MGD: A Utility Metric for Private Data Publication

Ninghui Li, Zitao Li, Trung Dang, Tianhao Wang
Team DPSyn, Purdue University
{ninghui, zitaoli, dang28, tianhaowang}@purdue.edu

1 Executive Summary

1.1 Metric Overview

We propose MarGinal Difference (`MGD`), a utility metric for private data publication. `MGD` assigns a difference score between a pair of datasets $\langle D_S, D_T \rangle$, where D_S is the synthesized dataset, and D_T is the ground truth. The high level idea behind `MGD` is to measure the differences between many pairs marginal tables, each pair having one computed from D_S and one from D_T. For measuring the difference between a pair of marginal tables, we introduce Approximate Earth Mover Cost (`AEMC`), which considers both semantic meanings of attribute values and the noisy nature of the synthesized dataset. While the current challenge focuses on temporal map data, `MGD` is designed for relational datasets in general, and can be naturally applied to temporal map data.

Design Desiderata . To turn the high-level idea of measuring differences between marginal tables into a concrete metric, we need to specify the following: which marginal tables are used in computing the score, how to measure the differences between two marginal tables, and how to combine these measurements into one score. We use the following design objectives to guide our choices for fleshing out the details of `MGD`.

- Flexibility. The set of marginal tables and their weights in deciding the final score can be configured. Under the constraints of Difference Privacy (DP), one cannot preserve all distribution information in the input dataset. Fortunately, it is often unnecessary for a synthetic dataset to preserver all distributions for it to be useful. Oftentimes not all distributions are equally important; however, the relative importance among different distributions, when they exist, cannot be determined without considering the attributes in the datasets and the application domain.

- Balancing Flexibility with Ease of Use. To provide flexibility, a metric needs to have configurable parameters; however, this may make a metric difficult to use. We solve this problem in two ways. First, instead of making everything configurable, we fix some aspects of the metric when we believe doing so preserves sufficient flexibility. Second, for almost all parameters, we provide default values that we hope would work for most application scenarios.

- Accommodation of Numerical Attributes. For attributes with numerical values, we want to take into consideration of the natural semantic distance between values. For example, for an age attribute, 19 is not very different from 20, but is very different from 90, and treating them just as three different values loses this semantic meaning.

- Accommodation of Structured Attributes. For many categorical attributes, the semantic meanings also suggest natural semantic distances between values. The metric should be able to use these semantic meanings when the application domain can benefit from doing so.

- Awareness of Noises. Under DP the data are noisy, and it is understood that small differences cannot be relied upon. The metric should reflect this, avoiding being dominated by many small differences.

Parameter of the `MGD` metric . The metric is parameterized by the following.

- Data Schema. This specifies what attributes are in the dataset, the domain of possible values for each attribute, and any relationships between the values. At the minimum, it should be clear whether each attribute is ordinal or not, and what are the set of values that can be taken for each attribute. Optionally, one can also provide generalization hierarchies for these attributes.

- Target Marginal Schemas and Weights. This includes a set of marginal schemas where each schema specifies the attributes to be included. One applies each schema to D_S and D_T to obtain marginal tables, and computes the AEMC score between the two resulting marginal tables. The final score is the weighted average of these AEMC scores, where the weights are equal by default, but can be configured if one wants to.

Approximate Earth Mover Cost (AEMC) between Two Marginals . For measuring differences between two marginals, commonly used metrics include L_1, L_2, KS divergence, Jensen-Shannon distance, and so on. However, these metrics do not take into consideration of the underlying semantic distances between the values. The desire to use semantic distance values naturally led us to the earth mover's distance (EMD). Informally, if two distributions are interpreted as two different ways of piling up a certain amount of dirt over the region, the EMD is the minimum cost of turning one pile into the other, where the cost is assumed to be the amount of moving dirt times the distance by which it is moved.

We adapt EMD in three ways, to be more suitable for our purpose, and call the result AEMC. First, instead of normalizing the margainal tables into probability distributions (i.e., all entries sum up to 1), we first use the unnormalized record counts to compute a score so that the absolute number of records also conveys information. The AEMC is obtained by dividing this score with the total number of records in the ground truth dataset. Second, in addition to moving counts from one bin to another, we also allow adding/removing counts directly. This helps deal with unequal total counts between the two datasets, and also make the metric a generalization of the L_1 distance between two marginal tables. Third, the earth moving does not need to get the two marginal tables exactly the same, but only so that the difference is below a threshold Δ. This avoids penalizing small differences in the counts. Without this feature, the score between two marginal tables can be dominated by cells that have small values when there are many such cells.

AEMC can be computed either by formulating it as a linear program, and using generic linear programming solver, or by formulating it as a flow problem in graphs and adapting existing algorithms to compute it. We have implemented both approaches, and provide the details in the Appendix. Experiments show that modeling AEMC as a flow problem results in a fast computation for fairly large marginals.

1.2 Real World Use Cases

We consider two example use cases. Here we describe, for each case, the datasets and the desired information that one wants to capture in the metrics. In Section 3, we show how these objectives can be achieved by using suitable configurations of MGD.

Example 1. Police Incident Data. For our first example, we consider the dataset used in Sprint 1 of the current NIST Challenge, namely Baltimore 911 Call and Police Incident Data. There sample dataset in the contest has three attributes: time (12 months), neighborhood (278 values), and incident type (174 values). One can envision the following types of information that one wants to extract from the data.

- Per neighborhood-month distribution of incident types. This is what the pie-chart metric used in Sprint 1 tries to measure. Conceptually, the dataset is first partitioned into $12 \times 278 = 3336$ parts (one for each month and neighborhood), and for each part the distribution over the 174 incident type is compared to the ground truth.

- Per neighborhood total incidents over time. Here we assume that one is only interested in how the total number of incidents (aggregating over all incident types) changes over time for each neighborhood.

- Distribution of certain types of incidents over neighborhoods. This may be helpful for determining whether adequate resources are available to handle incidents of certain types. When there exists meaningful distance data between neighborhoods, e.g., based on travel time or on organizational hierarchy, such information can be used in computing MGD.

Example 2. PUMS Data. For our second example, we use the dataset used in Phase 3 of the 2018-2019 Differential Privacy Synthetic Data challenge, the Public Use Microdata Sample (PUMS) of the 1940 USA Census Data for the State of Colorado, fetched from the IPUMS USA Website. The dataset has 98 attributes. In the challenge, three metrics were used.

- Density Estimation. For this metric, the scoring algorithm randomly samples 300 marginal schemas, each with three randomly chosen attributes. Then for each marginal schema, it computes the normalized marginal tables from the synthetic dataset and from the ground truth dataset, and then uses the L_1 distance between the two marginal table as the penalty. We will show how un-normalized L_1 distance, which is highly correlated with normalized L_1, can be specified in `MGD`.
- Range Query. For this metric, the scoring algorithm randomly samples 300 range queries and assesses the accuracy of using the synthetic dataset to answer these queries. To generate a range query, one first randomly samples a subset of attributes, with each attribute having 33% chance to be selected. Then, for each selected attribute, a query condition is randomly generated. This cannot be directly implemented using `MGD`; however, accuracy for range queries is highly correlated with scores computed from low-degree marginals.
- Gini Index and Rank Accuracy. The two scores are calculated based on the columns named SEX and INCWAGE (income wage) for each city in the dataset. The first score component is based on the mean-square deviation between Gini indices obtained for the original and the synthesized dataset, averaged over the cities in the original dataset. For the second score component, one ranks the cities by the gender pay gap, and calculates the mean-square deviation between the resulting city ranks in the original and privatized datasets. This is highly correlated with accuracy on the 3-way marginal over CITY, SEX, and INCWAGE.

2 Definition of the `MGD` Metric

2.1 Technical Background

Earth Mover Distance (EMD). In statistics, EMD measures the distances between two probability distributions. EMD has been considered as an useful tool in computer vision area for the tasks including image retrieval [3], feature matching [4], and face verification [10]. In recent years, EMD is also widely used in deep learning models, such as generative adversarial network (GAN) [2, 5, 6]. EMD is also used in document similarity analysis [7] on high dimensional and sparse Bag-of-words vectors.

More formally, EMD takes P and Q, each being a size T vector of non-negative values such that the sum over each vector is 1. Let $\mathbf{M}$ be a matrix of size $T \times T$ in which $\mathbf{M}_{ij} \geq 0$ is the cost of moving an element from the i^{th} position to the j^{th} position ($\mathbf{M}_{ii} = 0$ for all i values). The standard EMD is defined as:

$$\min_{\mathbf{X}} \sum_{i,j} \mathbf{X}_{ij}\mathbf{M}_{ij} \qquad \text{s.t. } \forall i, \quad \sum_{j} \mathbf{X}_{ij} = P_i \quad \forall j, \quad \sum_{i} \mathbf{X}_{ij} = Q_j, \quad \forall i, j, \mathbf{X}_{ij} \geq 0$$

Intuitively, $\mathbf{X}_{ij}$ gives the amount one moves from the i^{th} element to the j^{th} element. EMD can be easily extended to vectors that are not probability distributions. There are also extensions to EMD that accommodate the situation where the sum of elements in P and the sum of elements in Q are different, either by allowing free disposal of excess quantities, or introducing cost for adding/removing values. We use the latter approach in our proposal.

Linear Programs (LP). Linear programming is a technique used to optimize linear objective functions subject to linear equality and linear inequality constraints. One of the most canonical application of linear programming is solving network maximum flow problem. Also, it can be applied on regression problems and linear classification problems. Computing EMD is a special case of linear programming problem. There exist software packages that can solve large LP instances reasonably fast.

Flow Algorithms. EMD can be computed by solving an instance of the transportation problem, using any algorithm for minimum cost flow problem, e.g. the network simplex algorithm. Since such algorithms

exploit the specific nature of the transportation problem, they are faster than using LP solvers for the same problem size.

Additional Resources. Any software package that can solve linear programming problems can be used to calculate EMD and the new metric we proposed in this document. There are both open sourced and commercial packages. With Python, there are `CVXOPT` and `CVXPY`, both of which are open source library that can be used to solve linear programming problems. `CPLEX`, a the commercial package developed by IBM, is another choice to solve the optimization problem. To accelerate the computation of `AEMC`, we reduce the problem to a special kind of min-cost problem and use OR-Tools solver (`ortools` python package) to solve the problem.

2.2 Formal Definition of `AEMC`

`AEMC` extends EMD in a few ways, to be more suitable for our purpose. More formally, `AEMC` is parameterized by two parameters: (1) $\mathbf{M}$ is a $T \times T$ matrix where each cell $\mathbf{M}_{ij}$ is the distance between the i-th bin and the j-th bin. It is required that $\mathbf{M}_{ij} \in [0, 1] \cup \{\infty\}$ and $\forall_i\ \mathbf{M}_{ii} = 0$. Setting $\mathbf{M}_{ij} = \infty$ means prohibiting moving from bin i to bin j. And (2) $\Delta > 0$ is a threshold such that differences between Δ are tolerated. `AEMC` is applied to a pair of size T marginals P and Q, where each cell corresponds to a count in the marginals. We view P and Q as two vectorized/flattened marginal tables of size $T \times 1$. P denotes the marginal computed from the synthesized dataset, and Q the one computed from the ground truth. `AEMC` is defined as:

$$\texttt{AEMC}_{\mathbf{M},\Delta}(P, Q) = \frac{1}{||Q||_1} \min_{\mathbf{X}} \sum_{i,j} \mathbf{X}_{ij}\mathbf{M}_{ij} + \sum_j \max\left\{\left|\sum_i \mathbf{X}_{ij} - Q_j\right| - \Delta, 0\right\} \quad \text{s.t. } \forall i, \quad \sum_j \mathbf{X}_{ij} = P_i \quad (1)$$

Here $||Q||_1$ is the L_1 norm of Q, i.e., the sum of all components in the vector Q. The factor $\frac{1}{||Q||_1}$ normalizes the score so that it is independent from the total number of records in the dataset. Intuitively, each movement scheme is specified by a $T \times T$ matrix $\mathbf{X}$, such that $\mathbf{X}_{ij}$ gives the amount of moving from bin i to bin j. The condition $\sum_j \mathbf{X}_{ij} = P_i$ says that the total amount of quantity moving from bin i (some of which can be to itself) is exactly P_i. We want to minimize the cost, which has two components. The first one, $\sum_{i,j} \mathbf{X}_{ij}\mathbf{M}_{ij}$, is the cost of moving. Here we define $0 \cdot \infty$ to be 0; thus when $\mathbf{M}_{ij} = \infty$, the corresponding $\mathbf{X}_{ij}$ should be 0 to minimize the above formula. The second one, $\sum_j \max\{|\sum_i \mathbf{X}_{ij} - Q_j| - \Delta, 0\}$, computes the penalty where the result after moving still differs from the ground truth Q. When the difference is below a threshold Δ, the penalty stays at 0. However, when the difference is above Δ, the part above Δ is penalized, representing the cost of adding/removing counts.

When $\mathbf{M}_{ij} = \infty$ for all $i \neq j$, and $\Delta = 0$, $\texttt{AEMC}_{\mathbf{M},\Delta}$ approximates the L_1 distance. When P and Q are the same, or when their L_∞ distance is $\leq \Delta$, the `AEMC` between them is 0.

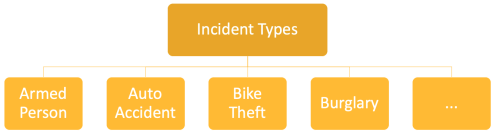
2.3 Parameters and Configurations for `MGD`

The `MGD` metric has the following parameters. While there are many parameters that can be configured, we list the default choices for almost all of them.

2.3.1 Data Schema

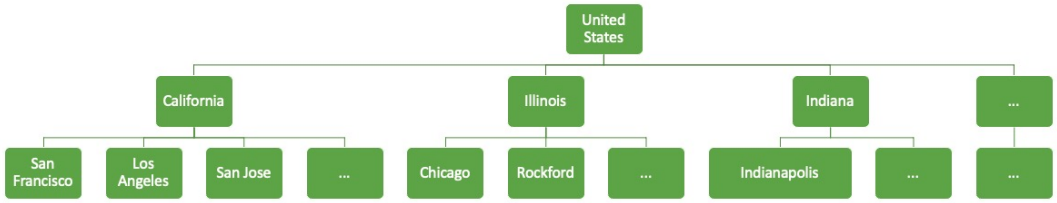
Any dataset needs accompanying data schemas for it to make sense. To apply `MGD`, we require that for each attribute, the following information can be extracted from the data schema.

- *IsOrdinal*. Whether the attribute is ordinal or not, i.e., whether there is a linear ordering among all values.
- *Domain* D. The set of all possible values for this attribute.
- *Generalization Hierarchy* H. A rooted tree such that:
  - There is one-to-one correspondence between the leaves and each value in the domain D.



(a) Generalization hierarchy of attribute "incident type" in 2019 Baltimore 911-Call and Police Incident data

.



(b) Generalization hierarchy of cities in United States

.

Figure 1: Generalization hierarchy of categorical attributes

.

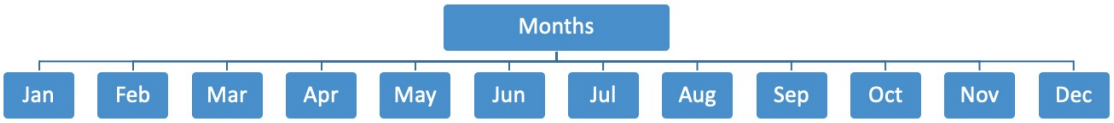
- All leaves are at the same depth (same distance from the root).
- If an attribute is ordinal, the nodes at each level should appear (from left to right) in the order of small to large.

Whether an attribute is ordinal or not and its domain are usually determined by the nature of the dataset. When an attribute does not have an explicitly specified generalization hierarchy, we use a default hierarchy that has two levels, where all values in the domain are the leaves, and there is one single root value. Allowing Generalization Hierarchies serves several purposes. First, they can be used to capture semantic distances among values in non-ordinal attributes. Second, they allow one to use marginals that involve attributes with many values while keeping the marginal size feasible. One can use generalized values for the attributes that have large domains. Third, for similar reasons as above, they enable marginals that involve more attributes.

Semantic Distance. Given a generalization hierarchy, we define the level of a node as the number of edges on the path from the node to the root. Thus the root has level 0, and the root's children have level 1, and so on. Whether an attribute is ordinal or not and its generalization hierarchy together fully determine the *Semantic Distance function* sd, which assigns a real value between 0 and 1 to *each pair of nodes in the same level*. For an ordinal attribute, when a level has $k > 1$ values, the distance between the i-th and the j-th value at that level is $\frac{|j-i|}{k-1}$. For two values of a non-ordinal attribute at level t, let s be the level of the lowest common ancestor of the two values, then the distance between the two values is defined to be $\frac{t-s}{t}$. Note that for any attribute in a marginal, only values from one specific level can appear, and we thus only need semantic distances between values at the same level. The definition normalizes the distances to be in $[0, 1]$ no matter which level is used.

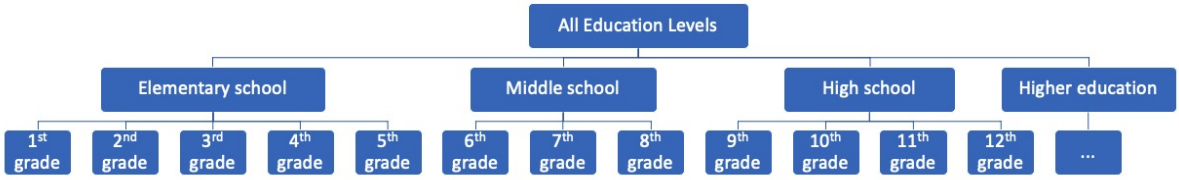
Figure 1 shows two categorical attribute examples in generalization hierarchy structure. Figure 1a shows a default generalization hierarchy that has only two levels. The semantic distances between any two values, for example sd("Armed Person", "Bike Theft"), are all 1. Figure 1b shows the case that a categorical attribute has a customized hierarchy. Thus, the distance between any cities in the same state is $\frac{1}{2}$; the distance between any cities in different states and the distance between two states are both 1.

Figure 2 shows two categorical attribute examples in generalization hierarchy structure. For Figure 2a, where months are ordinal values, the semantic distance between any two months is the ordinal difference between them divided by 11. Figure 2b shows a ordinal attribute hierarchy with more than two levels. In this case, the semantic distance are defined for nodes in the same level. sd("1^{st} grade", "8^{th} grade") =



(a) Generalization hierarchy of attribute "month" in 2019 Baltimore 911-Call and Police Incident data

.



(b) Generalization hierarchy of education levels in United States

.

Figure 2: Generalization hierarchy of categorical attributes

.

$\frac{8-1}{13-1} = \frac{7}{12}$ because there are 13 values in leave level and the distance between those two values are 7; $\mathsf{sd}(\text{"Elementary school"}, \text{"Middle school"}) = \frac{2-1}{4-1} = \frac{1}{3}$ because there are only 4 values in the second level.

2.3.2 Target Marginals

MGD requires the specification of a set Φ of target marginals, each $\phi \in \Phi$ consisting of the following:

- Attributes in the Marginal. A set of attributes that are included in the marginal, and for each selected attribute, which level of the generalization hierarchy is used in computing the marginal. The default level for each selected attribute is the leaf level. This defines all the bins in the target marginal. (Alternatively, this can be defined as selecting a generalization level for every attribute. If the root level is selected for an attribute, then all records have the same value, and that attribute is effectively not included in the marginal.)

- Attribute Weights. A weight w_a for each selected attribute a. The weight $w_a \in \{\infty\} \cup [0, 1]$, and if any attribute has a weight in $[0, 1]$, such weights for all attributes sum up to 1. If an attribute is assigned a weight of ∞, it means that when computing the difference score on this marginal, one does not want to consider moving between bins where this attribute has different values.

  The assigned weights and the semantic distance function sd together define the bin distance matrix $\mathbf{M}^{\phi}$ needed for computing the AEMC. Given two bins/cells b and b' in the marginal, if there exists an attribute such that b and b' differ in an attribute that has a weight of ∞, the distance between b and b' is ∞. Otherwise, the bin distance:

$$\mathbf{M}^{\phi}_{b,b'} = \sum_{a} w_a \cdot \mathsf{sd}(b_a, b'_a) \tag{2}$$

  where b_a denotes value for attribute a of the bin b. The matrix $\mathbf{M}^{\phi}$ is used in Equation (1) to calculate AEMC. Note that the distance between two cells are either ∞ or a value between $[0, 1]$. When the distance is ∞, this means that for this marginal one does not want to moving between the two bins in computing the earth mover distance.

  The default for weight assignment is to assign the weight of ∞ to each categorical attribute, and equal weight to each numerical attribute.

- Tolerance threshold Δ^ϕ. The threshold under which differences in bins are not penalized in AEMC. (See Equation (1)).
- Marginal Score Weight ω^ϕ, which is a positive number. This specifies how much the AEMC computed on the marginal ϕ contributes to the final MGD score.

Summary of Difference Measures in defining MGD . There are several notions that measure differences between two things here. For clarity, we recap them here.

- Semantic Distance sd is for the distance between two values for the same attribute. It is defined for any pair of values that are on the same level of the generalization hierarchy, and is fully determined by the generalization hierarchy and whether the attribute is ordinal or not. Its value is always in $[0, 1]$.
- Bin Distance $\mathbf{M}^\phi$, defined in Equation (2), uses the semantic distance to define the distance between two bins in one marginal. Each bin is specified by the value that each attribute can take. $\mathbf{M}^\phi$ is defined as the weighted sum of semantic distances on all attributes. It is a generalization of the Manhattan distance by adding a weight to each dimension. We choose to use generalization of Manhattan distance (instead of, e.g., Euclidean distance) in part because this enables faster computation of AEMC.
- AEMC, defined in Equation (1), is for measuring the difference between a pair of marginal tables, one computed from the synthesized dataset, and the other from the ground truth dataset.
- MGD score is the weighted average of the AEMC score for all marginals in Φ. More specifically:

$$\texttt{MGD}_\Phi(D_S, D_T) = \left(\sum_{\phi\in\Phi} \omega^\phi \cdot \texttt{AEMC}_{\mathbf{M}^\phi,\Delta^\phi}\left(\phi(D_S), \phi(D_T)\right)\right) / \left(\sum_{\phi\in\Phi} \omega^\phi\right) \tag{3}$$

  where $\phi(D)$ is the marginal table obtained from dataset D using the marginal schema ϕ and flattened to a vector.

2.4 Snapshot Mode and Deep Dive Mode

MGD provides a single final score, which provides a Snapshot Mode measurement. In addition, MGD is based on the AEMC scores for different marginals. Examining these AEMC scores provides a deeper dive, as one can see which marginals are preserved well, and which are not. It is also possible to expose more detailed information in computing AEMC to provide even deeper dive. For example, within one marginal, we can compute the contribution of each attribute to the AEMC score, illustrating the information on which attribute is less accurately preserved. It is even possible to compute the contribution of individual bins to the AEMC score and outputting the ones that have the most contribution, illustrating where the errors concentrate. We have implemented MGD and can provide APIs to enable extraction of these internal values.

3 Applications of the MGD Metric

Here we discuss applications of the MGD metric. We first discuss how to apply it to temporal map data, and then show how to configure the MGD metric to measure the desired values in the two examples in Section 1.2.

3.1 Application to Temporal Map Data

Temporal map data are relational data where some important attributes represent time and space. When applying MGD to temporal map data, there are several considerations.

- The temporal spatial nature of the dataset can guide the choice of which marginals are included in Φ to compute the MGD score. For example, when there are 40 attributes, considering all 3-way marginals may be too many, and one may want to consider all 3-way marginals that include at least one of the temporal or spatial attributes.

- The temporal attribute is naturally ordinal. When temporal attributes are used in NIST competitions, they are often processed to have a limited set of values. For example, the Police Incident Data used in Spring 1 generalizes the time attribute to month. Similarly the time attribute in the San Francisco Fire dataset, which spans several years, was bucketized to 100 values. In some sense, the ground truth dataset has already lost much information. This was because using finer grained values results in sparse distribution, which previous metrics cannot handle well. `MGD` has several features that make it more capable of handling finer-grained ordinal attributes. First the usage of semantic distance enables one to distinguish synthetic datasets that have the time attribute approximately, but not completely correct. Second, by defining a generalization hierarchy over the time attribute, one has the flexibility of measuring both more precise distribution over the time attribute (e.g., by using a one-way marginal over the time), and correlation of time with other attributes (e.g., by using multi-way marginals in which a more coarse-grained values for time are used).

- The spatial attributes are similar to the time attributes in that they can be ordinal. It is also possible to extend `MGD` to better handle spatial attributes. For example, for computational efficiency considerations, we use the generalization hierarchy to derive the semantic distance function. However, when there are natural distance measures for a spatial attribute, it is possible to directly specify a distance matrix that assigns a distance to any two values. The tradeoff is that we can no longer exploit the hierarchical structure underlying the semantic distance function, and have to explicit consider movement between any pair of values, resulting in a quadratic number of movement variables.

3.2 Application to Example 1: Police Incident Data

In Section 1.2, we listed example datasets and kinds of information one may be interested in. Here we discuss how they can be captured using `MGD`.

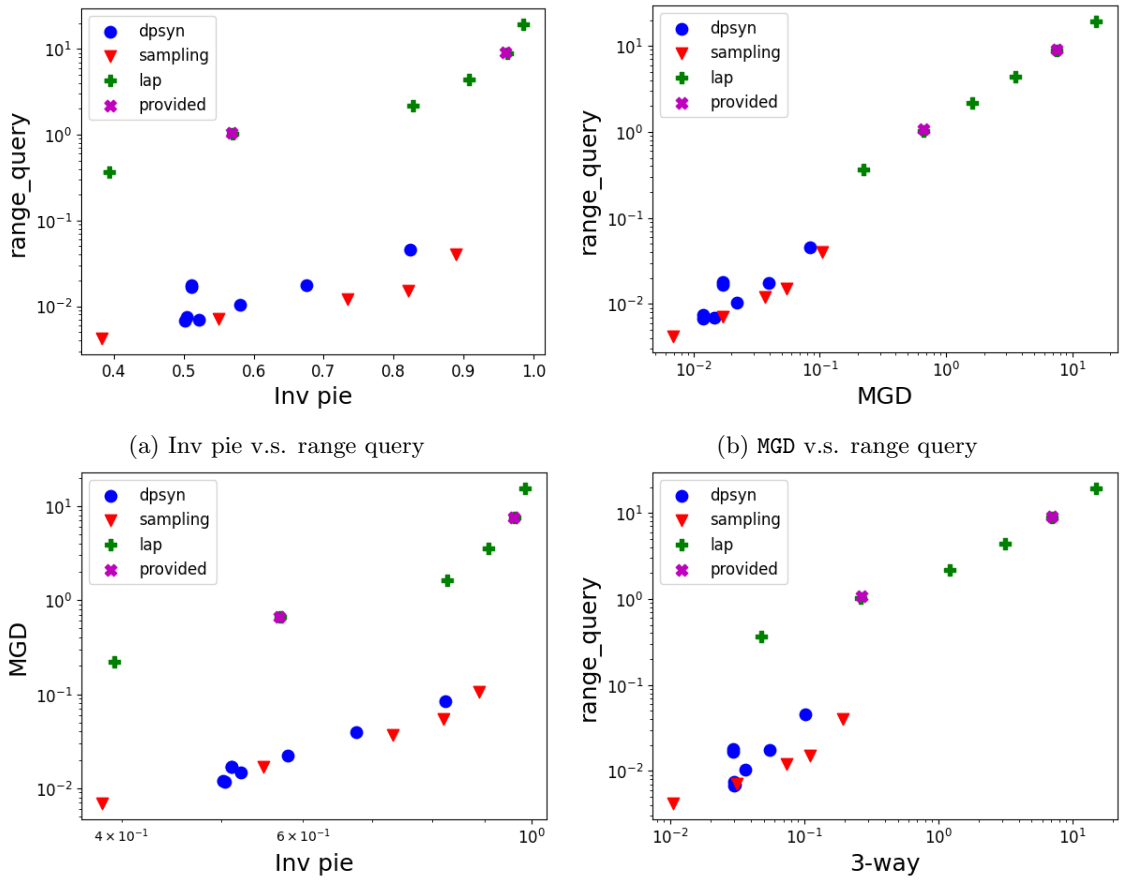
- Per neighborhood-month distribution of incident types. This is what the pie-chart metric used in Sprint 1 tries to measure. To simulate this using `MGD`, one uses a 3-way marginal over time, neighborhood, and incident type, with all weights being ∞, and a suitable Δ, which can be a small constant. (We use $\Delta = 2$ in our experiments. Such a metric is similar to L_1 distance, but ignores the impact of small differences, which is in the same spirit as the pie-chart metric's ignoring densities for incident types that are below 5%.

- Per neighborhood total incidents over time. Here we assume that one wants to examine how the total number of incidents changes over time for different neighborhood. One thus care about the accuracy of the two-way marginal over neighborhood and time.

- Distribution of certain types of incidents over neighborhoods. To determine whether adequate resources are available to handle incidents of certain types, we can consider a two-way marginal over neighborhood and incident type. When we want to analyze different incident types separately, we define the weight for incident type to be ∞ so that counts from bins with different incident types cannot flow into each other. The overall score thus reflects accuracy within each incident type. One can also dive down into the details and look at movement cost for each incident type separately. Assuming that a meaningful distance measure exists for neighborhood, one can run hierarchical clustering to come up with a generalization hierarchy.

When one does not have a specific goal in mind, one could choose to measure the three one-way marginals, the three two-way marginals, and the three-way marginal. One possible marginal weight assignment is 1/9 for each one-way marginals, 1/9 for two-way marginals, and 1/3 for the three-way marginal.

Figure 3 shows four scores for different synthetic datasets: (1) average error for randomly generated range queries; (2) Inv-Pie (short for Inverse Pie Chart), defined to be $(1 - S/3336)$, where S is the pie-chart score of the dataset; (3) the `MGD` score using the above weighting scheme (i.e., 1/9 for each of 1-way and 2-way marginals, and 1/3 for the 3-way marginal); and (4) the `AEMC` score for the 3-way marginal, which can also be viewed as `MGD` score that uses only the 3-way marginal. Because the range for (1), (3), and (4) is large, they are plotted on log-scale.



(a) Inv pie v.s. range query

(b) `MGD` v.s. range query

(c) Inv pie v.s. `AEMC` (3-way)

(d) `AEMC` (3-way) v.s. range query

Figure 3: Comparison between range query, Inv Pie, `MGD` and `AEMC` (3-way). For `MGD`, we user weight $\frac{1}{3}$ for the 3-way marginal, ant $\frac{1}{9}$ for the other 6 marginals

.

Each point in Figure 3 represents one dataset, and a few different methods are used to generate synthetic data. See Appendix B for details.

From Figure 3, we can see that MGD correlates well with the range query error. AEMC correlates less well, because it considers only accuracy of individual cells of 3-way marginals, and not accumulated errors over a range of values. The pie-chart score does not correlates well with range query error, in part because it was not designed to do so.

3.3 Application to Example 2: PUMS Data

For our second example, we use the dataset used in Phase 3 of the Differential Privacy Synthetic Data challenge, the Public Use Microdata Sample (PUMS) of the 1940 USA Census Data. The dataset has 98 attributes. In the challenge, three metrics were used.

- Density Estimation. For this metric, the scoring algorithm randomly samples 300 marginal schemas, each with three randomly chosen attributes. Then for each marginal schema, compute the normalized marginal tables from the synthetic dataset and from the ground truth dataset, and then use the L_1 distance between the two marginal table as the penalty. This can be defined in a straightforward way using MGD.
- Range Query. For this metric, the scoring algorithm randomly samples 300 range queries and assesses the accuracy of using the synthetic dataset to answer these queries. This cannot be directly implemented using MGD, although this score is highly correlated with the above density estimation score computed from 3-way marginals, as we observed during the competition.
- Gini Index and Rank Accuracy. For each city present in the dataset (as specified by the CITY column) we calculate, based on the SEX and INCWAGE columns, Gini index and gender pay gap. This can be captured by using the 3-way marginal over CITY, SEX, and INCWAGE. If a dataset more accurately preserves this 3-way marginal, it would also result in more accurate estimations of Gini index and rank of cities based on gender pay gap.

If one wants a MGD scheme that correlates highly with the scoring scheme used in this challenge (which involves the three above-mentioned metrics), one can use a sufficiently large number of 3-way marginals, and ensuring that the marginal involving CITY, SEX, and INCWAGE has a high weight, while treating INCWAGE as an ordinal attribute.

4 Computation Issues

Computing the MGD requires computing the AEMC between two marginals. We now present two approaches for computing AEMC. The first approach, which is more general, is to transform the optimization problem in computing AEMC into a linear program, and use existing LP solver to solve it. The second approach, which is more efficient, is to model AEMC as as a variant of the min-cost network flow problem.

4.1 Transform AEMC as a Linear Program

Recall that AEMC is defined as:

$$\texttt{AEMC}_{\mathbf{M},\Delta}(P,Q) = \min_{\mathbf{X}} \sum_{i,j} \mathbf{X}_{ij}\mathbf{M}_{ij} + \sum_j \max\left\{\left|\sum_i \mathbf{X}_{ij} - Q_j\right| - \Delta, 0\right\} \text{ s.t. } \forall i, \quad \sum_j \mathbf{X}_{ij} = P_i$$

The use of absolute values in the objective function of AEMC can be removed by introducing additional dummy variables. The above is equivalent to the following linear program:

$$\begin{aligned}
&\min_{\mathbf{X},\mathbf{e}} \sum_j e_j \\
&\text{s.t. } \forall j, \quad e_j \geq \sum_i \mathbf{X}_{ij}\mathbf{M}_{ij} + \left(\sum_i \mathbf{X}_{ij} - Q_j - \Delta\right) \\
&\qquad\quad e_j \geq \sum_i \mathbf{X}_{ij}\mathbf{M}_{ij} + \left(-\sum_i \mathbf{X}_{ij} + Q_j - \Delta\right) \\
&\qquad\quad e_j \geq \sum_i \mathbf{X}_{ij}\mathbf{M}_{ij} \\
&\qquad \forall i, \quad \sum_j \mathbf{X}_{ij} = P_i
\end{aligned}$$

With $\mathbf{M}, \Delta, P, Q$ as the input to an optimization solver, we can obtain the AEMC between two marginals represented by P and Q.

4.2 Model AEMC as Min-cost Flow Problem

The AEMC is equivalent to a min-cost flow problem with some edges requiring both maximum and minimum flow through them. To describe from a high level, the customized min-cost flow problem asks for the minimum cost of flows from source node s to sink node t. Between s and t, there are two sets, each having T nodes. The nodes in the first set corresponds to the bins of P, while the nodes in the second set corresponds to the bins of Q. There are flows in the following scenarios:

1. From s to all the nodes in first set. The flow from s to the i^{th} node in the first set is constrained to P_i, with cost 0.
2. From s to all the nodes in the second set. The flow from s to the j^{th} node in the second set is constrained to be non-negative, with cost 1.
3. From the nodes in the first set and the nodes in the second set. The flow from i^{th} node in the first set to the j^{th} node in the second set is constrained to be non-negative and have cost $\mathbf{M}_{ij}$.
4. From the nodes in the second set to t. There are two flows from the j^{th} node for $j \in [T]$ in the second set to t. One is constrained to be $[Q_j - \Delta, Q_j + \Delta]$ and with cost 0. The other flow between j^{th} node in the second set has non-negative constraints but cost 1.

We detailed the algorithm in Appendix A.

4.3 Optimization of AEMC

Generally speaking, there are $O(T^2)$ variables in the AEMC optimization problem, each of which corresponds to a movement of counts (flow) from bin b to bin b'. These $O(T^2)$ variables manifest as $O(T^2)$ edges in the second approach. However, as introduced in [9], the number of variables for solving earth-mover distance (also applied to AEMC in our case) can be reduced to $O(T)$ if each bin b has a set of "neighbours" $\mathtt{N}(b)$ such that the number of neighbors for each bin is a constant independent from the domain size of any attribute, and all flows between any two bins b and b' can be replaced by a sequence of flows between only neighbors, with the same cost. That is,

$$\forall b, b', \exists b_1 \in \mathtt{N}(b), b_2 \in \mathtt{N}(b_1), \ldots, b' \in \mathtt{N}(b_k),\ \mathbf{M}_{b,b'} = \mathbf{M}_{b,b_1} + \mathbf{M}_{b_1,b_2} + \ldots + \mathbf{M}_{b_k,b'}$$

Several design features of AEMC aim at enabling the above optimization, which we now discuss.

- The L_1 nature of bin distance. The way the bin distance matrix is defined, namely $\mathbf{M}_{bb'} = \sum_a w_a \cdot \mathtt{sd}(b_a, b'_a)$, means that we can limit neighboring bins for any bin only to those that differ in exactly one attribute.
- Ordinal Attributes. The way semantic distance is defined for ordinal attributes means that for each value v, one only needs to consider the value just above v and the value just below v as v's neighbors.
- Non-ordinal Attributes. Recall that we define the semantic distance between two values of a non-ordinal attribute at level t of the generalization hierarchy as $\frac{t-s}{t}$, where s is the level of the lowest common ancestor of the two values. Using this, we can introduce new dummy values for this attribute, with one value for each node at a level $< t$. Then, each value is only neighbors with its parent and descendent. The cost of moving between a parent and child node has cost $\frac{1}{2\times t}$. Adding these new dummy values will create new dummy bins, and constraints need to be added to ensure that they start and end with 0 counts in the flow.

4.4 Scalability/Feasibility

We have found that the linear programming approach of computing `AEMC` scales to thousands of bins in marginals, but have difficulty dealing with tens of thousands or more bins. On the other hand, modeling `AEMC` as a min-cost flow problem and solving the instance using OR-Tools (https://developers.google.com/optimization), we can compute the `AEMC` between the ground truth 3-way marginal of Police Incident Data (totally 580464 cells) and the privatized one in 2 minutes.

We suggest avoid using marginals that have too many cells, both for scalability concerns and for effectiveness in measuring meaningful differences between marginals. In general, if most cells in the the marginal table from the ground truth dataset has very low count, not much meaningful information can be obtained under the constraint of DP.

5 Metric Defense

In Section 3, we have illustrated the application of `MGD` in two applications. Discussions there already addresses issues such as parameter tuning, discrimintaive power, and coverages. Here we add some additional discussions.

5.1 Exploration of Parameter Tuning

Section 3.2 showed that depending on how one wants to use the final data, one can choose different weights for marginals. The generalization hierarchies will also depend on the application domain, and the nature of the attributes. Another parameter that one can choose is Δ. We currently recommend choosing Δ to be a small constant, e.g., some value between 2 and 10. Note that when accumulated small differences have a larger impact, that can be captured by using more coarse-grained marginals that have fewer cells. Due to the time limit, we have only limited experiences using `MGD`. Therefore we do not yet have a deep understanding on the impact of exact choice of Δ, even though we believe that the added flexibility of Δ is useful.

5.2 Discriminative Power and Coverages

One main challenge for designing a metric is that there is no obvious metric for assessing proposed metrics. There are many data analysis tasks one may be interested in, and one synthetic dataset may perform very well on one task, but poorly on another. Thus no single metric can be simultaneously "accurate" for all tasks. Without fixing the set of data analysis tasks, we cannot think of a better metric than measuring accuracy of many marginals. In our opinion, measuring L_1 differences on either all 2-way and 3-way marginals (or a randomly selected subset of them if there are too many) is an excellent starting point.

Our proposed `MGD` metric can be viewed as attempts to improve the basic L_1 difference on marginals, by addressing some limitations and issues we have experienced with the metric, both through our prior research and through our experiences in the NIST Differential Privacy Synthetic Data challenge. One goal is to exploit semantic meanings among the values, both ordinal ones and non-ordinal ones. In the last NIST

competition, we observed that when an attribute has a large domain, almost all marginals involving that attribute tend to be have close to maximal L_1 error. This is because the vast majority of the bins have 0 or close to 0 count. After adding noises, which cells have non-zero count is random. Thus even if a synthesized dataset can capture the few high counts of interest, the L_1 error would still be dominated by those caused by what are essentially white noises. We introduce the approximate nature to enable one to avoid this issue.

We believe that for almost any data analysis tasks, one can find suitable subset of marginals and corresponding weight so that the `MGD` scores are highly correlated with accuracy in the data analysis task. However, the challenge in using `MGD` may be that one has to identify the right configurations for the application, which may not be an easy task.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice hall, 1993.

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 214–223, 2017.

[3] S. Cohen and L. Guibasm. The earth mover's distance under transformation sets. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1076–1083. IEEE, 1999.

[4] K. Grauman and T. Darrell. Fast contour matching using approximate earth mover's distance. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. IEEE, 2004.

[5] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

[6] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

[7] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966, 2015.

[8] N. Li, Z. Zhang, and T. Wang. DPSyn: Differentially private synthetic data publication, 2018.

[9] H. Ling and K. Okada. An efficient earth mover's distance algorithm for robust histogram comparison. *IEEE transactions on pattern analysis and machine intelligence*, 29(5):840–853, 2007.

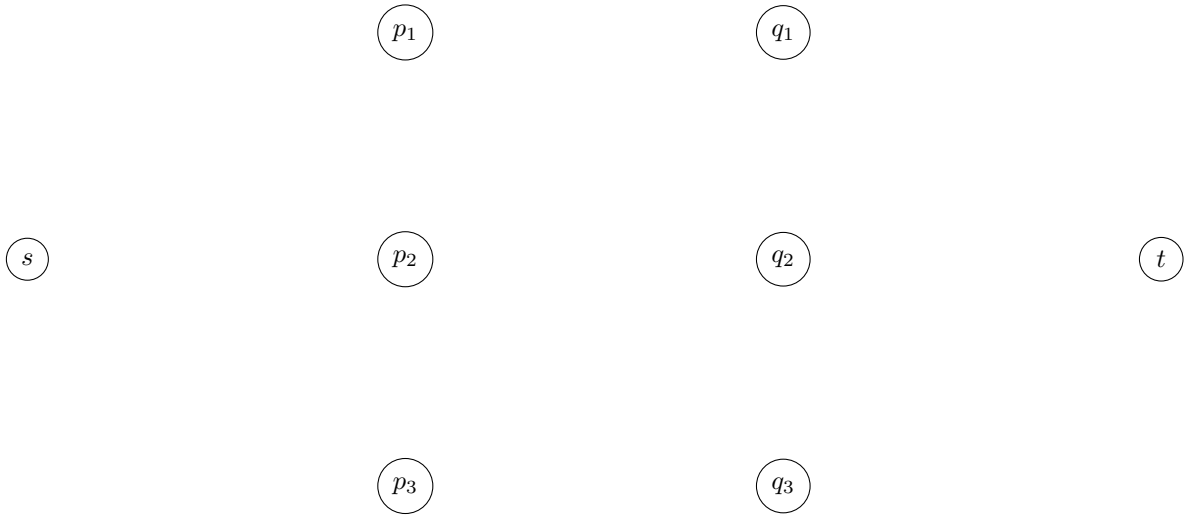
[10] F. Wang and L. J. Guibas. Supervised earth mover's distance learning and its computer vision applications. In *European Conference on Computer Vision*, pages 442–455. Springer, 2012.

A Using Min-cost Flow with Lower Bounds to Compute `AEMC`

We first show how to reduce the problem of computing `AEMC` to Min-cost Flow with Lower Bounds, and then show how this can be reduced to Min-cost circulation problem, for which we use the standard algorithm implemented in software packages such as OR-Tools to solve.

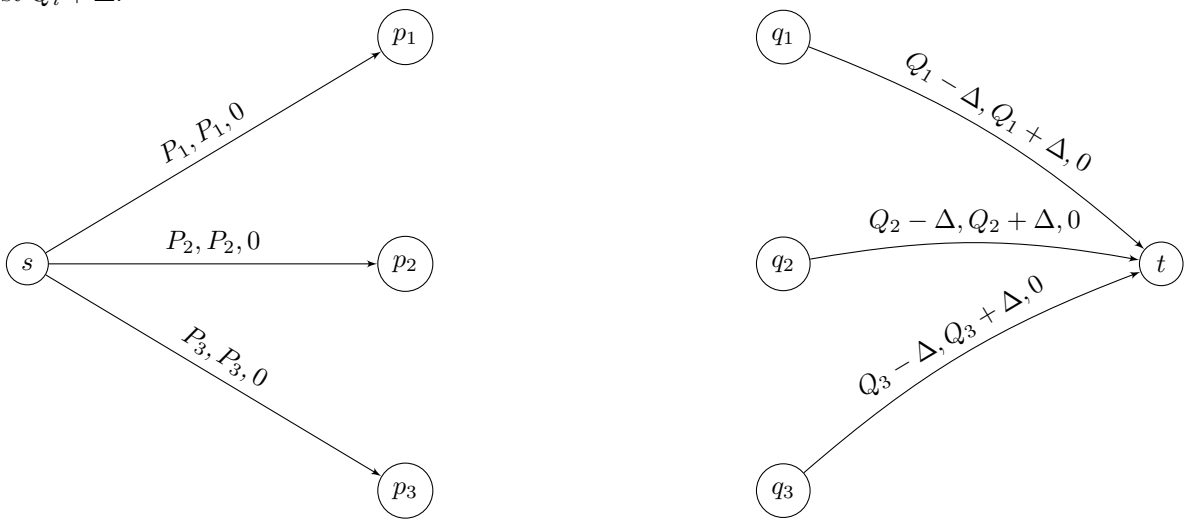
A.1 Reducing Computing `AEMC` to Min-cost Flow with Lower Bounds

For an `AEMC` instance, we create a graph with 4 layers, where the first layer contains only the source node s, the second layer contains n nodes representing cells of the marginal P computed from the privatized data, the third layer contains n nodes representing cells of the marginal Q computed from the ground truth, and the fourth layer contains only the sink node. The figure below shows an example of $n = 3$ bins.

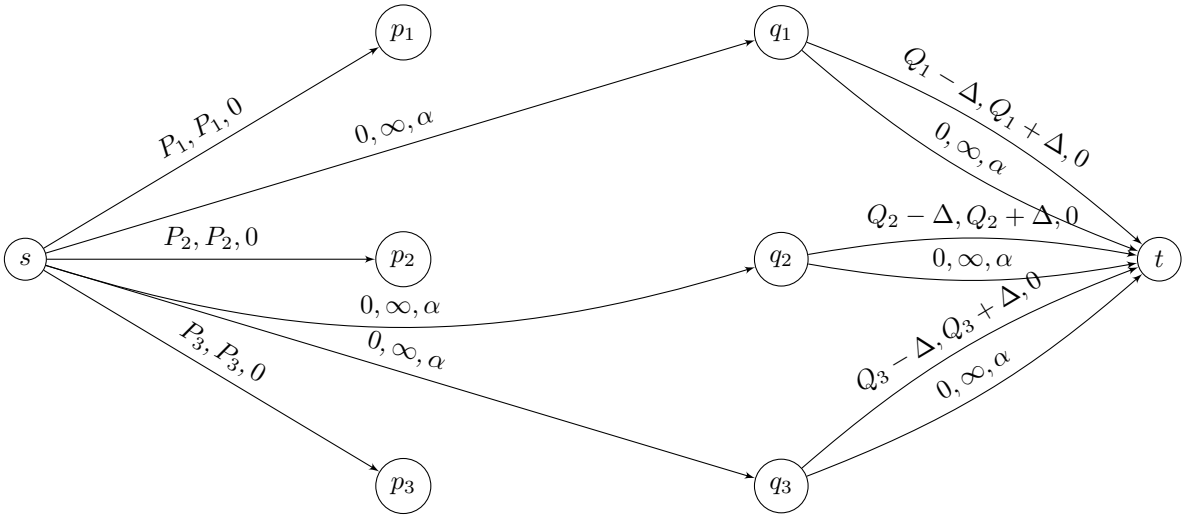


Let us first restrict our problem: suppose we can only move so that each bin q_i lies within the approximated range of Q_i, i.e. $Q_i - \Delta \leq q_i \leq Q_i + \Delta$.

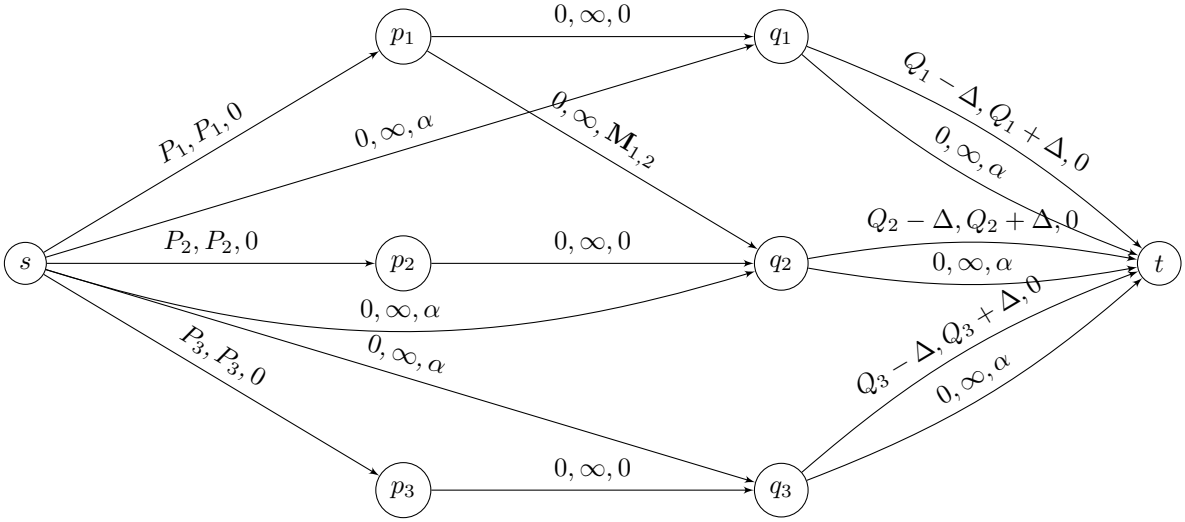
We will add edges (s, p_i) with lower bound $b(s, p_i) = P_i$, capacity $c(s, p_i) = P_i$, and unit cost $w(s, p_i) = 0$ for all $1 \leq i \leq n$. Intuitively, think of these edges as "whether we want it or not, each p_i must have P_i value at first". Similarly, we will add edges (q_i, t) with lower bound $b(q_i, t) = Q_i - \Delta$, capacity $c(q_i, t) = Q_i + \Delta$, and unit cost $w(q_i, t) = 0$, with the intuition that each resulting bin must contribute at least $Q_i - \Delta$ and at most $Q_i + \Delta$.



Let us relax the restriction and see how we can allow each resulting bin to exceed the predefined range with a unit cost of α. (In definition of `AEMC`, the value of α is implicitly set to 1. Here we describe the general case, when α can be set to other values). We can do it as follows: for each node q_i, add an edge (s, q_i) with lower bound $b = 0$, capacity $c = \infty$, and unit cost $w = \alpha$; additionally, add an edge (q_i, t) with lower bound $b = 0$, capacity $c = \infty$, and unit cost $w = \alpha$. Intuitively, the edge (s, q_i) helps q_i to achieve the predefined range $[Q_i - \Delta, Q_i + \Delta]$ with a unit cost of α; similarly, the edge (q_i, t) helps q_i to "relieve" some flow to allow it to reach the predefined range $[Q_i - \Delta, Q_i + \Delta]$ with a unit cost of α.

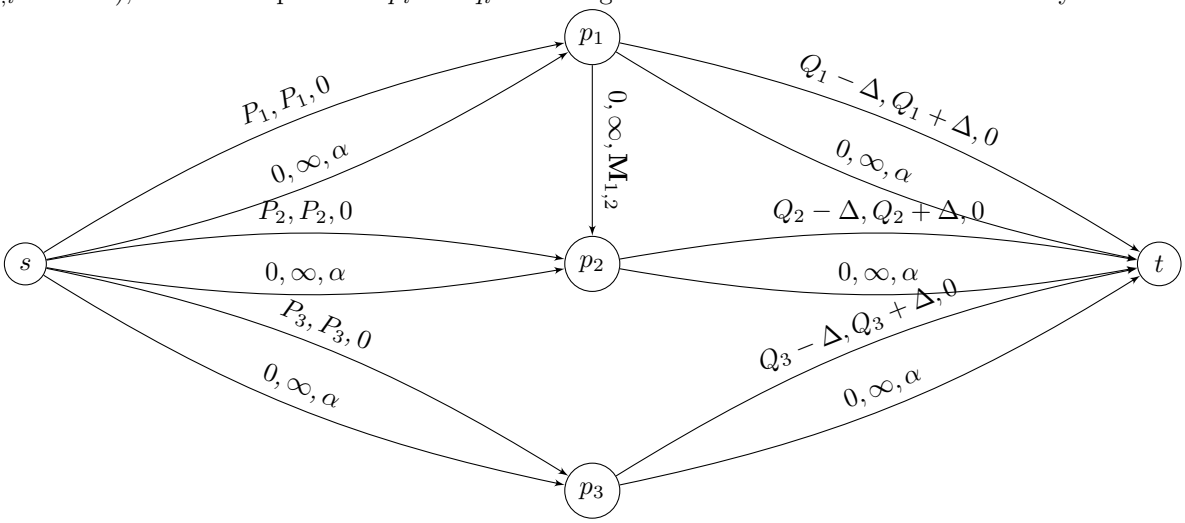


Finally, for every pair of bins (i, j), add an edge (p_i, q_j) with lower bound $b = 0$, capacity $c = \infty$, and cost $w = \mathbf{M}_{i,j}$. For example, suppose $\mathbf{M}_{i,i} = 0 \quad \forall i$ and only $\mathbf{M}_{1,2}$ is defined, then the resulting graph is:



Now we run the min cost flow with lower bounds solver on this graph to get the `AEMC`.

Moreover, since the number of cells in P and Q are the same, and the cost of not moving bins is 0 (i.e. $\mathbf{M}_{i,i} = 0 \quad \forall i$), we can compress the p_i and q_i nodes together to reduce the number of nodes by half.



A.2 Solving Min-cost Flow with Lower Bounds

The *flow with lower bounds problem* is similar to the classic flow problem but with an additional constraint that there is a lower bound on each edge. More formally, given a graph $G = (V, E)$ with a source s and a sink t, where each edge (u, v) has a capacity $c(u, v)$ and a lower bound $b(u, v)$, the goal is to find a flow f from s to t such that

$$\begin{aligned} & b(u,v) \le f(u,v) \le c(u,v) && \forall (u,v) \in E \\ & \sum_{(x,u)\in E} f(x,u) - \sum_{(u,y)\in E} f(u,y) = 0 && \forall u \in V \setminus \{s,t\} \end{aligned}$$

Such a flow is called a valid flow on G. The *min cost flow with lower bounds* problem gives each edge (u, v) a cost $w(u, v)$, and we need to find a valid flow that minimizes

$$\sum_{(u,v)\in E} f(u,v) \cdot w(u,v)$$

We solve the min cost flow with lower bounds by reducing it to the min circulation problem. The difference is that there are no source and sink in the graph. Any min flow with lower bounds problem can be reduced to an instance of the circulation problem by adding an edge (t, s) with lower bound $b(t, s) = 0$, capacity $c(t, s) = \infty$, and unit cost $w(t, s) = 0$.

Let us define another problem called *min circulation with node demands*: You are given a graph $G = (V, E)$, where each edge (u, v) has a capacity $c(u, v)$ and a unit cost $w(u, v)$, and each node has a demand $d(u)$. We need to assign a flow to each edge (u, v) such that each node's demand is satisfied, and the overall circulation is minimal. Formally, the problem can be described as the following linear program:

$$\begin{aligned} \min \ & \sum_{(u,v)\in E} f(u,v) \cdot w(u,v) \\ \text{s.t. } & 0 \le f(u,v) \le c(u,v) && \forall (u,v) \in E \\ & \sum_{(x,u)\in E} f(x,u) - \sum_{(u,y)\in E} f(u,y) = d(u) && \forall u \in V \end{aligned}$$

We can transform an instance of min circulation with lower bounds to an instance of min circulation with node demands using the following reduction:

- First, initialize the demands for all nodes to be zero, i.e. $d(u) = 0$.
- For each edge (u, v), add $b(u, v) \cdot w(u, v)$ to the answer, add $b(u, v)$ to $d(u)$, subtract $b(u, v)$ from $d(v)$, and reform this edge to have no lower bound and with a capacity of $c(u, v) - b(u, v)$.

Intuitively, since each edge (u, v) must have a lower bound of $b(u, v)$, we simply force this lower bound (thus we need to add $b(u, v) \cdot w(u, v)$ to the answer), and the remaining flexible capacity for this edge is only $c(u, v) - b(u, v)$. To enforce that there must be such flow going through that edge, we reserve a supply of $b(u, v)$ on the node u, therefore u must "eat" $b(u, v)$ flow from other edges, hence adding $b(u, v)$ to $d(u)$; similarly, v must "burp" $b(u, v)$ to other edges, hence subtracting $b(u, v)$ from $d(v)$.

The min circulation with node demands problem is a well known problem[1], with the state of the art algorithm having a time complexity of $O(mn \log W \log \log C)$, where $n = |V|$, $m = |E|$, $W = \max w(u, v)$, and $C = \max c(u, v)$. We are currently using OR-Tools solver, which has a time complexity of $O(n^2 m \log(nC))$.

Dataset	Inv Pie	RQ	AEMC						
			M	N	I	M+I	M+N	N+I	3-way
mediocre	0.5689	1.0666	0.8669	0.8666	0.8667	0.8641	0.8624	0.8117	0.2667
very-poor	0.96	9.0631	7.6956	7.6952	7.6954	7.6927	7.691	7.6401	7.1258
dpsyn-0.1	0.8236	0.0451	0.026	0.0598	0.1026	0.0748	0.0755	0.1169	0.1013
dpsyn-0.25	0.6751	0.0175	0.0086	0.0214	0.0413	0.0286	0.0266	0.0616	0.0556
dpsyn-0.5	0.5801	0.0103	0.0048	0.009	0.018	0.0131	0.0125	0.0332	0.0361
dpsyn-1.0	0.5222	0.007	0.0022	0.0044	0.0071	0.0053	0.0056	0.0157	0.0305
dpsyn-2.0	0.5039	0.0074	0.0011	0.002	0.0029	0.0025	0.0026	0.0061	0.0298
dpsyn-4.0	0.5113	0.0169	0.0142	0.0138	0.014	0.0117	0.0097	0.0016	0.0296
dpsyn-10.0	0.5017	0.007	0.0019	0.0028	0.0034	0.0025	0.0025	0.0053	0.03
lap-0.25	0.9853	19.3996	15.6775	15.6772	15.6773	15.6747	15.673	15.6216	15.1237
lap-0.5	0.9619	8.8361	7.6918	7.6915	7.6916	7.689	7.6873	7.6366	7.1222
lap-1.0	0.9072	4.3788	3.7608	3.7604	3.7606	3.7579	3.7562	3.706	3.1763
lap-2.0	0.8278	2.115	1.8199	1.8195	1.8196	1.817	1.8153	1.7651	1.224
lap-4.0	0.5701	1.0375	0.8661	0.8658	0.8659	0.8634	0.8616	0.8115	0.2661
lap-10.0	0.3935	0.369	0.3178	0.3174	0.3176	0.3151	0.3132	0.2622	0.0481
sample-0.01	0.8883	0.0407	0.019	0.046	0.0252	0.0461	0.0827	0.1428	0.1956
sample-0.05	0.821	0.0152	0.0083	0.0199	0.0096	0.0197	0.0344	0.0669	0.111
sample-0.1	0.7337	0.0122	0.0059	0.0145	0.0075	0.0142	0.0244	0.0443	0.074
sample-0.25	0.5494	0.0072	0.0028	0.008	0.0044	0.0077	0.0135	0.0222	0.0315
sample-0.5	0.3834	0.0042	0.0019	0.0045	0.0022	0.0043	0.0073	0.0103	0.0106

Table 1: Compare inversed Pie-chart metric (Inv Pie), range query (RQ) and `AEMC` on different marginals, M: "months", N: "neighborhood", I: "incident type". $\Delta = 2$ for `AEMC`.

B Detailed Results of Experiments

Table 1 shows different scores of synthetic datasets. The "mediocre" and "very-poor" datasets are from Example Temporal Map Data provided by the competition. The synthetic datasets, "dpsyn-X", are generated by algorithm in [8] with privacy budget $\epsilon = X$ and truncation which limits the per-user contribution to 2. The "lap-X" datasets are produced by basic Laplace mechanism in differential privacy with privacy budget $\epsilon = X$ and sensitivity is set to 20. The "sample-X" is for randomly sampling X portion of incidents from the incident datasets.

The Inv Pie score is computed by $(1 - S/3336)$, where S is the pie-chart score of the dataset. The range query (RQ) is the relative error of 300 randomly generated range queries, each of which queries on the count in 30% of month, neighborhood and incident types.