MELODY-JOIN: Efficient Earth Mover's Distance Similarity Joins Using MapReduce

Jin Huang [†1], Rui Zhang [†2], Rajkumar Buyya [†3], Jian Chen [‡4]

[†] *Department of Computing and Information Systems, University of Melbourne, Victoria, Australia*
{[1] jin.huang, [2] rui.zhang, [3] rbuyya}@unimelb.edu.au

[‡] *South China University of Technology, Guangzhou, China*
[4] ellachen@scut.edu.cn, the corresponding author

***Abstract*—The Earth Mover's Distance (EMD) similarity join retrieves pairs of records with EMD below a given threshold. It has a number of important applications such as near duplicate image retrieval and pattern analysis in probabilistic datasets. However, the computational cost of EMD is super cubic to the number of bins in the histograms used to represent the data objects. Consequently, the EMD similarity join operation is prohibitive for large datasets. This is the first paper that specifically addresses the EMD similarity join and we propose to use MapReduce to approach this problem. The MapReduce algorithms designed for generic metric distance similarity joins are inefficient for the EMD similarity join because they involve a large number of distance computations and have unbalanced workloads on reducers when dealing with skewed datasets. We propose a novel framework, named MELODY-JOIN, which transforms data into the space of EMD lower bounds and performs pruning and partitioning at a low cost because computing these EMD lower bounds has a constant complexity. Furthermore, we address two key problems, the limited pruning power and the unbalanced workloads, by enhancing each phase in the MELODY-JOIN framework. We conduct extensive experiments on real datasets. The results show that MELODY-JOIN outperforms the state-of-the-art technique by an order of magnitude, scales up better on large datasets than the state-of-the-art technique, and scales out well on distributed machines.**

I. INTRODUCTION

The similarity join retrieves all the pairs of objects from two datasets such that the similarity between the two objects in every pair is beyond a certain threshold. The similarity measure employed in the join predicate determines which data objects are similar to each other and thus has a large influence on the effectiveness of the operation. The *Earth Mover's Distance* (EMD) is an attractive measure for applications such as content-based image retrieval [16], video-based gesture recognition [13], near duplicate detection [23], and probabilistic data mining [24]. For example, given a dataset of copyright images and a dataset of user-uploaded images, the EMD similarity join will detect all potential copyright infringement among the user-uploaded contents. Another example is analyzing the similar patterns observed in probabilistic data collected by a large group of sensors. In the above applications, EMD is more popular than ℓ_p distances since it captures the visual similarity between data objects in accordance to human perception by defining the distance as the *minimal transformation cost* between two data objects. However, such minimization is a transshipment problem which has $O(n^3 \log n)$ complexity. In our experiment on a machine with 3.2GHz CPU, a single EMD computation on two histograms with 32 three-dimensional bins ($n = 32$) consumes 50 ms, which is about 25,000 times of the ℓ_2 distance's 0.002 ms on the same histograms. In real applications datasets may contain hundreds of thousands or even millions of objects. An EMD similarity join on them may *take tens of days to complete on a single machine*. The high cost of EMD similarity join has hindered its applications in practice despite its effectiveness in similarity analysis.

We propose to use the popular parallel computation paradigm, MapReduce [5], to tackle this problem. There have been enormous efforts in implementing join operations using MapReduce [7], [9], [11], [12], [18]. However, the state-of-the-art technique designed for metric space similarity joins, named *MRSimJoin* [18], is inefficient for the EMD similarity join due to the following two challenges.

- **Challenge 1**: Existing techniques such as MRSimJoin require performing pruning and partitioning in the EMD space, which is prohibitive.
- **Challenge 2**: Real-life data are usually skewed. MRSimJoin may have highly unbalanced workloads in case of skewed data, which results in long completion time.

To overcome Challenge 1, we propose a novel framework named *Mapreduce Earth mover's distance Lower bOund baseD similaritY Join* (**MELODY-JOIN**). Instead of pruning dissimilar pairs and partitioning data objects in the EMD space as MRSimJoin does, MELODY-JOIN transforms data objects into the space of EMD lower bounds, in which the pruning and partitioning can be performed based on the EMD lower bounds; these lower bounds can be computed at a constant cost in the transformed low-dimensional space, which is much cheaper than computing EMD in the original high-dimensional space. Conceptually, MELODY-JOIN has three phases.

1) Transform data objects into the space of the normal lower bound of EMD.
2) Divide the transformed space using a specially designed grid based on the characteristics of the lower bound and group the transformed records into cells.
3) Compute the lower bound of the EMD between every

record and every cell; any $\langle record, cell\rangle$ pair that has a lower bound of EMD greater than the threshold is pruned. The remaining $\langle record, cell\rangle$ pairs will be partitioned and go through further refinement steps.

To enhance the pruning power of MELODY-JOIN, we employ multiple lower bounds of EMD at the same time. Using multiple lower bounds, data objects are transformed to multiple lower bound spaces and are grouped by composite cells formed by the cells from *multiple* spaces instead of the cells in *one* space. Additionally, many types of EMD lower bounds can be easily plugged into the framework thanks to the generality of MELODY-JOIN.

To overcome Challenge 2, we propose to use a quantile based grid technique and a cardinality based grouping technique to balanced the workloads of the refinement steps in the third phase of MELODY-JOIN. The quantile grid divides the transformed space according to the distribution of transformed records, so that each cell contains a similar number of records. The cardinality based grouping technique makes use of the number of records contained in each cell, which is obtained as a byproduct of the second phase, to partition composite cells into groups such that all groups have balanced workloads.

To summarize, our contributions in this paper are as follows.

1) This is the first paper that specifically addresses the EMD similarity join and we propose to use MapReduce to approach this problem. This is also one of the rare studies that employs the MapReduce paradigm to tackle a *computation-intensive* problem.
2) We propose a novel framework named MELODY-JOIN, which prunes dissimilar pairs and partitions data in the space of EMD lower bounds to avoid any EMD computations during the pruning and partitioning process.
3) We employ multiple lower bounds of EMD at the same time to enhance the pruning power of MELODY-JOIN. We address the unbalanced workloads problem by enhancing MELODY-JOIN with the quantile based grid and the cardinality based grouping techniques.
4) We conduct extensive experiments on large real datasets and confirm the effectiveness and efficiency of MELODY-JOIN. The improvement of MELODY-JOIN over the state-of-the-art technique is typically an order of magnitude and the improvement increases when dataset size becomes larger. The results also show that MELODY-JOIN scales out well.

The remainder of the paper is organized as follows. Section II presents the preliminaries. Section III describes the MELODY-JOIN framework and Section IV elaborates the enhancement of the framework. Section V discusses related studies. Section VI shows the empirical results and Section VII concludes the paper.

II. PRELIMINARIES

We first present problem formulation and then briefly describe techniques that MELODY-JOIN is built on. Table I lists the frequently used symbols and their meanings.

TABLE I
FREQUENTLY USED SYMBOLS

Symbol	Meanings
H	A set of histograms
h	A histogram
n	The number of bins in a histogram
$\vec{l_i}$	The location vector of the $i^{th}(1 \le i \le n)$ bin in a histogram
w_i	The weight of $i^{th}(1 \le i \le n)$ bin in a histogram
L	The bin location vectors $\{\vec{l_i}\|1 \le i \le n\}$ of a histogram
$L^{\hat{v}}$	The projected bin location values $\{\vec{l_i} \cdot \hat{v}\|1 \le i \le n\}$
W_h	The weights $\{w_i\|1 \le i \le n\}$ of a histogram h
ϵ	The EMD threshold of the similarity join
p	The number of random project vectors
$\hat{v_j}$	The $j^{th}(1 \le j \le p)$ random unit projection vector
μ, σ	The mean and standard deviation of the normal $\mathcal{N}(\mu, \sigma^2)$
C_h^j	The approximation error values of the normal CDF has on the histogram CDF of j^{th} projected histogram
(m, b)	The Hough transformed normal, $m = \frac{1}{\sigma}, b = \frac{-\mu}{\sigma}$
z^2	The number of cells in the grid
$-t_{min}$, $-t_{max}$	The slopes of lines that divide the space into grid cells, $t_{min} = \min_{1 \le i \le n} \vec{l_i} \cdot \hat{v}$, $t_{max} = \max_{1 \le i \le n} \vec{l_i} \cdot \hat{v}$
g_h^j	The cell that h belongs to in the j^{th} Hough transformed space
G_h	The composite cell key $\{g_h^j\|1 \le j \le p\}$ of h in p spaces
m', b'	The projected segments of (m, b) on two reference lines
Q_m, Q_b	The quantile values $\{q_k\|0 \le k \le z\}$ of all m' or b'
Π, π	The dual feasible solution and its key
λ_Π, ρ_Π	The two hashed values of histogram, obtained through a Π
e_G	The group key for a composite cell key G

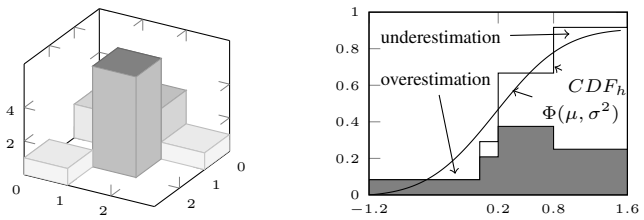
A. Earth Mover's Distance and Problem Definition

In our problem, data objects are represented as histograms and in the remainder of the paper *we refer to the histogram of a data object simply as a histogram*. A histogram h is represented as n bins. Each bin consists of a location which is a multi- dimensional vector $\vec{l_i}$ and a weight which is a nonnegative value w_i. The EMD between a pair of histograms is the *minimum cost* of transforming one histogram to the other histogram, where the cost is defined as the amount of weight moved times the *ground distance* between bins that the weight is moved [14]. We follow the literature [15], [22], [24] and focus on EMD with ℓ_2 ground distance in this paper. In many image-related applications, features (bins) are collected locally corresponding to a region in the image. Hence, the ℓ_2 distance between bin locations naturally captures the distance between the regions. Formally, we have

$$EMD(h_\alpha, h_\beta) = \min \sum_i^n \sum_j^n f_{i,j} d_{i,j}$$

$$s.t. \forall i : \sum_j f_{i,j} = w_i;\ \forall j : \sum_i f_{i,j} = w_j;\ \forall i, j : f_{i,j} \ge 0,$$

where $d_{i,j}$ is the ground distance between i^{th} and j^{th} bins, i.e., $d_{i,j} = d_{\ell_2}(\vec{l_i}, \vec{l_j})$. EMD is a special case of the Kantorovich-Rubinstein transshipment problem, which can be solved by the transportation simplex method with an average-case time



(a) a 2D histogram h with $L = \{(0,0),(0,1),(0,2),(1,1),(2,1)\}$ and $W = \{2.5, 3, 1, 4.5, 1\}$

(b) *normalized* projected histogram $L^{\hat{v}} = \{-1.2, 0, 0.2, 0.8, 0.6\}$ and $W = \{0.083, 0.208, 0.375, 0.25, 0.083\}$

Fig. 1. Projecting a two-dimensional histograms to an one-dimensional histogram with a unit vector $\hat{v} = \{0.6, -0.8\}$, and then use a normal CDF $\Phi(\mu, \sigma^2)$ to approximate the histogram CDF

complexity of $O(n^3 \log n)$.

The EMD similarity join retrieves all the pairs consisting of one histogram from each of the two histogram sets H_R and H_S such that the two histograms in the retrieved pair has an EMD not larger than a given threshold ϵ. Formally,

Definition 1. ***EMD Similarity Join*** *Given two histogram datasets H_R and H_S and an EMD threshold ϵ, the join returns $\{(h_R, h_S) | EMD(h_R, h_S) \leq \epsilon, h_R \in H_R, h_S \in H_S\}$.*

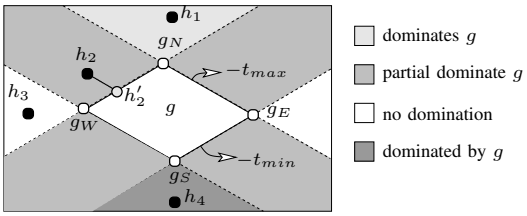
The EMD similarity join has an $O(|H_R||H_S|n^3 \log n)$ cost, where $|H_R|$ and $|H_S|$ are the cardinality of the sets H_R and H_S, respectively. This is a very expensive operation even on datasets of a few hundred thousand data objects as described in the introduction.

B. Normal Lower Bounds of EMD

Lower bounds of EMD are useful for similarity joins since any pairs with a lower bound larger than ϵ can be discarded. Many lower bounds of EMD have been studied and are discussed in Section V-A. Below we describe the normal lower bound of EMD (***normal LB***) [15], which we employ to prune dissimilar pairs and partition data in MELODY-JOIN.

The normal LB between a histogram and a group of histograms is obtained by the following steps.

1) ***Projecting*** *the original histograms to one-dimensional histograms*. Fig. 1 shows an example where a unit vector is used to project a two-dimensional histogram to an one-dimensional histogram.
2) *Constructing Cumulative Distribution Functions (**CDF**) for the one-dimensional histograms*. In the example shown in Fig. 1b, the shaded histogram is the projected one-dimensional histogram and the white histogram is its corresponding CDF.
3) *Approximating the histogram CDF with the CDF of a **normal** distribution*. This is done by computing μ and σ for the histogram CDF and compute the approximation error values, denoted by C, of the normal CDF, denoted by $\Phi(\mu, \sigma^2)$. In Fig. 1b, in different intervals the normal CDF either overestimates or underestimates the histogram CDF. Hence, here C is a set of values computed for predefined intervals in the range $[t_{min}, t_{max}]$, where $t_{min} = min(\vec{l} \cdot \hat{v})$ and $t_{max} = max(\vec{l} \cdot \hat{v})$.
4) ***Transforming*** *normal CDF to Hough normal space*. For normal CDF $\Phi(\mu, \sigma^2)$, we transform it to a record (m, b), where $m = \frac{1}{\sigma}$ and $b = \frac{-\mu}{\sigma}$. *Each transformed record in the Hough normal space therefore corresponds to one histogram in the original space*.



$$LB_{normal}(h_1, g) = N(h_1, g_N) + E(h_1, g_N)$$
$$LB_{normal}(h_2, g) = \tfrac{1}{2}(N(h_2, g_N) + N(h_2', h_2) - N(h_2', g_N)) + E(h_2, g_N)$$
$$LB_{normal}(h_3, g) = min(\tfrac{1}{2}(N(h_3, g_N) + N(g_W, h_3) - N(g_W, g_N)) + E(h_3, g_N), \tfrac{1}{2}(N(h_3, g_S) + N(g_W, h_3) - N(g_W, g_S)) + E(h_3, g_S))$$
$$LB_{normal}(h_4, g) = N(h_4, g_S) + E(h_4, g_S)$$

Fig. 2. Domination relationships in the transformed Hough space; h_2' is the projection of h_2 along a line with slope of $-t_{max}$ to the edge of g

5) ***Grouping*** *transformed records into diamond-shape region and computing the normal LB*. The transformed records are grouped to a diamond-shape region using lines with slopes of $-t_{min}$ and $-t_{max}$. For a record h and a region g, let h' denote the the projection of h on its nearest edge of g, t denote the intersection between Φ_a and Φ_b,we have

$$EMD(h, h_g \in g) \geq LB_{normal}(h, g)$$
$$= \begin{cases} N(h, g_N) + E(h, g_N) & \text{if } h \text{ dominates } g, \\ F(h, h', \arg\min_{g=\{g_N, g_S\}} d_{\ell_2}(g, h)) & \text{if } h \text{ partially dominates } g, \\ \min(F(h, g_R, g_N), F(h, g_R, g_S)) & \text{if } h \text{ has no domination relationship with } g, \\ N(h, g_S) + E(h, g_S) & \text{if } h \text{ is dominated by } g, \end{cases}$$

where,

$$F(a, b, c) = \frac{1}{2}(N(a, c) + N(b, a) - N(b, c)) + E(a, c),$$

$$N(a, b) = \begin{cases} |\int_{t_{min}}^{t} \Phi_a - \int_{t_{min}}^{t} \Phi_b| + |\int_{t}^{t_{max}} \Phi_a - \int_{t}^{t_{max}} \Phi_b| & \text{if } t_{min} \leq t \leq t_{max} \\ |\int_{t_{min}}^{t_{max}} \Phi_a - \int_{t_{min}}^{t_{max}} \Phi_b| & \text{if } t_{min} > t \text{ or } t > t_{max}, \end{cases}$$

$$E(a, b) = \begin{cases} C_a[t] - C_b[t] & \text{if } \Phi_b \text{ dominates } \Phi_a, \\ C_b[t] - C_a[t] & \text{if } \Phi_a \text{ dominates } \Phi_b, \end{cases}$$

$$g_R = \arg\min_{g=\{g_W, g_E\}} d_{\ell_2}(g, h),$$

where g_N, g_W, g_S, and g_E represent the northern, western, southern, and eastern vertex of g, respectively. The dominance relationship between h and g is determined by the stochastic dominance between their corresponding normal CDF. There are four possible relationships. Fig. 2 shows an example, where h_1 dominates g, h_2 partially dominates g, h_3 has no dominance relationship with g, and h_4 is dominated by g.

This normal LB can be computed at an $O(1)$ cost since 1) two normal CDF with different σ only have one intersection; 2) there is a closed-form formula on integrating normal CDF, i.e., $\int_{x_1}^{x_2} \Phi(\mu, \sigma^2) = \sigma(x_2 \Phi_{std}(x_2) + \phi_{std}(x_2) - x_1 \Phi_{std}(x_1) -$

$\phi_{std}(x_1))$, where Φ_{std} and ϕ_{std} are the CDF and density function of the standard normal distribution $\mathcal{N}(0,1)$, respectively.

Besides using the normal LB to transform data objects into the EMD lower bound space, we also use another type of EMD lower bound, the dual lower bound (***dual LB***) to show how multiple lower bounds can be plugged into MELODY-JOIN. Dual LB [24] is computed via *feasible solutions* to the dual form problem of the optimization in EMD. A feasible solution, denoted by Π, is a set of variables satisfying the constraints of the dual form problem, with a *solution key*, denoted by π, computed by aggregating these variables. The feasible solution can be computed by sampling the histograms. Each feasible solution can transform a histogram to two one-dimensional values λ and ρ, denoted as *dual keys*. Given two histograms h and h', two lower bounds can be computed as $(\lambda_h + \rho_{h'})$ and $(\lambda_h - \lambda_{h'} + \pi)$, respectively.

C. The MapReduce Paradigm

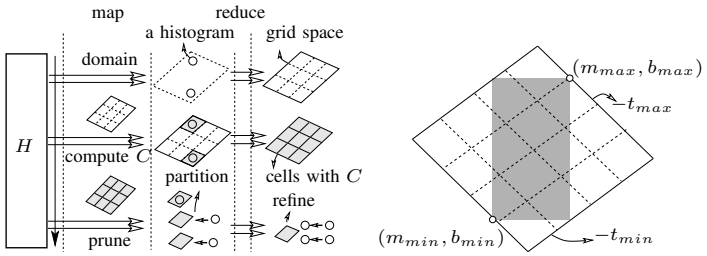
The MapReduce paradigm [5] is designed to simplify parallel programming on a cluster of commodity machines. It involves a map and a $reduce$ function. The map and $reduce$ functions are executed independently on the machines referred to as the mappers and the reducers, respectively. Briefly, the mappers first parse and process the input key-value pairs $\langle K_1|V_1\rangle$ that are read from a distributed file system, then distribute intermediate pairs $\langle K_2|V_2\rangle$ to reducers. MapReduce guarantees that intermediate pairs $\langle K_2|V_2\rangle$ with the same key value K_2 are processed by the same reducer. The reducers compute aggregate values based on $\langle K_2|V_2\rangle$ they receive and write these aggregate values back to the distributed file system. A map-$reduce$ procedure corresponds to a *MapReduce job*.

III. THE MELODY-JOIN FRAMEWORK

MELODY-JOIN employs the normal LB to transform data objects into the Hough normal space, divides the space into a grid, collects statistics on grouped records in the cells, and then conducts pruning, partitioning, and refining. We avoid pruning and partitioning in the EMD space because the original histograms are likely to have a large number of bins; the super cubic cost of computing EMD on these high-dimensional histograms will outweigh the benefits gained from pruning. We choose normal LB to transform data objects because it provides 1) an efficient way to compute the lower bound between a histogram and a group of histograms, and 2) good geometric properties for partitioning the dataset.

MELODY-JOIN involves three MapReduce jobs corresponding to the three phases described in the introduction. The first two jobs are *lightweight since they only perform linear cost operations* on the datasets. It is a common practice to employing multiple lightweight jobs to achieve high overall efficiency. Fig 3a illustrates the three jobs in MELODY-JOIN. Specifically, the three jobs are as follows.

1) ***Job 1****: Obtaining the domain of transformed space*. The mappers of this job transform the histograms into a Hough normal space where each histogram is represented by a two-dimensional record (m, b). A single reducer in this job obtains the space domain by aggregating the maximum and minimum values of m and b. The space then is conceptually divided into grid cells.
2) ***Job 2****: Computing the approximation errors for cells*. The mappers compute the approximation error values of the normal CDF and the reducers aggregate the maximum and minimum approximation errors values for each cell. These aggregated errors are used for computing the normal LB between a record and a cell.
3) ***Job 3****: Pruning, partitioning, and refining records*. The mappers prune every record against every cell and distribute it to reducers accordingly. The pruning is done by comparing ϵ with the normal LB and discarding the $\langle record, cell\rangle$ pairs with normal LB greater than ϵ. The distribution ensures that the remaining $\langle record, cell\rangle$ pairs with the same cell are refined on the same reducers, i.e., using the cell id as the key of the intermediate pairs. The reducers in this job further refine received pairs using a chain of lower bounds, and compute EMD if necessary to obtain the results.



(a) an overview on three jobs in the proposed framework

(b) dividing space into cells, shaded area is the domain

Fig. 3. The overview of the three jobs in MELODY-JOIN and dividing the transformed space into grid cells so that records are grouped accordingly

MELODY-JOIN directly supports two-way joins. In this case we build two separate grids for the objects in the two datasets and conduct pruning and partitioning on one dataset using the grid built for the other dataset. Below, we describe the details of MELODY-JOIN using a self join on a set of histogram H as an example.

A. Job 1: Obtaining the Domain of the Transformed Space

This job is designed for obtaining the domain of the transformed Hough normal space. On the mappers, the normal LB technique is applied to histograms to transform each of them to a two-dimensional record. One reducer aggregates the maximum and minimum values of each dimension, serving as the domain. The job is lightweight since the transformation is of linear cost and there is only a small portion of data distributed from mappers to reducers.

Since the location of bins are shared by all the histograms, we leverage the *DistributedCache* capability provided by Hadoop (the popular open-source implementation of MapReduce) to store the the projected bin locations $L^{\hat{v}} = \{\vec{l_i} \cdot \hat{v} | 1 \leq i \leq n\}$. *DistributedCache* makes its stored content (in this case the projected bin locations) available to all mappers and reducers in the later jobs.

Algorithm 1 lists the detailed steps in this job. On mappers, the transformation from multi-dimensional histograms to two-

Algorithm 1: Job 1 Obtaining Domain

```
   DistributedCache: L^v̂
1  map
2  foreach h ∈ H do
3      CDF_h ← W_h, L^v̂          // construct histogram CDF
4      Φ(μ, σ²) ← CDF_h          // approximate with normal CDF
5      m ← 1/σ , b ← −μ/σ        // transform in Hough space
6      output ⟨0|(m, b)⟩
7  reduce
8  foreach (m, b) do
9      update m_min, m_max, b_min, b_max   // aggregate domain
10 output ⟨0|[m_min, m_max], [b_min, b_max]⟩
```

Algorithm 2: Job 2 Computing Approximation Errors for Cells

```
   DistributedCache: L^v̂, [m_min, m_max], [b_min, b_max]
1  map
2  foreach h ∈ H do
3      CDF_h ← W_h, L^v̂
4      Φ(μ, σ²) ← CDF_h
5      C_h ← (CDF_h − ∫ Φ(μ, σ²))   // compute error values
6      g_h ← (m, b), [m_min, m_max], [b_min, b_max]   // find cell
7      output ⟨g_h|C_h⟩
8  reduce
9  foreach g do
10     foreach ⟨g|C⟩ do
11         update C_g   // aggregate min/max error values
12     output ⟨g|C_g⟩
```

Algorithm 3: Job 3 Pruning, Partitioning, and Refining

```
   DistributedCache: L, L^v̂, [m_min, m_max], [b_min, b_max], {C_g|∀g}
1  map
2  foreach h ∈ H do
3      CDF_h ← W_h, L^v̂
4      Φ(μ, σ²) ← CDF_h
5      C ← (∫ Φ(μ, σ²) − CDF_h)
6      foreach g do
7          if h ∈ g then                /* prune against cell */
8              output ⟨g|h, W_h, in⟩        /* native record */
9          else if LB_normal(h, g) ≤ ε then
10             output ⟨g|h, W_h, out⟩       /* guest record */
11 reduce
12 foreach g do
13     foreach ⟨g|h_α, W_hα, in⟩ do
14         foreach ⟨g|h_β, W_hβ, in⟩ do          /* self join */
15             if h_α ≠ h_β and EMDJoin(h_α, h_β) then
16                 output ⟨h_α h_β⟩
17         foreach ⟨g|h_β, W_hβ, out⟩ do        /* nested-loop */
18             if EMDJoin(h_α, h_β) then
19                 output ⟨h_α|h_β⟩
```

dimensional records is carried out in three steps. First, a histogram CDF of the projected one-dimensional histogram is constructed (line 3). Second, the normal CDF $\Phi(\mu, \sigma^2)$ is used to approximate this histogram CDF (line 4). Third, the record (m, b) is computed as $m = \frac{1}{\sigma}$ and $b = \frac{-\mu}{\sigma}$ (line 5). All these steps can be done at a linear cost. Optionally, a *combine* function, which aggregates the local maximum and minimum values on mappers, can be employed so that less records are distributed from the mappers to the reducers.

After the domain is obtained, the space can be divided into z^2 grid cells. As described in Section II-B, lines with slopes of $-t_{max}$ and $-t_{min}$ are used to conduct the division. We denote the two lower boundary lines of the grid as the *reference lines*. In order to cover the space domain, the grid is a *minimum bounding quadrilateral* for the domain with edges of specific slopes. The straightforward division is to *evenly* divide each reference line into z segments, which gives z^2 cells with the identical size. Fig. 3b shows an example where the shaded area is the domain and the space is divided into 4^2 cells.

B. Job 2: Computing the Approximation Errors for Cells

To prune a record from a cell, we need to compute the normal LB between them, which requires the aggregated approximation error values of the cell. To obtain these values for all cells, we employ another MapReduce job. The reason for adding another job is that Job 1 requires *one* reducer for the aggregation. If error values are to be aggregated in Job 1, all of them are required to be distributed to this reducer, suggesting limited scalability. Moreover, since the mappers of Job 1 do not have the grid information, it is impossible to leverage the $combine$ function to locally pre-aggregate error values on mappers. Aggregating them in a separate job means that a $combine$ function and multiple (up to z^2) reducers can be employed to achieve better scalability.

Algorithm 2 presents the detailed steps in this job. Specifically, for each histogram, the mappers in Job 2 compute the approximation error values C by comparing the histogram CDF with the normal CDF used for approximation (line 3 to line 5). As discussed in Section II-B, C is a set of error values computed for a set of predefined intervals. Following the conclusion in [15], we compute error values for five intervals $[t_{min}, \frac{s}{5}(t_{max} - t_{min})]$, where s is $1, 2, 3, 4$, and 5. Given z, if an even grid division is employed, the domain provides adequate information for the mappers to determine the containing cell for a record (line 6). A record is distributed to reducers using the id of its containing cell as the key. The reducers then aggregate the maximum and minimum values of (each value in) C for each cell (line 8 to line 11). The output maximum and minimum approximation error values for each cell are stored in $DistributedCache$ for further use.

C. Job 3: Pruning, Partitioning, and Refining Records

The mappers in this job prune records from cells and partition them based on the pruning results. The reducers further prune candidate pairs using a chain of lower bounds and compute EMD if necessary to refine the results.

The detailed steps are shown in Algorithm 3. On mappers, a histogram h is first transformed to (m_h, b_h) with its approximation error values C_h (line 3 to line 5). Then for each cell g and its aggregated error values C_g, $LB_{normal}(h, g)$ is computed (line 9) using the formula described in Section II-B (cell vertexes can be computed based on the domain and z). If this $LB_{normal}(h, g)$ is smaller than or equal to ϵ, h should be refined with records in g (line 10). Otherwise the record is pruned from g. Additionally, a record should be refined with records that lie in the same cell (line 7 to line 8). Hence,

Algorithm 4: EMDJoin on Reducers of Job 3

Input: two histograms h_α and h_β
Returns: true or false
1 **if** $LB_{proj}(h_\alpha, h_\beta) \leq \epsilon$ **then**
2 **if** $LB_{dual}(h_\alpha, h_\beta) \leq \epsilon$ **then**
3 **if** $LB_{redu}(h_\alpha, h_\beta) \leq \epsilon$ **then**
4 **if** $LB_{indm}(h_\alpha, h_\beta) \leq \epsilon$ **then**
5 **if** $EMD(h_\alpha, h_\beta) \leq \epsilon$ **then**
6 **return** true
7 **return** false

for each cell, there are two types of records: the *native* ones who lie in the cell and the *guest* ones who are not pruned from the cell using the lower bound. We use the flags in or out to denote whether a record is native for the cell. The straightforward way to distribute this refinement workload is to create one reduce task for one cell, i.e., using the cell id g as the key value of the intermediate pair. On reducers, a self join is performed on the native records (line 14 to line 16), while a nested-loop join is performed between the native records and the guest records (line 17 to line 19). As shown in Algorithm 4, a chain of EMD lower bounds including the projection lower bound LB_{proj}, the dual lower bound LB_{dual}, the dimensionality reduction lower bound LB_{redu}, and the independent minimization lower bound LB_{indm}, are computed to further filter dissimilar candidates. Details of these lower bounds are discussed in Section V-A.

IV. Enhancing Pruning Power and Addressing Load Balancing

Two key performance issues in MELODY-JOIN are the limited pruning power of *a single* normal LB and the unbalanced workloads on reducers in Job 3. We propose to use the following techniques to address these two issues.

- Multiple lower bounds are employed at the same time. ***Multiple projection vectors*** are used to produce multiple Hough normal spaces, so that a histogram is pruned using multiple normal LB. Moreover, we use dual LB as an example to show that ***multiple types of lower bounds*** can be easily plugged into MELODY-JOIN.
- The quantile values in each dimension of the Hough normal space are used to divide the space into a quantile based grid (which we call ***quantile grid***), such that the grid cells contain more balanced numbers of records.
- When multiple projections are used, the *composite* quantile grids may not be adequate for balancing the workloads. To produce balanced workloads for reduce tasks in Job 3, we propose to use a ***cardinality based grouping*** technique to partition composite cells *in groups* for reduce tasks to balance their workloads.

We elaborate the details of the above techniques as follows.

A. Multiple Projections and Multiple Lower Bounds

There are two ways to integrate multiple lower bounds into pruning: to use multiple projection vectors so multiple normal LB are available, and to include different types of

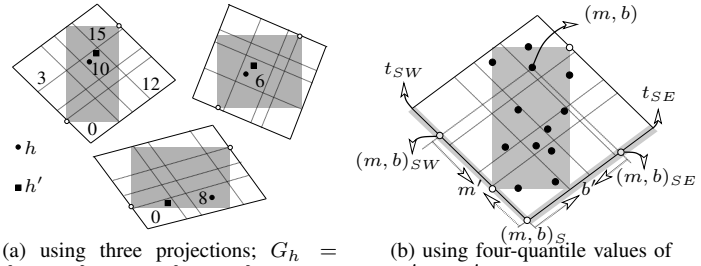
Algorithm 5: Job 1 with Quantile and Multiple Bounds

DistributedCache: $\{L^{\hat{v_j}} | 1 \leq j \leq p\}$
1 **map**
2 **foreach** $h \in H$ **do**
3 **for** $j \leftarrow 1$ **to** p **do** /* multiple projections */
4 same to **map** in Algorithm 1 with $\hat{v}$ replaced by $\hat{v_j}$
5 **output** $\langle j|(m,b)\rangle$
6 **reduce**
7 **for** $j \leftarrow 1$ **to** p **do**
8 **foreach** $\langle j|(m,b)\rangle$ **do** /* each space */
9 update $m_{min}, m_{max}, b_{min}, b_{max}$
10 $(m,b)_S \leftarrow$ project (m_{min}, b_{min}) along $-t_{max}$ to t_{SE}
11 **foreach** $\langle j|(m,b)\rangle$ **do**
12 $(m,b)_{SW} \leftarrow$ project (m,b) along $-t_{min}$ to t_{SW}
13 $m' \leftarrow d_{\ell_2}((m,b)_{SW}, (m,b)_S)$
 $(m,b)_{SE} \leftarrow$ project (m,b) along $-t_{max}$ to t_{SE}
14 $b' \leftarrow d_{\ell_2}((m,b)_{SE}, (m,b)_S)$
15 $Q_m \leftarrow \{m'\}$, $Q_b \leftarrow \{b'\}$ /* quantile values */
16 **output** $\langle j|Q_m, Q_b\rangle$
17 **cleanup**
18 $\Pi \leftarrow h_\alpha, h_\beta$ /* compute a feasible solution */

lower bounds in addition to normal LB. We consider both and modify the three jobs in MELODY-JOIN to Algorithm 5, Algorithm 6, and Algorithm 7, respectively.

To employ multiple normal LB, we use p vectors $\{\hat{v_j}|1 \leq j \leq p\}$ to obtain multiple projected one-dimensional histograms for each original histogram. This brings p independent Hough normal spaces and p grids. To group histograms, rather than grouping their corresponding transformed records by cells in a single space, we group them by the *composite cell*, which is composed of the containing cells of their transformed records in each space, i.e., $G = \{g^j|1 \leq j \leq p\}$. A histogram is a *native histogram* of a composite cell if its containing cells in all space exactly match the composite cell. Otherwise, it is a *guest histogram* of the composite cell. Fig. 4a shows an example where three projection vectors are used. A histogram h has its transformed records laying in the cell 10, the cell 6, the cell 8 in the 1^{st}, the 2^{nd}, and the 3^{rd} space, respectively. Hence, h is a native histogram of the composite cell $\{10, 6, 8\}$. Similarly, h' is a native histogram of the composite cell $\{10, 6, 0\}$ and a guest histogram of the composite cell $\{10, 6, 8\}$. To implement the idea of grouping by composite cells, Job 1 transforms the histograms to p spaces (line 3 to line 5 in Algorithm 5). In job 2, the approximation error values are aggregated for each composite cell rather than cells in one space (line 14 to line 15 in Algorithm 6). In Job 3, when trying to prune a histogram from a group of histograms in a composite cell, p normal LB are computed and compared with ϵ (line 9 to line 13 in Algorithm 7). A histogram is pruned from the group of histograms in the composite cell if *any normal LB between them is larger than* ϵ.

Other types of lower bounds can also be effortlessly plugged into MELODY-JOIN. The plugging requirements on the lower bound are: 1) it can be computed in a transformed *low-dimensional* space; and 2) it can be computed in an aggregate fashion. The requirement 1) is to ensure that Job 1 and Job 2 are at low costs. The requirement 2) is to enable chaining



(a) using three projections; $G_h = \{10, 6, 8\}, G_{h'} = \{10, 6, 0\}$

(b) using four-quantile values of m' and b' to divide space

Fig. 4. Multiple projection vectors and the quantile grids in MELODY-JOIN; shaded areas are the domains corresponding to the Hough normal spaces

Algorithm 6: Job 2 with Multiple Lower Bounds

DistributedCache: $\{L^{\hat{v}_j}(1 \le j \le p)\}, \{Q_m^j, Q_b^j (1 \le j \le p)\}, \{\Pi\}$
1 **map**
2 **foreach** $h \in H$ **do**
3 **for** $j \leftarrow 1$ **to** p **do**
4 same to **map** in Algorithm 2 with $\hat{v}$ replaced by $\hat{v_j}$
5 $g^j \leftarrow (m, b), Q_m^j, Q_b^j$
6 $G \leftarrow \{g^j | 1 \le j \le p\}$ /* find composite cell */
7 **foreach** Π **do**
8 $\lambda \leftarrow h, \Pi$ /* hash dual key */
9 **output** $\langle G | \{C^j | 1 \le j \le p\}, \{\lambda\}\rangle$
10 **reduce**
11 **foreach** G **do**
12 $O \leftarrow 0$ /* record number count */
13 **foreach** $\langle G | \{C^j | 1 \le j \le p\}, \{\lambda\}\rangle$ **do**
14 **for** $j \leftarrow 1$ **to** p **do** /* multiple spaces */
15 update C_g^j
16 **foreach** Π **do** /* multiple duals */
17 **foreach** λ **do**
18 update $\lambda_{min}, \lambda_{max}$
19 $O \leftarrow O + 1$
20 **output** $\langle G | \{C_g^j | 1 \le j \le p\}, \{[\lambda_{min}, \lambda_{max}]\}, O\rangle$

Algorithm 7: Join 3 with Multiple Lower Bounds

DistributedCache: $L, \{L^{\hat{v}_j}\}, \{Q_m^j, Q_b^j\}, \{\Pi\}$, $\langle G | \{C_g^j\}, \{[\lambda_{min}, \lambda_{max}], O\rangle, (1 \le j \le p)$
1 **map**
2 **foreach** $h \in H$ **do**
3 **for** $j \leftarrow 1$ **to** p **do**
4 same to **map** in Algorithm 3 with $\hat{v}$ replaced by $\hat{v_j}$
5 **foreach** Π **do**
6 $\lambda_\Pi, \rho_\Pi \leftarrow h, \Pi$ /* hash two dual keys */
7 **foreach** G **do**
8 $flag \leftarrow true$ /* pruned flag */
9 **for** $j \leftarrow 1$ **to** p **do**
10 **if** $h \in g^j$ **then**
11 $G_h \leftarrow g^j$; // native in composite cell
12 **else if** $LB_{normal}(h, g^j) > \epsilon$ **then**
13 $flag \leftarrow false$ /* pruned by normal */
14 **if** $G = G_h$ **then**
15 **output** $\langle G | h, W_h, in\rangle$ /* native record */
16 **else if** $flag$ **then**
17 **foreach** Π **do**
18 **if** $\lambda_{min} > (\epsilon - \rho_\Pi)$ *or* $\lambda_{max} < (\pi + \lambda_\Pi - \epsilon)$ **then**
19 $flag \leftarrow false$ /* pruned by dual */
20 **if** $flag$ **then**
21 **output** $\langle G | h, W_h, out\rangle$ /* guest record */
22 **reduce** same to **reduce** in Algorithm 3 with g being replaced by G

the lower bound when pruning histograms from a group of histograms in a composite cell. The dual LB is a good example of such lower bounds. To employ it we first need to compute several feasible solutions to the dual problem and then transform all histograms to dual keys according to these feasible solutions. This can be done in MELODY-JOIN as follows. In Job 1, each mapper samples histograms and employs the $cleanup$ function to compute a feasible solution, denoted as Π, and writes it out to $DistributedCache$ (line 17 to line 18 in Algorithm 5). The $cleanup$ function is provided by Hadoop and is executed after the map function is completed for all input. Then in Job 2, these dual feasible solutions are read by each mapper, which transforms input histograms to a set of one-dimensional dual keys, denoted as $\{\lambda\}$ (line 7 to line 8 in Algorithm 6). These values are then distributed to reducers together with the approximation error values and aggregated to generate a set of *dual ranges* $\{[\lambda_{min}, \lambda_{max}]\}$ for each composite cell (line 16 to line 18 in Algorithm 6). In Job 3, in addition to the normal EMD computation, each histogram is transformed to a set of two dual keys $\{\lambda, \rho\}$ using the feasible solutions (line 5 to line 6 in Algorithm 7). These values are used to filtered against the set of one-dimensional range $\{[\lambda_{min}, \lambda_{max}]\}$ of each composite cell using dual LB (line 17 to line 19 in Algorithm 7). Specifically, given a feasible solution Π with the solution key π, the corresponding dual keys λ, ρ of two histograms h and h', we have

$$\begin{aligned} EMD(h, h') \le \epsilon &\rightarrow LB_{dual}(h, h') \le \epsilon \\ &\rightarrow \lambda_{h'} + \rho_h \le \epsilon \text{ and } \lambda_h - \lambda_{h'} + \pi \le \epsilon \\ &\rightarrow \pi + \lambda_h - \epsilon \le \lambda_{h'} \le \epsilon - \rho_h. \end{aligned}$$

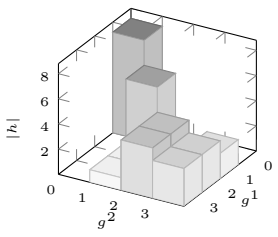
That is, a histogram is pruned from G with $[\lambda_{min}, \lambda_{max}]$ if

$$\lambda_{min} > (\epsilon - \rho) \text{ or } \lambda_{max} < (\pi + \lambda - \epsilon).$$

A histogram will be refined with a composite cell on reducers only if neither normal LB nor dual LB is able to prune it (line 21 in Algorithm 7). More types of lower bounds can be plugged into MELODY-JOIN by leveraging Job 1 and Job 2 to aggregate transformed values and chaining the pruning into the map function of Job 3.

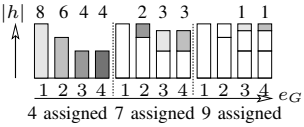
B. The Quantile Grid

The q-quantile values are $(q - 1)$ values that divide an ordered dataset into q intervals with the same number of records in each interval. A *quantile grid* is a grid that has each of its dimension divided by the quantile values of record values in that dimension. We propose to use a quantile grid instead of an even grid to divide the Hough normal space so that each cell has a similar number of records. Since we need to obtain z^2 cells by dividing each reference line into z segments, the quantile values should be collected on the *projected segments* of the record (m, b) on the two reference lines. Specifically, let t_{SW} and t_{SE} denote the reference line with slope of $-t_{max}$ and $-t_{min}$, respectively. To project



g	$\|h\|$	$\{g^1, g^2\}$	$\|h\|$
$g^1 = 0$	8	$\{0, 0\}$	8
$g^1 = 1$	8	$\{1, 1\}$	6
$g^1 = 2$	8	$\{2, 2\}$	4
$g^1 = 3$	8	$\{3, 2\}$	4
$g^2 = 0$	8	$\{2, 3\}$	3
$g^2 = 1$	8	$\{3, 3\}$	3
$g^2 = 2$	8	$\{1, 3\}$	2
$g^2 = 3$	8	$\{2, 1\}$	1
		$\{3, 1\}$	1
		Others	0

Fig. 5. The composite cells have a skewed distribution on the number of histograms even when cells in each space have the same number of histograms



e_G	G	$\|h\|$
1	{0, 0}	8
2	{1, 1}, {1, 3}	8
3	{2, 2}, {2, 3}, {2, 1}	8
4	{3, 2}, {3, 3}, {3, 1}	8

Fig. 6. Grouping composite cells into groups to balance workloads

(m, b) on these lines is to transform (m, b) along a line with the slope $-t_{max}$ to the reference line t_{SE} or along a line with the slope $-t_{min}$ to the reference line t_{SW}. Specifically, let $(m, b)_{SW}$ and $(m, b)_{SE}$ denote the projections of (m, b) on reference line t_{SW} and t_{SE}, respectively; $(m, b)_S$ denote the projection of (m_{min}, b_{min}) on reference line t_{SE}. Then for (m, b), the projected segment on t_{SW} is computed by $m' = d_{\ell_2}((m, b)_{SW}, (m, b)_S)$; the projected segment on t_{SE} is computed by $b' = d_{\ell_2}((m, b)_{SE}, (m, b)_S)$. Fig. 4b demonstrates an example on using four-quantile values to divide the Hough transform space into 4^2 cells. The cells in the shaded domain area have either one or two records, forming a grid division that has similar number of records in each cell.

Algorithm 5 shows the detailed steps of collecting quantile values in Job 1. Specifically, after obtaining the domain space on reducers (line 8 to line 9), for each (m, b) we compute m' and b' (line 11 to line 14) and aggregate them to obtain the quantile values (line 15). To divide a space into z^2 cells, we collect $(z - 1)$-quantile values for both m' and b'.

C. The Cardinality Based Grouping

When multiple projection vectors are used, the quantile grids still suffer from skewed datasets. Fig. 5 shows an example. In each space, the two quantile grids generate four balanced cells, each of which contain eight histograms. However, the resultant composite cells contain different number of histograms. If one reduce task is created for one composite cell, this skewed distribution will lead to vastly unbalanced workloads of reduce tasks and ultimately low efficiency.

To tackle this problem, we propose to use a *cardinality based estimation* on the workloads to *group* multiple composite cells together for reduce tasks, such that one reduce task is created for one group.

The grouping goal is to balance the refinement workloads in each group. The refinement workload of a composite cell consists of the self join on native histograms and the join between the native and the guest histograms. However, when mappers conduct partitioning, *there is no global information on the number of guest histograms for each composite cell* because the pruning is performed in a distributed manner on mappers.

Algorithm 8: Job 3 with Load Balancing

```
1 map-setup
2 sort {⟨G|O⟩}    /* sort G descendingly based on O */
  foreach ⟨G|O⟩ do    /* assign one by one in order */
3     e ← arg min_e Σ_{G→e} O    // group with least records
4     e ← G                      // assign G to the group
5 map same to map in Algorithm 7 with output ⟨e_G|G, h, W_h, in⟩
6 reduce
7 foreach e_G do
8     foreach G do /* records in same composite cell */
9         same to reduce in Algorithm 3 with g replaced by G
```

We therefore use the cardinality of the native histograms as an estimation of the workload of one composite cell. The problem then becomes to group composite cells to r groups such that each group has similar number of native histograms. This can be done by a simple algorithm. First we sort the composite cell in a descending order on the number of native histograms in it. Then we assign composite cells one by one to groups according to the sorted order. For each composite cell, we assign it to the group who has the smallest number of native histograms so far. Note that this simple solution may not achieve the optimal results, i.e., the minimum standard deviation on the workloads of all reduce tasks, it is adequate for our problem since it computes roughly balanced groups with negligible computational overhead. Fig. 6 demonstrates how the skewed composite cells in the previous example are assigned to four groups to form balanced workloads.

Integrating this technique into MELODY-JOIN is easy since the *the cardinality of native histograms can be obtained as a byproduct in Job 2*. The details are shown in Algorithm 6 and Algorithm 8. The cardinality of native histograms in a composite cell is collected by the reducers of the Job 2 (line 19 to line 20 in Algorithm 6). These numbers are then stored in $DistributedCache$. Later, the mappers of Job 3 makes use of these numbers and employ the aforementioned sorting based algorithm to construct a hash table from the composite cell to the group id (line 1 to line 4 in Algorithm 8). For the native histograms and not pruned guest histograms of a composite cell G, the mappers output a pair with key e_G to reducers (line 5). The composite cell G now is stored as a value in the key-value pairs (line 5 in Algorithm 8). This is used by reducers to distinguish records associated to different composite cells. Optionally, this grouping idea can be implemented in the $partition$ function provided by Hadoop.

One may consider exploiting Hadoop's scheduler to achieve the balanced workloads on reducers. This is done by setting the number of reduce tasks to a large number. The reducers which process tasks with less workload finish earlier and become idle. Hadoop then assigns pending tasks to these idle reducers so that all computation power in the cluster can be utilized. Even though, the load balancing technique here is still useful since it is *a prerequisite for the scheduler to perform well*. One example scenario is that if reduce tasks are of vastly different amounts of workloads, the one with the largest workload will still dominate the running time no matter how other tasks are

assigned in a balanced manner. Moreover, in similarity joins, increasing the number of reduce tasks directly increases the replication of records, which decreases efficiency. Overall, our proposed load balancing techniques provide essential support on handling skewed data in MELODY-JOIN.

V. RELATED WORK

In this section, we discuss the work on lower bounds of EMD and the studies on processing joins using MapReduce.

A. Lower Bounds of EMD

Lower bounds of EMD help avoid expensive EMD computations and thus are useful in the similarity search [19] in addition to the similarity join. Devising computationally cheap lower bounds for EMD has received much efforts from different fields in the community [1], [2], [4], [15], [17], [22], [24]. Generally, the techniques can be categorized into *simplification based methods* and *transformation based methods*. While all lower bounds can be chained to assist the filtering before EMD computations, the transformation based lower bounds can be easily plugged into MELODY-JOIN to enhance the pruning.

Simplification Methods. Simplification based methods generally use a cheaper EMD computation to approximate the original EMD. Several approaches are available to achieve the goal. First, EMD is of $O(n)$ cost when the location vectors are one-dimensional. The random projection [4] follows this approach. Second, the number of bins n in a histogram can be reduced, which significantly decreases the cost due to the super cubic complexity. Dimensionality reduction [22] is a technique that multiples the location vectors of bins by a reduction matrix and adjusts the ground distance accordingly. The intuition behind adjusting the ground distance is that the maximum transformation cost on the reduced bins should be preserved by the adjusted ground distance. Third, the optimization constraints can be relaxed to speedup the computation. The independent minimization technique [2] can be applied where the optimization constraints in EMD definition can be simplified by only requiring $\forall j : \sum_i f_{i,j} \leq w_j$ rather than $\forall j : \sum_i f_{i,j} = w_j$. The simplified optimization problem can be computed at an $O(n^2)$ cost.

Transformation Methods. Transformation based methods transform high-dimensional histograms to lower dimension and perform a computationally cheap operation (rather than EMD) on these transformed values to provide the lower bound of the EMD. Both the normal LB and dual LB belong to this family. Other similar lower bounds include sketching lower bound [1], which maps data into a non-metric space to achieve a sub-linear computation and small distortion.

EMD Approximation algorithms, e.g., [17], are related to the bounds of EMD. These algorithms approximate EMD and bound the approximation errors with certain ratios. Hoverer, the errors can be either overestimates or underestimates of the EMD. Whether an particular error is overestimate or underestimate can not be determined without computing the EMD. Therefore, we cannot use these approximate EMD algorithms to provide guaranteed lower bounds for EMD.

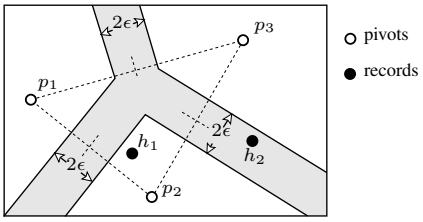


Fig. 7. Pruning and partitioning records in MRSimJoin with three pivots $\{p_1, p_2, p_3\}$; six reducers are responsible for records in p_1, p_2, p_3, and regions between $\{p_1, p_2\}$, $\{p_2, p_3\}$, $\{p_1, p_3\}$, respectively; the record h_1 is distributed to p_2 while the record h_2 is distributed to p_3 and $\{p_2, p_3\}$

B. Processing Join Using MapReduce

Implementing relational database operators using MapReduce attracts vast amount of attention [8]. The join operator and its variations are the focus of many studies. Four approaches for processing *equi-join* are studied [3] and the *theta-join* variation [12]. Much efforts have been devoted on the *set similarity join* [11], [21] and the *metric distance similarity join* [18]. In the set similarity join, data records are sets containing a small portion of elements from the element universe. This property enables the effective pruning techniques such as prefix-filtering [21] and inverted index [11]. In our problem, the property is not present since every histogram share exactly the same bin locations. It is not viable to prune a pair of histograms based on the absence of bins. In recent years, variations of similarity join such as top-k similarity join [7] and kNN similarity join [9], [25] using MapReduce have also been studied. Moreover, enhancement on MapReduce mechanisms may also speed up the join operations [6].

MRSimJoin [18] is the state-of-the-art MapReduce based solution for metric distance similarity joins. The intuition is to divide the space using *pivot* points such that data objects are assigned to nearest pivot to form multiple clusters. The data objects in the margin of clusters may have small distances hence should also be joined to guarantee the completeness. In MRSimJoin, mappers sample the pivots and assign data objects to clusters and then reducers join the records in the same cluster and the same margin. Fig. 7 illustrates an example where three pivots are used. Shaded areas are the three marginal regions with a width of 2ϵ. The technique involves substantial amount of distance computations on mappers since each data object needs to find its nearest pivot. The lower bounds of EMD cannot be integrated into the mappers of MRSimJoin since 1) most lower bounds are not metric; 2) even a metric lower bound may not preserve the *locality of EMD*, i.e., for an object, its nearest pivot in terms of the lower bound is not necessarily its nearest pivot in terms of EMD. Furthermore, in MRSimJoin if a reducer receives a large number of objects, it further samples pivots and partitions those objects. This is inefficient in our problem as it introduces more EMD computations.

VI. EXPERIMENTS

We present an extensive experimental study to evaluate the performance of MELODY-JOIN. Since there is no existing solutions specifically designed for the EMD distance similarity

TABLE II
DATASET SETTINGS USED IN EXPERIMENTS

Dataset	Cardinality	# of bins	Threshold values
CC	12K, 24K, 36K, **48K**	**32**	0.05, **0.1**, 0.15, 0.2
CL	12K, **24K**, 36K, 48K	**32**	0.025, **0.05**, 0.075, 0.1
MV, MH	12K, 24K, **48K**,	**30**	**0.04**, 0.05, 0.06, 0.07
MS	96K, 192K		0.03, **0.04**, 0.05, 0.06

TABLE III
EMD DISTRIBUTION IN DATASETS USED IN EXPERIMENTS

Dataset	0.001%	0.01%	0.1%	1%	100%	$D_{n,u}$
CC	0.0802	0.1422	0.2464	0.4102	4.1075	0.2181
CL	0.0145	0.0364	0.0819	0.1934	3.1073	0.1608
MV	0.0438	0.0605	0.0810	0.1184	2.3454	0.4282
MH	0.0468	0.0614	0.0814	0.1158	1.9938	0.3781
MS	0.0453	0.0566	0.0734	0.1052	1.7382	0.3309

join, we choose MRSimJoin, the state-of-the-art technique for metric distance similarity join, as the *baseline method*.

A. Datasets and Experimental Configurations

Datasets. We use five real datasets generated from two public image collections. The first collection is the COREL [20] containing 68,040 images. We use MPEG-7 *Dominant Color Histogram* and *Color Layout Histogram* descriptors to represent each image. For each descriptor, there are 32 bins (features). The location of bin is three-dimensional and the weight represents the color value in the specific locations on the image. We denote the two generated datasets as COREL Color (CC) and COREL Layout (CL). They are treated as separate datasets since we employ ℓ_2 distance as the ground distance yet the ℓ_2 distance between different descriptors are undefined. The second collection is the MIRFLICKR 1M [10]. This collection contains one million public-domain images collected from Flickr image sharing site. We use MPEG-7 *Edge Histogram* descriptor to represent each image. The edge histogram descriptor includes five types of edge features: vertical, horizontal, slash, backslash, and non-directional. The feature values are collected locally, semi- globally, and globally from the image, resulting to a histogram with 30 three-dimensional bins for each feature. We show the experiments on the histograms corresponding to the first three features, i.e., vertical, horizontal, and slash; the results of other two features are similar and omitted. We denote the three generated datasets as MIRFLICKR Vertical (MV), MIRFLICKR Horizontal (MH), and MIRFLICKR Slash (MS), respectively. Because there is no definition of ℓ_2 distance between different edge features, we use them as separate datasets. For all datasets, we uniformly sample specific numbers of images as listed in Table II. The default settings are in bold style. The EMD distribution of the five datasets are shown in Table III. The last column $D_{n,u}$ is the Kolmogorov-Smirnov statistic (ranging from 0 to 1) between the distances of a uniform distribution and a sample of EMD in the corresponding dataset. The larger this value suggests a more skewed distribution of the dataset. We choose the threshold values for datasets such that about $1\% \times 1\%$ (0.01%) pairs are retrieved by the query.

We use the elapsed time as the performance measure. This is because the EMD similarity join is computation-intensive, i.e., the CPU time dominates the response time. Also, the elapsed time is typically the basic unit for calculating the usage fee in a cloud service. For the same reason, our sampled datasets are relatively small because we report results that can be evaluated within a reasonable amount of time (10 hours). We run each experiment five times and report the median.

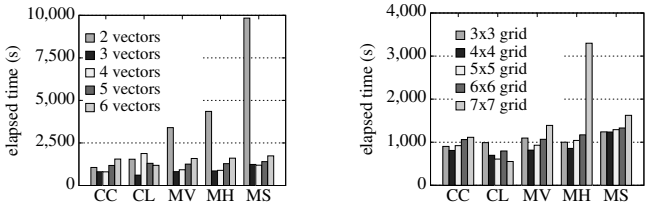
Experimental configurations. We conduct the experiments on an in-house 48-node Hadoop cluster (Cloudera CDH4 distribution). Each node runs at 3.2GHz and has 4GB memory. To further demonstrate the scalability of MELODY-JOIN, we conduct one set experiment for each dataset using the default settings on a cloud research facility provided by our university. We attempt several settings for MRSimJoin, yet it is always outperformed by MELODY-JOIN by an order of magnitude. Hence we focus on the scalability of MELODY-JOIN and omit the results of MRSimJoin here. On this cloud service, we provision 24 to 72 nodes.

Following the practice in related studies [9], [11], we conduct all experiments using self-joins. For MELODY-JOIN, we conduct experiments using jobs described in Algorithm 5, Algorithm 6, and Algorithm 8. For MRSimJoin, more pivots suggest larger pruning overhead of EMD computations while less pivots are more likely to produce unbalanced workloads. As a trade-off, we choose the number of pivots so that the number of reduce tasks generated fits to the capacity of our in-house cluster. *We integrate the same chain of lower bounds employed in* MELODY-JOIN *to the reducers of MRSimJoin to avoid unnecessary EMD computations*.

B. Evaluation of Parameters in MELODY-JOIN

Effect of the number p of projection vectors. By employing more projection vectors, MELODY-JOIN gains larger pruning power from multiple lower bounds. However, more projection vectors also suggests more composite cells, and therefore a larger overhead on pruning. We vary p from 2 to 6. The results are shown in Fig. 8a. As expected, when p grows, the performance MELODY- JOIN first improves then slightly degrades. The value that achieves the best performance is 3 in all datasets. This can be explained by the fact that the locations of bins in all datasets are three-dimensional. Three random projection vectors are adequate to nicely capture shapes of histograms in the projected spaces.

Effect of the granularity z^2 of grid division. The granularity of grids directly determines the granularity of the load balancing technique (as we group composite cells into groups to balance the workloads). Intuitively, a fine-grained grid (larger z) provides more composite cell cells with small number of records. This is more likely to balance the workloads, though may lead to a larger overhead of pruning. A coarse-grained grid (smaller z) provides less composite cell cells with large number of records. This risks of failures on balancing workloads, trading for a low overhead of pruning. The specific value of z that provides the best performance is likely to be correlated to the size of cluster. We vary the granularity z^2



(a) varying p from 2 to 6 (b) varying z^2 from 3^2 to 7^2

Fig. 8. Effects of parameters on the performance of MELODY-JOIN

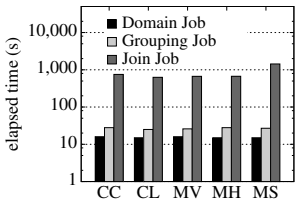


Fig. 9. The breakdown job completion time in MELODY-JOIN

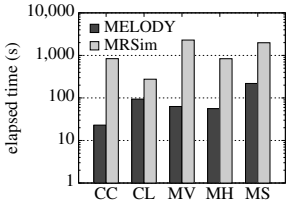


Fig. 10. The standard deviations on completion time of reducers

from 3^2 to 7^2 and evaluate the performance of MELODY-JOIN on all datasets. As shown in Fig. 8b, as z increases, the performance of MELODY-JOIN first improves then degrades. The best value z is 4 in most datasets, except that in CL datasets where 7 is the best value. This can be explained by the fact that the default setting of CL has the smallest cardinality among all datasets. The overhead of pruning is surpassed by the gained efficiency from more balanced workloads when the number of composite cell grows. In the following experiments, we set the value of p and z to 3 and 4 respectively.

C. Comparative Study

In this section, we first show the completion time of each job in MELODY-JOIN and measure the standard deviation on completion time of reducers in MRSimJoin and the last job of MELODY-JOIN. Then we vary dataset cardinality and the threshold value. In the end we vary the number of nodes in the cloud service to evaluate the scalability of MELODY-JOIN.

Breakdown Job Completion Time of MELODY-JOIN. The first two jobs in MELODY-JOIN are lightweight since they only perform operations of linear cost. We measure completion time for each job and present them in Fig. 9. As confirmed by the results, Job 3 dominates the overall performance of MELODY-JOIN by taking orders of magnitude longer to complete.

Standard Deviation of Reducer Completion Time. A job is complete only when all reduce tasks complete. Therefore, given the same workload, a smaller standard deviation on the completion time of reducers suggests more balanced workloads and therefore higher efficiency. We measure the standard deviations of reducers in Job 3 of MELODY-JOIN and MRSimJoin, the results of which is shown in Fig. 10. For all datasets, the standard deviation of reducer completion time in MELODY-JOIN is *an order of magnitude smaller* than that of MRSimJoin. This confirms the effectiveness of our proposed quantile grid and cardinality based grouping techniques in MELODY-JOIN.

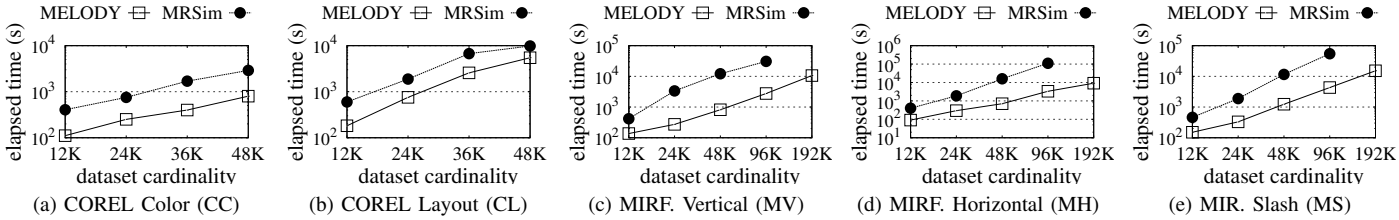
Effects of dataset cardinality $|H|$. The experimental results on varying dataset cardinality are presented in Fig. 11. The results for MRSimJoin on some large datasets are not shown because MRSimJoin did not complete within 10 hours. As shown in the figure, the elapsed time of both methods increases when the cardinality of datasets grows. On the two COREL datasets, MELODY-JOIN beats MRSimJoin by **1.8 to 4.2** times while on the three MIRFLICKR datasets, MELODY-JOIN beats MRSimJoin by **5.7 to 33.1** times. This can be explained by the fact that MIRFLICKR datasets are more skewed than COREL datasets, as shown in Table III, where MIRFLICKR datasets have larger $D_{n,u}$ values than COREL datasets. Unbalanced workloads are more likely to present when joining data objects in MIRFLCIKR datasets. While MRSimJoin severally suffers from the unbalanced workloads, MELODY-JOIN maintains its efficiency attributing to the effective quantile grid and load balance techniques. Moreover, the improvement of MELODY-JOIN over MRSimJoin also becomes larger when cardinality grows, suggesting the better scalability to dataset size of MELODY-JOIN when compared with MRSimJoin.

Effects of threshold value ϵ. Fig. 12 shows the results for varying the threshold value in five datasets. As demonstrated in the figure, both methods spend more time on larger threshold values, while MELODY-JOIN outperforms MRSimJoin in all settings by up to an order of magnitude. Again, due to the larger $D_{n,u}$ values in MIRFLICKR datasets, the improvement of MELODY-JOIN over MRSimJoin is larger in the three MIRFLICKR datasets (**8.7 to 22.6** times) than two COREL datasets (**2.5 to 13.4** times). Additionally, MELODY-JOIN is more responsive than MRSimJoin when the selectivity is high as it always one order of magnitude quicker than MRSimJoin when ϵ is small. This suggests the superior value of MELODY-JOIN in real applications that prefers the high selectivity.

Effects of the number of nodes in the cluster. Fig. 13 illustrates the results on all five datasets when we vary the number of nodes from 24 to 72 on the cloud research facility. MELODY-JOIN gradually speeds up as the number of nodes increases. On the CC, CL, MV, MH, and MS datasets, the speed-up ratios of MELODY-JOIN are 38.68%, 25.88%, 32.32%, 47.64%, and 35.15%, respectively. These speed-up ratios are not linear to the number of nodes because the more nodes in cluster, a record will be replicated and distributed to more nodes, ultimately increasing the overall workloads.

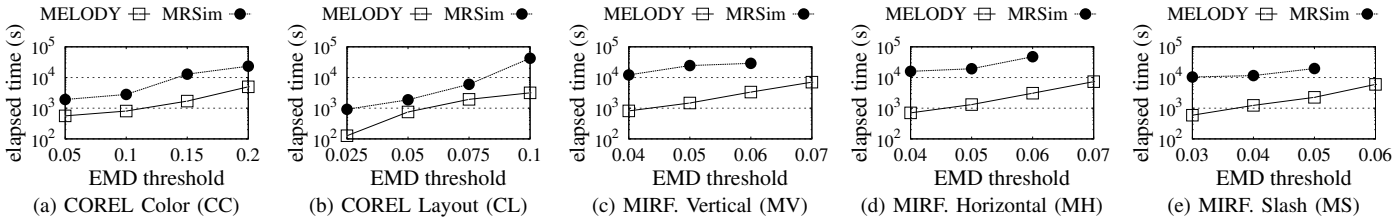
VII. CONCLUSION

In this paper we proposed MELODY-JOIN, a novel framework for processing the EMD similarity join based on MapReduce. MELODY-JOIN employs the computationally cheap lower bounds to prune and partition data which avoids a large number of EMD computations. Multiple EMD lower bounds can be plugged into MELODY-JOIN. We further proposed the quantile based grid and the cardinality based grouping techniques to address the problem of unbalanced workloads. We conducted extensive experiments on various real datasets, confirming the effectiveness and efficiency of MELODY-JOIN. As demonstrated by the results, MELODY-JOIN outperforms the state-of-the-art technique typically by an order of magnitude and it scales up and out well.



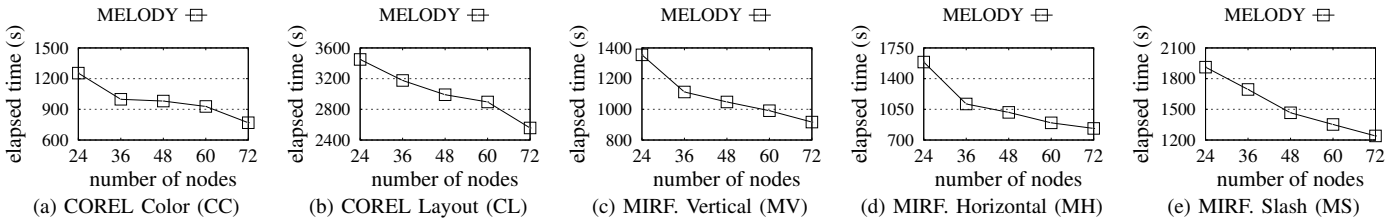
(a) COREL Color (CC) (b) COREL Layout (CL) (c) MIRF. Vertical (MV) (d) MIRF. Horizontal (MH) (e) MIR. Slash (MS)

Fig. 11. Varying the cardinality of H in five datasets on the 48-node in-house Hadoop cluster



(a) COREL Color (CC) (b) COREL Layout (CL) (c) MIRF. Vertical (MV) (d) MIRF. Horizontal (MH) (e) MIRF. Slash (MS)

Fig. 12. Varying the EMD threshold ϵ in five datasets on the 48-node in-house Hadoop cluster



(a) COREL Color (CC) (b) COREL Layout (CL) (c) MIRF. Vertical (MV) (d) MIRF. Horizontal (MH) (e) MIRF. Slash (MS)

Fig. 13. Scaling MELODY-JOIN from 24 nodes to 72 nodes on the cloud research facility provided by our university

ACKNOWLEDGMENT

The work is supported by the Australian Research Council (ARC) *Discovery Project* DP130104587, the National Natural Science Foundation of China (No. 61272065), the Natural Science Foundation of Guangdong Province, China (No. S2012010009311), and the Fundamental Research Funds for the Central Universities, SCUT(Grant No. 2012ZZ0088). Dr. Rui Zhang is supported by the ARC *Future Fellowships Project* FT120100832.

REFERENCES

[1] A. Andoni, K. D. Ba, P. Indyk, and D. Woodruff, "Efficient sketches for earth-mover distance, with applications," in *FOCS*, 2008.

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

[3] S. Blanas, J. M. Patel, V. Ercegovac, and J. Rao, "A comparison of join algorithms for log processing in mapreduce," in *SIGMOD*, 2010.

[4] S. Cohen and L. Guibas, "The earth mover's distance: Lower bounds and invariance under translation," Stanford University, DTIC Report, 1997.

[5] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *OSDI*, 2004.

[6] L. Ding, G. Wang, J. Xin, X. Wang, S. Huang, and R. Zhang, "Commapreduce: An improvement of mapreduce with lightweight communication mechanisms," *Data and Knowledge Engineering*, 2012.

[7] Y. Kim and K. Shim, "Parallel top-k similarity join algorithms using mapreduce," in *ICDE*, 2012.

[8] F. Li, B. C. Ooi, M. T. Ozsu, and S. Wu, "Distributed data management using mapreduce," *ACM Computing Survey*, 2013.

[9] W. Lu, Y. Shen, S. Chen, and B. C. Ooi, "Efficient processing of k nearest neighbor joins using mapreduce," *PVLDB*, 2012.

[10] B. T. Mark J. Huiskes and M. S. Lew, "New trends and ideas in visual concept detection: The mir flickr retrieval evaluation initiative," in *MIR*, 2010.

[11] A. Metwally and C. Faloutsos, "V-smart-join: A scalable mapreduce framework for all-pair similarity joins of multisets and vectors," *PVLDB*, vol. 5, no. 8, 2012.

[12] A. Okcan and M. Riedewald, "Processing theta-join using mapreduce," in *SIGMOD*, 2011.

[13] Z. Ren, J. Yuan, and Z. Zhang, "Robust hand gesture recognition based on finger-earth mover's distance with commodity depth camera," in *MM*, 2011.

[14] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, pp. 99–121, 2000.

[15] B. E. Ruttenberg and A. K. Singh, "Indexing the earth mover's distance using normal distributions," *PVLDB*, 2012.

[16] M. A. Ruzon and C. Tomasi, "Edge, junction, and corner detection using color distributions," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 2001.

[17] S. Shirdhonkar and D. W. Jacobs, "Approximate earth mover's distance in linear time," in *CVPR*, 2008.

[18] Y. N. Silva and J. M. Reed, "Exploiting mapreduce-based similarity joins," in *SIGMOD*, 2012.

[19] A. Tung, R. Zhang, N. Koudas, and B. C. Ooi, "Similarity search: A matching based approach," in *VLDB*, 2006.

[20] UCI. (2013) Corel image features data set. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Corel+Image+Features

[21] R. Vernica, M. J. Carey, and C. Li, "Efficient parallel set-similarity joins using mapreduce," in *SIGMOD*, 2010.

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

[23] D. Xu, T.-J. Cham, S. Yan, and S.-F. Chang, "Near duplicate image identication with spatially aligned pyramid matching," in *CVPR*, 2008.

[24] 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," *The VLDB Journal*, 2012.

[25] C. Zhang, F. Li, and J. Jestes, "Efficient parallel knn joins for large data in mapreduce," in *EDBT*, 2012.