a shortest path, $sp = \langle e(q_1, p_3), e(p_3, q_2), e(q_2, p_4) \rangle$. The cost of sp is 0.4, which is smaller than the distance bound Π, thus we return sp as the result of the current search and augment 1 unit of flow from q_1 and q_2 to p_3 and p_4, while 1 unit of flow from q_2 to p_3 is canceled.

[4]For clarity, $\Pi = c_{max}(E - E') - \tau_{max}$, where $c_{max}(\cdot)$ returns the maximum cost in a set of edges, E' denotes the edges in G', and τ_{max} indicates the largest potential value of the vertices. As a note, Π can be further tightened if considering only a subset of E' where the edges are from the vertices being *visited* by the current Dijkstra search. The correctness proof and optimization details are given in [29].

[5]Since G (and G') may contain edges of negative costs (i.e., the reverse of edges that currently carries flow), Dijkstra's algorithm cannot be directly applied. To make its application possible, we need to iteratively maintain a *potential* value at every vertex, which transforms the costs of the feasible edges to non-negative values. The details (see [29]) are omitted for the sake of readability.