Robust Set Reconciliation

Di Chen[1] Christian Konrad[2*] Ke Yi[1†] Wei Yu[3] Qin Zhang[4]

[1]Hong Kong University of Science and Technology, Hong Kong, China
[2]Reykjavik University, Reykjavik, Iceland
[3]Aarhus University, Aarhus, Denmark
[4]Indiana University Bloomington, Bloomington, IN, USA

ABSTRACT

Set reconciliation is a fundamental problem in distributed databases, where two parties each holding a set of elements wish to find their difference, so as to establish data consistency. Efficient algorithms exist for this problem with communication cost proportional only to the difference of the two sets, as opposed to the cardinality of the sets themselves. However, all existing work on set reconciliation considers two elements to be the same only if they are exactly equal. We observe that, in many applications, the elements correspond to objects on which a distance function can be defined, e.g., points in the Euclidean space, and close points often actually represent the same object. During the reconciliation, the algorithm should only find the truly different elements in the two sets while tolerating small perturbations. In this paper, we propose the *robust set reconciliation* problem, and take a principled approach to address this issue via the *earth mover's distance*. We have developed a communication and time-efficient algorithm with provable guarantees on the quality of the reconciliation. This is then complemented with an essentially matching lower bound showing the optimality of the algorithm. Our experimental results on both synthetic and real data sets have demonstrated that our algorithm also performs very well in practice.

1. INTRODUCTION

Data synchronization is a fundamental problem in distributed databases, where two or more parties wish to establish consistency for the data sets they hold. This problem is now becoming more important as big data systems and cloud data services invariably rely on replication to achieve high availability and fault tolerance. The classical approach to data synchronization is *wholesale transfer*, in which one party sends all its data to the other. This method incurs linear communication cost and is clearly not scalable to the large data sets nowadays. Thus, one seeks methods with communication costs proportional only to the difference of the two data sets.

Depending on the data type we are dealing with, we arrive at different data synchronization problems. In this paper, we are interested in sets (or bags), and the corresponding data synchronization problem is also known in the literature as the *set reconciliation problem*. Specifically, let the two parties, Alice and Bob, each hold a set S_A and S_B, respectively. The goal is to compute the set difference $S_A \oplus S_B = (S_A - S_B) \cup (S_B - S_A)$ using minimum communication. The set reconciliation problem has been well studied in various research communities [7, 18, 9], and can be solved with communication cost $O(|S_A \oplus S_B|)$. As a concrete application, suppose Alice and Bob each have a large set of images on their phones, and wish to synchronize their collection. Using the techniques developed, they can do so with communication cost that is proportional to their difference, namely, the number of images that one of them has, but not both.

1.1 Motivation

The underlying assumption in the example above is that each image can be identified uniquely. This, however, is only true when the image is always stored as exactly the same file. Dozens of image file formats exist; even using the same file format, different codecs handle things in numerous ways that are slightly different, not to mention the many possible values for the quality parameters. To detect the same or similar images, extensive efforts in computer vision and multimedia research have led to effective feature extraction methods that map an image to a point in some feature space, together with suitable distance functions defined to measure the similarity. However, this is incompatible with the existing set reconciliation methods, which demand strict equality when comparing elements in the two sets.

The above is merely a motivating example of the problem we are facing. More broadly, this problem exists whenever we try to reconcile two numerical data sets where small errors should be tolerated. In addition to multimedia data, small errors could have been introduced for a variety of reasons: (1) They can be introduced during initial data synchronization, if the transmission channel is noisy, or one party may just decide to do a lossy compression of the data before sharing its data with the others, as the precise values do not matter much anyway. (2) Different parties holding the same data set may perform different transformations on the data but that are mathematically equivalent. However, due to rounding errors in floating-point computations, the

*Supported by the Icelandic Research Fund grant-of-excellence no. 120032011.
†Supported by HKRGC under grant GRF-621413.

results are rarely exactly the same (`sqrt(1.1) * sqrt(1.1)` is almost never exactly equal to `1.1`). Note that small rounding errors could potentially accumulate to the point of being disastrous, if one is not careful about the computation procedure, which is actually a main topic of study in the field of numerical analysis [11]. In this case we would want the reconciliation algorithm to detect these large deviations. (3) Finally, small errors are sometimes intentional due to privacy concerns. In privacy-preserving data publishing, adding controlled noise is a basic technique used by many algorithms [6]. Thus, the publisher and the subscriber will have two slightly different values for the same data record.

These observations have led us to study a *robust* version of the set reconciliation problem, where two elements should be considered as the same if their difference is small, while the algorithm will only find the "true differences" between the two sets S_A and S_B. Correspondingly, the communication cost of the algorithm should also be proportional to the number of "true differences". The existing algorithms for (exact) set reconciliation, when facing data sets with many small errors, would either fail or try to identify all differences, large or small, while incurring a huge communication overhead.

1.2 Problem Definition

For concreteness, let S_A and S_B be two sets each with n points drawn from the d-dimensional Euclidean space[1]. How should we define the "true differences" between S_A and S_B? One obvious way is to set a threshold δ, and declare two points $x \in S_A, y \in S_B$ to be truly different if $\|x - y\| > \delta$. However, setting an appropriate value of δ is a bit tricky and arbitrary, as noise usually follows some continuous distribution (e.g., Gaussian). The other issue is that since S_A and S_B are sets, there is no fixed mapping between their points, an $x \in S_A$ that is "truly different" from a $y \in S_B$ may just be considered to have a small error when compared with another $y' \in S_B$. Therefore, we propose the following more principled definition based on the *earth mover's distance* (EMD), which is a popular similarity measure between two point sets [1].

Definition 1 (EMD) *The earth mover's distance (EMD) between two point sets $S_A = \{x_1, \dots, x_n\}$ and $S_B = \{y_1, \dots, y_n\}$ is the cost of the min-cost perfect matching between S_A and S_B, that is,*

$$\mathrm{EMD}(S_A, S_B) = \min_{\text{bijection } \pi:[n]\to[n]} \sum_{1\le i\le n} \left\|x_i - y_{\pi(i)}\right\|_2 .$$

The ℓ_2 norm $\|\cdot\|_2$ can be replaced by other norms, which will result in different versions of the EMD. Computing the EMD in one dimension is quite easy: We just sort S_A and S_B respectively, and match the corresponding points in order. However, in two or more dimensions, finding the min-cost perfect matching requires $O(n^3)$ time using the Hungarian method [16].

We note that the EMD can be more generally applied to other types of data, such as probability distributions and vectors [2]. Here we adopt its form when applied to measuring the distance between two point sets. In this case, it is also known as the *transportation distance* or *bipartite matching distance*. This distance measure is more appropriate for our problem than other measures for point sets such as the Hausdorff distance, since we are looking for a one-to-one matching between the two point sets, whereas the latter allows many-to-one matchings.

Definition 2 (Robust set reconciliation) *Suppose Alice has a set of points S_A and Bob has S_B. Given an upper bound on the communication cost, the goal of the* robust set reconciliation *problem is for Bob to find two sets $S_B^- \subset S_B$ and $S_B^+ \subset S_A$ of equal size, such that $\mathrm{EMD}(S_A, S'_B = (S_B - S_B^-) \cup S_B^+)$ is minimized.*

Note that this definition only requires Bob to learn the difference (S_B^- and S_B^+), but the other direction is symmetric. In the problem definition we do not have any hard limit on the size of S_B^- and S_B^+, but obviously, given a communication budget, there is limited information that Bob can learn, so they cannot be too large.

The above definition using the EMD avoids setting arbitrary parameters (the only parameter is the communication budget), and naturally puts this problem under the general problem of data synchronization, where the goal is to reduce some distance measure between the two data sets, ideally to zero.

A technical assumption we will make in this paper is that the coordinates of the points take integer values in some range $[0, \Delta - 1]$. We feel that this is not a restriction, since floating-point numbers have limited precision and can be appropriately scaled and converted into integers. For example, if the application has a known precision requirement, say $\delta = 0.0001$, then a floating-point number coordinate x can be converted to an integer $\lfloor x/\delta \rfloor$.

1.3 Our Contributions

For any $0 \le k \le n$, define $\mathrm{EMD}_k(S_A, S_B)$ to be the cost of the min-cost perfect matching between S_A and S'_B, after at most k points in S_B have been relocated. More precisely,

$$\mathrm{EMD}_k(S_A, S_B) = \min_{|S_B^-|=|S_B^+|\le k} \mathrm{EMD}(S_A, S'_B = (S_B - S_B^-) \cup S_B^+).$$

It should be clear that EMD_k monotonically decreases as k. Moreover, if k is larger than the number of true differences, then EMD_k will only include the cost over the small errors.

We have made the following contributions to the robust set reconciliation problem.

1. We have designed an algorithm that, for any $0 < k < n$, has a communication cost of $O(kd \log(n\Delta^d) \log \Delta)$ bits and can find an $S'_B = (S_B - S_B^-) \cup S_B^+$ such that $\mathrm{EMD}(S_A, S'_B) \le O(d) \cdot \mathrm{EMD}_k(S_A, S_B)$ with high probability. This guarantee holds for the EMD defined by any ℓ_p norm for $p \ge 1$ and holds in any dimensions. The algorithm only needs Alice to send one single message to Bob. Alice spends near-linear time $O(dn \log \Delta)$ to construct the message (note that the input size is $O(dn)$), while Bob also spends $O(dn \log \Delta)$ time to decode the message and finds S_B^- and S_B^+. This result in some sense can be considered as an approximation algorithm with approximation ratio of $O(d)$, as compared with the optimal solution with the restriction $|S_B^-| = |S_B^+| = k$, although we do not impose this restriction to our algorithm. Instead, we put a limit on

[1]We actually use the bag semantics and allow duplicated points in S_A and S_B, but to be consistent with the literature we still use the term "set reconciliation".

the communication cost, which is more meaningful as this is the resource we actually care.

Note that if there are only k true differences and no small errors between the two data sets, then we have $\mathrm{EMD}_k(S_A, S_B) = 0$ and our algorithm will reconcile the two data sets such that $\mathrm{EMD}(S_A, S'_B) = 0$, i.e., S_A and S'_B are identical. This means that our algorithm strictly generalizes the standard set reconciliation problem.

2. We have complemented our algorithm with a communication lower bound of $\Omega(k \log(\Delta^d/k) \log \Delta)$ bits, for any randomized one-way communication algorithm that solves the robust set reconciliation problem that achieves a quality guarantee of $\mathrm{EMD}(S_A, S'_B) = O(1) \cdot \mathrm{EMD}_k(S_A, S_B)$. Note that for typical settings where $d = O(1), n = \Delta^{O(1)}$, and $k = O(\Delta^{d-\epsilon})$ for any small constant ϵ, our algorithm exactly matches this lower bound, showing the optimality of the algorithm, even on the bits level.

3. We have conducted experiments on real-world data sets, both one-dimensional and multi-dimensional, to test the efficiency and reconciliation quality of our algorithm. The experimental results demonstrate that our algorithm's running time scales linearly in both data size and dimensionality, and achieves a reconciliation quality that is order-of-magnitude better than a baseline method using lossy compression. Finally, we apply our algorithm to the task of image reconciliation, and have achieved very satisfactory results.

1.4 Related Work

The problem studied in this paper falls into the general problem of *data synchronization*, which is the process of establishing consistency among data from a source (Alice) to a target data storage (Bob) or vice versa. This fundamental problem has been extensively studied by various research communities in databases, networking, and information theory. As accurate and timely synchronization is expensive to achieve, attention has primarily been focused on devising more relaxed synchronization models. One dimension is to relax the time constraint, which resulted in various consistency models on the fundamental trade-offs between consistency, performance, and availability [15, 21, 10], as mandated by the CAP theorem [3]. The other dimension is to relax accuracy, i.e., reducing the distance between the two data sets as much as possible instead of making the two data sets exactly identical. Depending on the data type and distance measure, we arrive at different instances of the problem. For *ordered data* (i.e., sequence of elements) with Hamming distance as the distance measure, the problem can be solved by *error correction code* [17]. The problem gets more challenging when using the edit distance [5], which is more useful when synchronizing two text files. For unordered data (i.e., sets) with the standard set-difference measure (or equivalently the *Jaccard similarity*), the problem is exactly the *set reconciliation* problem [7, 18, 9], which was mentioned at the beginning of the paper. For our problem, the elements in the sets are points in the Euclidean space and the distance measure is the EMD.

The robustness issue has been raised for synchronizing ordered data. Candes and Randall [4] studied the following problem. Alice holds a vector of reals $\mathbf{x} = (x_1, \dots, x_n)$ and Bob has $\mathbf{y} = (y_1, \dots, y_n)$. In addition to a small number of true differences where $|x_i - y_i|$ is large, every entry has some noise added $y_i = x_i + z$. They designed an algorithm that corrects $\mathbf{y}$ into some $\mathbf{y}'$ such that $\|\mathbf{x} - \mathbf{y}'\|_2$ is roughly $n\sigma^2$, where σ is the standard deviation of the noise. In some sense, the algorithm has identified all the true differences, leaving only the noise in the recovered vector $\mathbf{y}'$. This result is actually very similar to our guarantee using EMD_k: when k is the number of true differences, EMD_k is essentially the noise. However, our problem is more challenging as the EMD depends on the min-cost perfect matching, while for ordered data, the matching is trivial (i.e., x_i is always matched with y_i for all $1 \le i \le n$). Furthermore, [4] only studied the problem where each x_i is a real number (i.e., a 1D point), while our algorithm works for the x_i's being multi-dimensional points.

The EMD has been popularly used as a similarity measure between point sets, but computing the EMD is no simple matter. If there are no additional structures, the best known algorithm is the Hungarian algorithm with running time $O(n^3)$ [16]. In the d-dimensional Euclidean space for constant d, there are approximation algorithms with near-linear running time [12, 19], but it is not clear if they are practical. Computing the EMD between two data sets held by two parties is even more difficult. There is no polylogarithmic communication algorithm that achieves a constant approximation in the one-way communication model. The best known algorithms have communication costs polynomial in Δ [1, 20]. Interestingly, although our algorithm gives a quality guarantee of $\mathrm{EMD}(S_A, S'_B) = O(d) \cdot \mathrm{EMD}_k(S_A, S_B)$, it does not actually compute $\mathrm{EMD}(S_A, S'_B)$ or $\mathrm{EMD}_k(S_A, S_B)$ (which is difficult).

Finally, there is a considerable amount of work on dealing with noise in data, particularly on data integration [14], similarity search [2], and joins [13]. But the problem of dealing with noise under the context of set reconciliation has not been studied before.

2. THE ALGORITHM

2.1 Invertible Bloom Lookup Table (IBLT)

We will make use of an elegant data structure called the *invertible Bloom lookup table (IBLT)* [8], which has been used to solve the (exact) set reconciliation problem [7]. Below we give a brief but self-contained description of the IBLT. Please refer to [8, 7] for more detailed discussions.

The IBLT uses a table T of m cells, and a set of r random hash functions $h_1, \dots, h_r$ to store a set of keys. Similar to a Bloom filter, a key x is inserted into the cells designated by the r hash functions: $T[h_1(x)], \dots, T[h_r(x)]$. To use the IBLT for the set reconciliation problem, we use another hash function $f(x)$ that computes a fingerprint for each key x. Each cell in T contains two fields: a `keySum`, which is the XOR of all keys that have been inserted to this cell, and a `fpSum`, which is the XOR of all the fingerprints of the keys that have been inserted to this cell[2]. To solve the set reconciliation problem, Alice simply inserts all her keys into an IBLT, and passes it to Bob, who then inserts all of his keys. Note that due to the use of XOR, inserting

[2]The original IBLT has a third field, `count`, but this is not needed for our problem.

the same key twice leaves no effects on the table, thus only the keys in $S_A \oplus S_B$ remain in the IBLT. The value of r is a constant [8, 7], so it takes constant time to insert a key into the IBLT (assuming a hash function can be evaluated in constant time).

Example: Figure 1 shows an example how the IBLT is constructed. The two sets of keys S_A and S_B each have two keys that are different from the other set (the last two keys). The fingerprints have been computed using the fingerprint function $f(x) = \hat{x}(2\check{x}+1)+\check{x}^2$ (truncated to the last 8 bits), where $\hat{x}$ and $\check{x}$ respectively are the first and last 4 bits of x. Here each key is inserted to $r = 2$ cells (dictated by 2 hash functions). Due to the use of XOR, all the keys common in S_A and S_B leave no effects on the IBLT (the dashed arrows), while only the 4 keys in $S_A \oplus S_B$ remain (the solid arrows). □

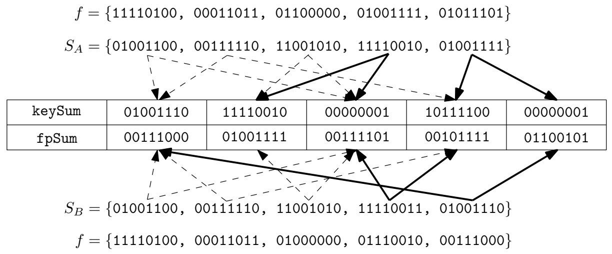
To retrieve the keys remaining in the IBLT, we first scan the table to find cells in which $f(\texttt{keySum}) = \texttt{fpSum}$. This indicates that the cell has only one key which is `keySum` (as long as f is an appropriately chosen, nonlinear fingerprint function). We output this key, and then insert this key again to the r cells that contain it to eliminate its effects on the IBLT. This in turn may make those cells meet the condition $f(\texttt{keySum}) = \texttt{fpSum}$, and we can recover more keys. The whole process ends when no cell satisfies this condition. When this happens, if all cells are empty, then we have succeeded in recovering all keys in the IBLT; otherwise we only have retrieved a partial list. The analysis [8, 7] shows that as long as $m \geq \gamma \cdot |S_A \oplus S_B|$ where γ is a fairly small constant[3], the retrieval process will succeed with high probability $1 - 1/m^{O(1)}$. The retrieval algorithm can be easily implemented to run in $O(m)$ time. Note that although the IBLT itself does not distinguish keys in $S_A - S_B$ from those in $S_B - S_A$, the person who retrieves $S_A \oplus S_B$ from the IBLT (i.e., Bob) can easily tell them apart.

Example (continued): Suppose we now want to retrieve the 4 keys stored in the IBLT in Figure 1. We first check each cell if $f(\texttt{keySum}) = \texttt{fpSum}$. In this example only the first two cells meet this condition, while the last 3 cells have collisions. Thus we first retrieve the two keys `01001110` and `11110010` from the first two cells. Then we insert these two keys back to the IBLT. Due to the use of XOR, this has the effect of removing their effects, leaving the IBLT to the state shown in Figure 2, where the solid arrows indicate the remaining two keys. They still collide in cell number 4, but cell number 3 and 5 now have only one key left, which can be checked by the condition $f(\texttt{keySum}) = \texttt{fpSum}$. Then, we retrieve the keys `11110011` and `01001111` from these two cells. In this example, the retrieval process stops here, but in general could continue iteratively until no more keys can be retrieved. By crosschecking the 4 retrieved keys with his own set of keys S_B, Bob can tell that `11110011` and `01001110` belong to $S_B - S_A$ while `11110010` and `01001111` belong to $S_A - S_B$. □

2.2 The Encoding Algorithm

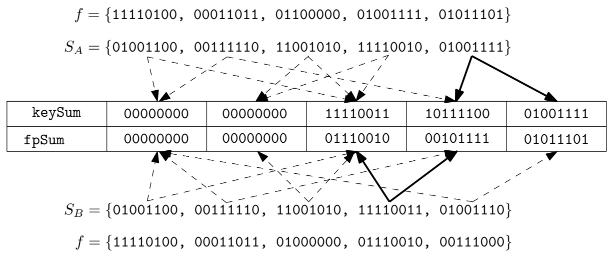
Suppose Alice has a set of n points $S_A = \{x_1, \ldots, x_n\}$ drawn from the d-dimensional integer grid $[\Delta]^d$. Without

[3]This constant affects the exponent in the $1 - 1/m^{O(1)}$ success probability. It suffices to use $r = 5$ hash functions with $\gamma = 1.425$.



f = {11110100, 00011011, 01100000, 01001111, 01011101}
S_A = {01001100, 00111110, 11001010, 11110010, 01001111}
keySum
01001110
11110010
00000001
10111100
00000001
fpSum
00111000
01001111
00111101
00101111
01100101
S_B = {01001100, 00111110, 11001010, 11110011, 01001110}
f = {11110100, 00011011, 01000000, 01110010, 00111000}

Figure 1: An example on how the IBLT is constructed.



f = {11110100, 00011011, 01100000, 01001111, 01011101}
S_A = {01001100, 00111110, 11001010, 11110010, 01001111}
keySum
00000000
00000000
11110011
10111100
01001111
fpSum
00000000
00000000
01110010
00101111
01011101
S_B = {01001100, 00111110, 11001010, 11110011, 01001110}
f = {11110100, 00011011, 01000000, 01110010, 00111000}

Figure 2: An example on retrieving keys from the IBLT.

loss of generality we assume that $\Delta = 2^L$ for some integer L. Let the coordinates of x_i be $x_i = (x_{i1}, \ldots, x_{id})$. Below we describe how she constructs the message to be sent to Bob for reconciliation. The algorithm consists of three simple steps: (1) perform a random shift of the whole data set; (2) build a quadtree; and (3) insert the quadtree nodes to an IBLT. The detailed steps are as follows.

1. *Random shift.* Alice picks a vector $\xi = (\xi_1, \ldots \xi_d)$ uniformly at random from $[\Delta]^d$ and add it to all points in S_A, wrapped around Δ, i.e., we set $x_{ij} \leftarrow (x_{ij} + \xi_j) \bmod \Delta$ for all $1 \leq i \leq n, 1 \leq j \leq d$.

2. *Build a quadtree.* Alice builds a quadtree on the whole grid $[\Delta]^d$. Recall that a quadtree is a hierarchical decomposition of space. The root corresponds to the whole grid $[\Delta]^d$. It has 2^d children which correspond to the 2^d quadrants of their parents. The subdivision then continues recursively until we reach the single cells of the grid, resulting in $L + 1$ levels. For each node in the quadtree, Alice records the number of points in S_A that fall inside its cell, and removes all empty nodes. In fact, the empty nodes do not have to be created at all if we build the quadtree bottom up. The whole construction can be implemented to run in $O(nd \log \Delta)$ time, by storing all nonempty nodes in a hash table.

3. *Insert into IBLT.* For each level of the quadtree, Alice inserts all the (nonempty) nodes into an IBLT. More precisely, if a node at level ℓ (the leaves are said to be on level 0) corresponds to cell $[q_1, q_1 + 2^\ell - 1] \times [q_2, q_2 + 2^\ell - 1] \times \cdots \times [q_d, q_d + 2^\ell - 1]$ with count c, then she inserts the combined key $(q_1, q_2, \ldots, q_d, c)$ into the IBLT. The size of the IBLT is $\gamma \cdot \alpha k$, where α is a parameter controlling the reconciliation quality (whose value will be determined later in the analysis), and k is chosen such that the total message size (i.e., $L + 1$ IBLTs) meets the communication budget. As it takes

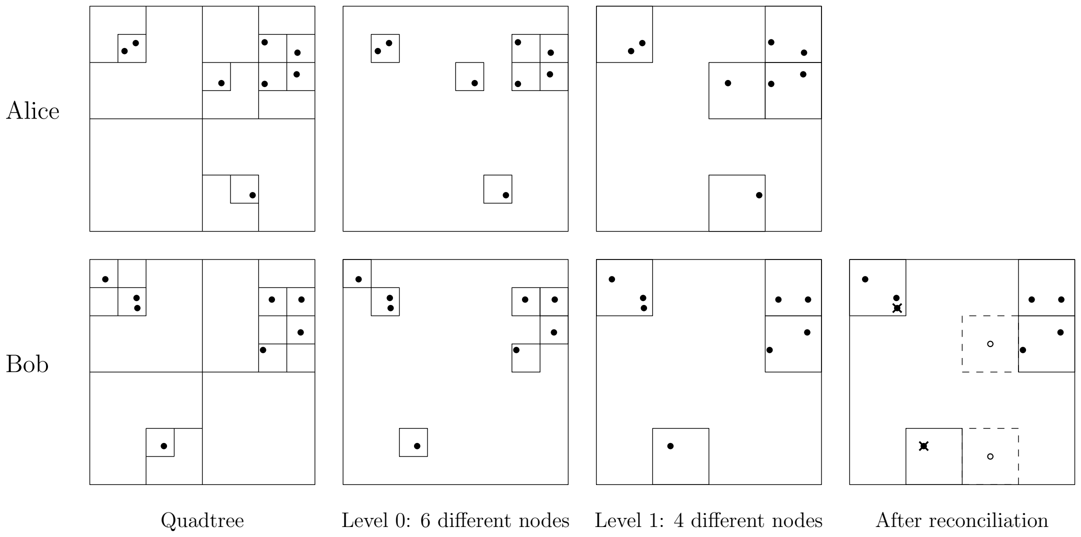


Figure 3: An example illustrating the encoding and decoding algorithm.

$O(d)$ time to evaluate a hash function on a combined key, the total time to build the IBLTs is $O(dn \log \Delta)$.

In the end, Alice sends the $L + 1$ IBLTs, together with ξ, to Bob. It should be quite clear that the total message size is $O(\alpha k \log(n\Delta^d) \log \Delta)$ bits, as each combined key requires $O(\log(n\Delta^d))$ bits. Of course, Alice should also share the hash functions and the fingerprint function she uses to construct the IBLTs. These are usually constant-degree polynomials, so their descriptions take negligible space.

It is easy to see that the encoding algorithm runs in time $O(dn \log \Delta)$.

Example: Figure 3 shows a complete 2D example with $n = 8$ points on a $[8] \times [8]$ grid (except that we omit the random shift step for simplicity). Here we first look at the encoding algorithm on Alice's side. She first builds a quadtree on her set of points S_A, retaining only the nonempty quadtree cells. Then starting from level 0 (where the quadtree cells have size 1×1), she constructs an IBLT for each level. For each nonempty quadtree cell, we take its coordinates and the number of points in the cell, and combine them into a key. In this example, we use 3 bits for each coordinate, as well as for the count, which results in 9-bit keys. The figure gives examples on how the keys are generated for two of the quadtree cells. Alice then inserts these keys to an IBLT. Similarly, on level 1, there are 5 nonempty quadtree cells, and Alice builds an IBLT containing 5 keys. She does so for each level of the quadtree, and finally sends all the IBLTs to Bob. □

2.3 The Decoding Algorithm

When Bob receives the message from Alice, he tries to relocate his points in S_B to minimize the EMD, i.e., find S_A^- and S_B^+ so that $\text{EMD}(S_A, S_B' = (S_B - S_B^-) \cup S_B^+)$ can be reduced as much as possible. The first two steps are the same as in the encoding algorithm, i.e., he performs the random shift (using the same ξ he gets from Alice), and builds the quadtree on his data set S_B. In the third step, starting from the bottom level $\ell = 0$, Bob inserts all his nonempty quadtree nodes into the corresponding IBLT he receives from Alice. From the properties of the IBLT, only different nodes will remain, these include different nodes, or the same nodes but with different counts. Then he tries to recover these different nodes, and stops at the lowest level where this declares success.

When Bob reaches a level ℓ on which the IBLT successfully recovers all the differences, he is ready to construct S_A^- and S_B^+. The idea is that he will relocate his points so that the quadtree nodes and counts on this level on his data set are the same as those on Alice's. Specifically, there are two cases:

1. For each quadtree node $[q_1, q_1 + 2^\ell - 1] \times \cdots \times [q_d, q_d + 2^\ell - 1]$ with count c' that Bob has, but for which Alice has a different count $c < c'$ (c could be 0), we add any $c' - c$ of Bob's points inside $[q_1, q_1 + 2^\ell - 1] \times \cdots \times [q_d, q_d + 2^\ell - 1]$ to S_B^-.

2. For each quadtree node $[q_1, q_1 + 2^\ell - 1] \times \cdots \times [q_d, q_d + 2^\ell - 1]$ with count c that Alice has, but for which Bob has a different count $c' < c$ (c' could be 0), we add any $c - c'$ points to S_B^+ with coordinates $(q_1 + 2^{\ell-1}, q_2 + 2^{\ell-1}, \ldots, q_d + 2^{\ell-1})$, namely, the center point in this quadtree cell, to S_B^+.

Finally, Bob should subtract ξ from all points in S_B^- and S_B^+ (modulo Δ) to get them back to their original locations.

The running time of the decoding algorithm is also $O(nd \log \Delta)$.

Example (continued): We continue the example in Figure 3, when Bob receives the IBLTs, one per level of the quadtree. He first also builds a quadtree on his point set S_B. Then level by level, he tries to recover the different quadtree nodes from the IBLT he has received. For each level, he inserts all his nonempty quadtree cells (after converting them to 9-bit keys in the same way as Alice did) into the IBLT, and then performs the retrieval process as described in Section 2.1. Suppose in this example, the IBLT at level 1 declares success on finding all differences, which corresponds to the 5 solid squares shown in Figure 3. Now, for each of Bob's cell with count larger than its counterpart at Alice's side, he deletes points (i.e., put them into S_B^-), arbitrarily chosen from the cell, so that the new count matches that of Alice's. In this example, Bob deletes one point in cell (a) and one point in cell (b) (the crossed points in Figure 3). Next, for each of Alice's cell with count larger than its counterpart at Bob's side, Bob inserts (i.e., put into S_B^+) as many copies of the centerpoints of the cell as necessary so that the counts of the two cells match. In this example, Bob adds one copy of the centerpoint in cell (c) and the centerpoint in cell (d) (the hollow points in Figure 3). This completes the reconciliation process. □

Theorem 1 *The communication cost of our algorithm is* $O(\alpha k \log(n\Delta^d)\log\Delta)$ *bits. The running time of the encoding and the decoding algorithm is both* $O(dn\log\Delta)$*.*

Optimizations. There are a number of optimizations for our basic algorithm described above, which can reduce the hidden constant in the big-Oh. First, a simple observation is that, once the number of nonempty quadtree cells drops below the size of the IBLT, $\gamma\alpha k$, there is no advantage to "compress" them into an IBLT any more. Instead, Alice can simply send over all these quadtree cells together with their counts directly to Bob. Moreover, since the decoding is guaranteed to succeed at this level, there is no need send IBLTs for any higher levels, either.

As we move up one level, we perform a division by 2 to every coordinate of the data points, to fit them in the appropriate quadtree cells. This has the effect of deleting the least significant bits. Thus, at level ℓ, only $\lceil\log\Delta\rceil - \ell$ bits are needed to represent a coordinate of any quadtree cell. This reduces the key length hence the size of the IBLT (in terms of bits).

Meanwhile, we do not need to spend $\log n$ bits to record a quadtree cell count, which should be much smaller than n, especially for the lower levels of the quadtree. Instead, for each level, Alice can first find the maximum cell count, say m, and only use $\log m$ bits for every cell count on this level. Then she also communicates the value of m to Bob. At Bob's side, if he sees one of his quadtree cells on this level with count exceeding m, he already knows that this cell is different from Alice's, so can report it directly instead of inserting it into the IBLT.

Remarks. Our algorithm only requires Alice to send one message to Bob, who then completes the reconciliation all by himself. This reduces latency, and also works well when there are many parties whose data all have to be reconciled with Alice, in which case Alice can simply broadcasts her message. However, if multi-round communication is allowed, we can do a binary search on the $\log\Delta + 1$ levels to find the lowest level on which the IBLT succeeds. This way, Bob can just request the necessary levels from Alice as the binary search proceeds. This reduces the communication cost to $O(\alpha k \log(n\Delta^d)\log\log\Delta)$ bits. But doing a binary search leads to some theoretical subtleties with respect to the probabilistic analysis; please see Section 2.4 for the details.

Similarly, Bob can also do a binary search among the $\log\Delta + 1$ levels, and build the levels of the quadtree only for those needed. Each level of the quadtree can be built in linear $O(nd)$ time, so the running time can be reduced to $O(nd\log\log\Delta)$. However, the bottom-up construction of the quadtree is often more efficient, especially for dense data sets, for which the number of nodes decrease quickly as we go up the quadtree.

2.4 Analysis

In this subsection, we give a theoretical analysis showing that, with high probability, Bob can reduce the EMD between S_A and S_B to at most $O(d)\cdot \mathrm{EMD}_k(S_A, S_B)$. We first give some intuition behind the analysis. W.l.o.g., suppose the optimal solution that moves k points so as to minimize $\mathrm{EMD}(S_A, S_B)$ has moved $S_B^- = \{y_1, \dots, y_k\}$ to $S_B^+ = \{x_1, \dots, x_k\}$, and suppose the remaining points are matched as $(x_{k+1}, y_{k+1}), (x_{k+2}, y_{k+2}), \dots, (x_n, y_n)$. Of course, the algorithm does not know which points have been moved and how the remaining points are matched; the x_i's and y_i' are indexed this way just for the purpose of analysis. Then, $\mathrm{EMD}_k(S_A, S_B) = \sum_{i=k+1}^{n} \|x_i - y_i\|_2$. We will just shorthand this as EMD_k in the following when there is no ambiguity. Intuitively, to reduce the EMD to within a constant factor of EMD_k, Bob needs to correctly identify the top-k pairs $(x_1, y_1), \dots, (x_k, y_k)$, and reduce their average distance to $O(\mathrm{EMD}_k / k)$.

Recall that in our algorithm, both Alice and Bob build a quadtree, and try to reconcile their difference bottom up, level by level, where the IBLT on any level can identify up to αk differences. It is easy to see that for a pair (x_i, y_i) such that $\|x_i - y_i\|_2 > 2^\ell\sqrt{d}$, they must fall in different quadtree cells on level ℓ or below. On the other hand, if $\|x_i - y_i\|_2 < 2^\ell$, they are *likely* to fall in the same quadtree cell on level ℓ or above. So we expect the number of differences to gradually decrease as we go up, and the IBLT will eventually succeed at some level. The hope is that when this happens, Bob can identify $(x_1, y_1), \dots, (x_k, y_k)$, and reduce their distance. However, there are two technical difficulties that we need to overcome:

1. For a close pair (x_i, y_i) with $i > k$, it is only *likely* that they fall into the same quadtree cell on a given level. When it is not the case, they potentially introduce another difference, adding burden to the IBLT's recovery ability. Even if the IBLT succeeds, this might be a "false positive" since y_i should not be related.

2. For a faraway pair (x_i, y_i) with $i \le k$, Bob can only learn from the IBLT that x_i lies in some quadtree cell, without knowing its exact location. In our algorithm, Bob simply relocates to the center point of the cell. This introduces additional error that needs to be accounted for.

The first issue can be properly handled by the random shift. In doing so, we can avoid adversarial cases and guarantee that with good probability, there are not many false positives. To handle the second issue, the redundancy factor $\alpha > 1$ introduced in the algorithm proves useful, which allows us to recover more differences than the top-k pairs in the matching. This lowers the level on which the IBLT can succeed, so that the uncertainty in the quadtree cell can be bounded. Some extra difficulties come from the interplay of these two issues. For example, it is possible that the relocation of a false positive will again introduce some additional error. Thus we need to carefully formulate the charging argument so that the all errors can be charged to EMD_k. In the rest of this subsection, we formalize the above intuition. The analysis below holds for any ℓ_p norm with $p \geq 1$.

First, we need the following observation.

Lemma 1 *With the random shift, the probability that a pair of points (x_i, y_i) fall into different quadtree cells on level ℓ is at most $d^{1-1/p}\|x_i - y_i\|_p/2^\ell$.*

PROOF. Let the coordinates of x_i and y_i respectively be $x_i = (x_{i1}, \ldots, x_{id})$ and $y_i = (y_{i1}, \ldots, y_{id})$. The two points fall in different quadtree cells if and only if for at least one $1 \leq j \leq d$, x_{ij} and y_{ij} are separated by $t2^\ell$ for some integer t. By a union bound, this probability is no more than

$$\begin{aligned} & \sum_{j=1}^{d} \frac{|x_{ij} - y_{ij}|}{2^\ell} \\ \leq \; & \frac{d}{2^\ell}\left(\frac{\sum_{i=1}^{d}|x_{ij} - y_{ij}|^p}{d}\right)^{1/p} \qquad \text{(Jensen's inequality)} \\ = \; & \frac{d^{1-1/p}\|x_i - y_i\|_p}{2^\ell}. \end{aligned}$$

□

The count at any quadtree node is the sum of those at its 2^d children. Thus, for any node v at some level ℓ, if its counts are different in Alice's quadtree and Bob's quadtree, then on level $\ell+1$, at least one of v's children must also have different counts in the two quadtrees. Thus, the number of different nodes monotonically decreases (or stays the same) as we go up the quadtree levels. Let ℓ^* be the lowest level on which there are no more than αk different nodes. With high probability, the IBLT will succeed at this level, possibly even earlier. We will just assume that Bob stops at level ℓ^*; if he stops earlier, the result can only be better.

Let $A = d^{1/p} \cdot 2^{\ell^*}$ be the maximum ℓ_p distance between any two points in a quadtree cell at level ℓ^*. Obviously, for all those pairs (x_i, y_i) with $\|x_i - y_i\|_p \geq A$, they will go to different quadtree cells at this level. The issue is that it is possible for a pair (x_i, y_i) with $\|x_i - y_i\|_p < A$ to go to different cells as well. In this case the algorithm will introduce a false positive. This might increase the EMD since Bob only knows the cell which contains x_i. Our decoding algorithm simply relocates y_i to the center point of the cell, but doing so might introduce an error up to $A/2$, which could be larger than $\|x_i - y_i\|_p$ itself. Below we will show that such false positives will not happen too much, and even for those that do happen, the extra error introduced can be charged to EMD_k.

Set $\eta = \frac{\alpha/2-1}{4d^{1-1/p}}$. We focus on a level ℓ such that $2^\ell = \mathrm{EMD}_k/(\eta k)$. [4] We will show that with probability at least $3/4$, there are at most αk different quadtree nodes on this level. If this is the case, then according to the definition of ℓ^*, we have $\ell^* \leq \ell$.

For $i = k+1, \ldots, n$, let T_i be the indicator variable of the event that the pair (x_i, y_i) go to different cells at level ℓ. Then by Lemma 1, we have that $\Pr[T_i = 1] \leq \frac{d^{1-1/p}\|x_i - y_i\|_p}{2^\ell}$. Let $T = \sum_{k+1}^{n} T_i$. We have

$$\begin{aligned} \mathbb{E}[T] &\leq \sum_{i=k+1}^{n} \frac{d^{1-1/p}\|x_i - y_i\|_p}{2^\ell} \\ &= \frac{d^{1-1/p} \cdot \sum_{i=k+1}^{n}\|x_i - y_i\|_p}{\mathrm{EMD}_k/(\eta k)} \\ &= \frac{d^{1-1/p} \cdot \mathrm{EMD}_k}{\mathrm{EMD}_k/(\eta k)} \\ &= d^{1-1/p}\eta k. \end{aligned}$$

By Markov inequality, we have $T \leq 4d^{1-1/p}\eta k = (\alpha/2-1)k$ with probability at least $3/4$. Each of the T pairs contribute at most 2 different quadtree nodes, while the first k pairs $(x_1, y_1), \ldots, (x_k, y_k)$ contribute at most $2k$ different nodes. So we have that with probability at least $3/4$, there are no more than $2(\alpha/2-1)k + 2k = \alpha k$ different nodes at level ℓ, which means that $\ell^* \leq \ell$ and

$$\begin{aligned} A &= d^{1/p} \cdot 2^{\ell^*} \\ &\leq d^{1/p} \cdot 2^\ell \\ &= d^{1/p} \cdot \mathrm{EMD}_k/(\eta k) \\ &= \frac{4d}{(\alpha/2-1)k}\mathrm{EMD}_k. \end{aligned}$$

It remains to bound the EMD after the reconciliation stops at level ℓ^*. Note that at most $\alpha k/2$ pairs were put into different cells, and $\alpha k/2$ points were moved after reconciliation. Then, for each cell at level ℓ^*, the cell counts for Alice and Bob's become equal. We say that a pair (x_i, y_i) is 'broken' if the x_i and y_i were put in different cells pre-reconciliation, or if either one of the points was moved during reconciliation. A point is 'broken-off' if it belongs to a pair that is broken. Observe in each cell the number of Alice's broken-off points is exactly the same as Bob's, or else their number of points in the cells will not be equal; in total at most αk pairs would be broken, so $2\alpha k$ points would be broken-off. To bound the EMD by above, we simply pair of each broken-off point of Alice to a broken-off point of Bob in the same cell arbitrarily, to produce a perfect matching between their points. The distances between these newly paired up points will be at most A. So the EMD after reconciliation is

$$\begin{aligned} \mathrm{EMD}(S_A, S'_B) &\leq \alpha k A + EMD_k \\ &= \alpha k \frac{4d}{(\alpha/2-1)k} EMD_k + EMD_k \\ &= \left(1 + 8d + \frac{16d}{\alpha - 2}\right) EMD_k \end{aligned}$$

By setting $\alpha > 2$ the approximation ratio is $O(d)$.

Theorem 2 *Our algorithm for the robust set reconciliation problem can find an S'_B such at that* $\mathrm{EMD}(S_A, S'_B) \leq O(d) \cdot \mathrm{EMD}_k(S_A, S_B)$ *with probability at least* $3/4$. *This holds for the EMD defined by any ℓ_p norm for $p \geq 1$ and in any dimensions.*

Remarks. It should be easy to see from the analysis that we can increase the success probability from $3/4$ to a constant arbitrarily close to 1, which only affects the value of α by a constant factor. In the analysis above, we have also ignored

[4] For convenience, we assume that $\mathrm{EMD}_k/(\eta k)$ is a power of 2. Such an assumption will not change the approximation ratio by more than a factor of 2.

the failure probability of the IBLT, but since it is only $o(1)$, it does not change the result: We only require one IBLT to succeed, i.e., the one on level ℓ^*, for the guarantee to hold. If the IBLT on some lower levels succeeds earlier in the bottom-up search, the resulting EMD can only be smaller.

As one can observe, our analysis is quite conservative, the actual approximation factor is actually much better as demonstrated in our experiments in Section 4.

Multi-round communication protocol. As mentioned at the end of Section 2.3, if multi-round communication is allowed, we can reduce communication cost by doing a binary search on the $\log\Delta + 1$ levels of the quadtree to find the lowest level on which the IBLT succeeds. However, this will require all the $O(\log\log\Delta)$ IBLTs probed in the binary search to succeed (provided that there are no more than αk different quadtree nodes on that level), which means that we need to set the failure probability to $O(1/\log\log\Delta)$ in the IBLTs and then apply a union bound. Recall that the failure probability of an IBLT is $1/m^{O(1)} = 1/k^{O(1)}$. To reduce this to $O(1/\log\log\Delta)$, we can keep $O(\lceil\log\log\log\Delta/\log k\rceil)$ independent instances of the IBLT so that the IBLT will succeed as long as one of the instances succeeds. Thus, we have the following variant of our algorithm.

Corollary 1 *Our algorithm can be made to run in $O(\log\log\Delta)$ rounds of communication, with the total communication cost being $O(k\log(n\Delta^d)\log\log\Delta\lceil\log\log\log\Delta/\log k\rceil)$ bits. It finds an S'_B such that $\mathrm{EMD}(S_A, S'_B) \le O(d)\cdot\mathrm{EMD}_k(S_A, S_B)$ with probability at least $3/4$.*

2.5 When the Number of True Differences is Unknown

k is an important parameter in our algorithm, which should be set to be at least the number of truly different points between S_A and S_B, so that EMD_k is small. However, in most cases we do not know the number of true differences. In the (exact) set reconciliation problem, one can first estimate the size of $S_A \oplus S_B$ so that proper parameters can be set in the IBLT [7]. Unfortunately, in the presence of small errors, the estimation does not work, as it counts all differences, large or small. In fact, if we want to exclude small differences from the estimation, we will have to compute (at least approximately) the EMD between two distributed data sets, which is known to be a difficult problem [1, 20].

Thus, we propose the following two ways to deal with an unknown k. First, if a communication budget is given, we can easily decide the maximum k allowed, and just use this k. This is actually how we defined the robust set reconciliation problem in the first place, which represents a "best-effort" approach.

Alternatively, if no explicit communication budget is given, but the user desires a certain quality of the reconciliation, say, the resulting EMD must be below some threshold τ. Then we can take the following approach. Starting with a small k, we iteratively run our algorithm with a doubled k. Since the communication cost is linear in k, the total communication cost is just twice that for the last round. At the end of each iteration, we use the following conservative estimation for the EMD. Suppose the decoding algorithm succeeds at level ℓ. Then we know that after the reconciliation, all pairs must have been corrected to within a distance of at most $d^{1-1/p}2^\ell$, so the EMD is at most $nd^{1-1/p}2^\ell$. We stop the process when this estimate is below τ.

3. THE LOWER BOUND

We first observe that when there are only k differences between the two data sets, then our problem degenerates into the standard set reconciliation problem. In this case, $\mathrm{EMD}_k = 0$, so an algorithm that wishes to give any (multiplicative) approximation to EMD_k has to recover all the k differences exactly. Therefore, the standard communication lower bound for set reconciliation still holds for our problem, namely, the algorithm has to at least transmit all the differences. Since in our case, each element is a point in $[\Delta]^d$, we have a lower bound of $\Omega(k\log(\Delta^d))$ bits. This holds even for multi-round communication protocols. Note that our multi-round communication protocol is only an $O(\log\log\Delta)$ factor away from this lower bound for constant dimensions (ignoring even lower order terms).

Our one-way communication algorithm has an extra $O(\log\Delta)$ factor from the lower bound above, and we present a lower bound showing that this is essentially necessary. Specifically, we show that any one-way communication protocol that solves the robust set reconciliation problem with a quality guarantee of $\mathrm{EMD}(S_A, S'_B) = O(C)\cdot\mathrm{EMD}_k(S_A, S_B)$ requires a message with at least $\Omega(k\log(\Delta^d/k)\log\Delta/\log C)$ bits. This lower bound holds for the EMD defined by any ℓ_p norm for $p \ge 1$ and for any dimensions d.

We first illustrate the idea through the one-dimensional case. We first present a family of hard instances for the robust set reconciliation problem on the one dimensional grid $[\Delta]$. Alice and Bob hold sets of n points S_A and respectively S_B on grid $[\Delta]$. The construction is performed in two steps. In the first step, we choose p *point center* locations $1, \Delta/p+1, 2\Delta/p+1, \ldots, (p-1)\Delta/p+1$, and in both S_A and S_B we assign n/p points to each point center. In the second step, we move points from these point centers in S_A and S_B to the right. At this step we make the point sets S_A and S_B different. We pick L $(= \Theta(\log\Delta))$ subsets $X_1, \ldots, X_L \subseteq [p]$ such that $|X_i| = k$ for all $i \in [L]$. In S_A, for all $i \in [L]$, for all $j \in X_i$, we move one point in the j-th point center by a distance of 2^{Bi} where B is a technical parameter. In S_B we perform similar operations: we first pick a random $I \in [L]$, and then for all $i = \{I+1, \ldots, L\}$, for all $j \in X_i$, we move one point from the j-th point center by a distance of 2^{Bi}. Note that S_A and S_B differ by those points that are moved in S_A indicated by $X_1, \ldots, X_I$. These points remain in point centers in S_B. The k most significant differences in point set S_A and S_B are the k points that Alice moved by distance 2^{BI}, that is, those points indicated by X_I. Intuitively, if Bob wants to correct most of these points, Bob has to learn X_I approximately.

The technical implementation of this idea is a reduction from a variant of the one-way two-party communication problem called *Augmented Indexing*. Due to space constraints, we leave the details of this reduction to the technical report version of the paper, while only state the result here.

Theorem 3 *Any randomized communication protocol that computes a C-approximation to the robust set reconciliation problem on the d-dimensional grid $[\Delta]^d$ with error probability at most $1/3$ requires a message of size*

$$\Omega(k\log(\Delta^d/k)\log\Delta/\log C),$$

assuming that $k < \Delta^{d/2}$. This holds for the EMD defined by any ℓ_p norm for $p \ge 1$.

4. EXPERIMENTS

4.1 Implementation Details

For the IBLTs we use 5 hash functions, so we take $\gamma = 1.425$ and use a table size of $\lceil \gamma\alpha k \rceil$ cells [8]. The 5 hash functions are chosen randomly from the following 5-wise independent family of hash functions [22]:

$$h(x) = \left(\sum_{i=0}^{4} a_i x^i \mod p \right) \mod \lceil \gamma\alpha k \rceil,$$

where $a_i \in \mathbb{Z}_p$ for $i = 0, 1, \ldots, 4$, and p is a large prime, here taken as $2^{31} - 1$. The fingerprint function we use is $f(x) = \hat{x}(2\check{x} + 1) + \check{x}^2$ (truncated to the last 32 bits), where $\hat{x}$ and $\check{x}$ respectively are the first $\frac{\log x}{2}$ and the last $\frac{\log x}{2}$ bits of x.

4.2 A Baseline Method

Existing algorithms for the standard set reconciliation problem clearly do not work for our problem, since they would try to correct all errors, large or small, incurring linear communication cost, which is the same as the naive wholesale transfer approach. A more reasonable method is to do a lossy compression of the data before sending it over, as small errors are tolerated in our scenario. It also gives us a communication-quality tradeoff on which we can compare with our algorithm. Specifically, we use the Haar wavelet compression with the same communication costs as our algorithm. That is, if the communication cost for our algorithm is set to x% of the raw data, then we retain the top x% of the coefficients of the wavelet transform in terms of absolute value, while zeroing the rest. Then Bob will simply invert the wavelet transform using these coefficients to get an approximate version of Alice's data. Note that Bob's own data is not used in this baseline method. Finally, we compute the EMD between Alice's original data set and the one after this lossy compression-decompression as quality assessment.

One optimization we did that turned out to be quite effective is to sort the data (for 1D numerical data sets) before feeding it to the wavelet transform. This is allowed as the data is unordered set, and sorting increases the smoothness of data, which improves the effectiveness of wavelet transformation.

4.3 One-Dimensional Experiments

We start with one-dimensional experiments, i.e., the data set consists of n numerical values. This represents an important use case of the problem, and more importantly, it allows us to compute the EMD efficiently in $O(n \log n)$ time, so that we can accurately measure the reconciliation quality.

Data sets. For 1D experiments, we used the MPCAT-OBS data set, which is an observation archive available from the Minor Planet Center[5]. We picked $n = 10^6$ right ascensions records[6], which are integers ranging from 0 to $\Delta = 8,639,999$. This data set is given to Alice as S_A.

Then we generated Bob's point set S_B as follows. We first injected $k = 100$ "true differences by randomly picking k points in S_A and moving each of them to another location that is uniformly distributed in the region $[\Delta]$. Then for a perturbation level ε, we add to every point in S_A some noise that is uniformly distributed in $[-\varepsilon, \varepsilon]$.

Setup and quality measure. Although there appear to be multiple parameters k, α, γ, but we should emphasize that they are only used in the analysis and for stating the quality guarantee: α determines the approximation ratio, k is the target (i.e., EMD_k) on which the approximation ratio is calculated, and γ decides the failure probability. They are actually not distinguishable by the algorithm at all. For the algorithm, the IBLT table size, $s = \gamma\alpha k$, is the only parameter, which is solely decided by the communication budget. So no parameter tuning is needed for the algorithm.

The quality measure we used is the approximation ratio, i.e., $\text{EMD}(S_A, S'_B)/\,\text{EMD}_k(S_A, S_B)$, where $k = 100$ as set before. Note that this approximation ratio is computed after the algorithm is done on the raw data and the output of the algorithm. The algorithm itself does not know k and cannot compute EMD or EMD_k.

We set $\varepsilon = 1$ as the default perturbation level; note that this actually represents the hardest case as this makes $\text{EMD}_k(S_A, S_B)$, the denominator in the approximation ratio, small. We will vary ε in one set of experiments.

Varying communication cost. First we vary the communication budget of the algorithm, as a percentage of the raw data size. For each communication budget, we run our algorithm 11 times, each time with a different random shift and different hash functions in the IBLTs. We plot the minimum, median and maximum approximation ratios among the 11 runs in Figure 4a. For the wavelet method, we similarly also compute the ratio of its resulting EMD to EMD_k.

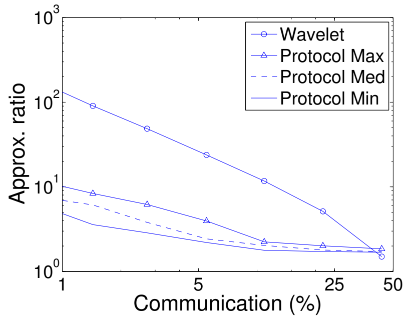
We can see that our algorithm performs much better than wavelet compression, consistently beating the latter by one to two orders of magnitude. The gap becomes smaller when the communication cost is higher than 10% of the raw data. Wavelet compression eventually catches up in performance at very large communication costs, near 50%, and gives near-perfect reconciliation at 87% cost. However, high communication cost scenarios are not very interesting for our problem, whose goal is to reconcile a small number of true differences under the abundance of noise. We also observe that the approximation ratio is roughly inversely proportional to communication cost (note the log-log scale of the plot), and flatten off for very large communication. This is quite consistent with the theoretical analysis that the approximation ratio is $1 + 8d + \frac{16d}{\alpha - 2}$, except that the actual constants are somehow better.

The average encoding and decoding times are plotted in Figure 4d. They are about 1 second, scaling very mildly with communication. This is because the running times are dominated by quadtree construction and inserting quadtree nodes in to the IBLTs, both of which are independent of the size of the IBLTs. The running time depends very slightly on the randomness in the algorithm, with less than 10% variation over 11 runs.

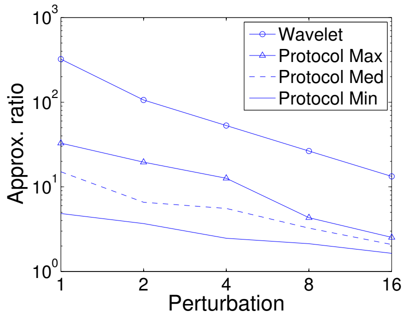
Varying perturbation. Note that our data set has $\Delta/n = 3.2$, so it is quite dense, a perturbation level $\varepsilon \geq 2$ would often mix signal with noise. So we reduce n to 10^5, to reduce the density by a magnitude. Now there is room for perturbations of size $\varepsilon = 1, 2, 4, 8, 16$. We also reduce k to 10 by proportion. We fix the communication budget at 4% and vary ε. For each value of ε, we again run our algorithm

[5]`http://www.minorplanetcenter.net/iau/ecs/mpcat-obs/mpcat-obs.html`

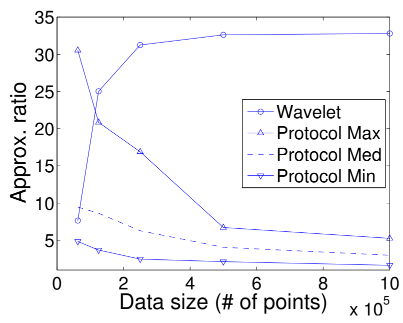
[6]Right ascension is an astronomical term used to locate a point in the equatorial coordinate system.



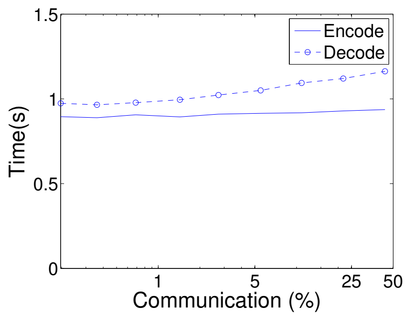
(a) Approx. ratio vs. communication



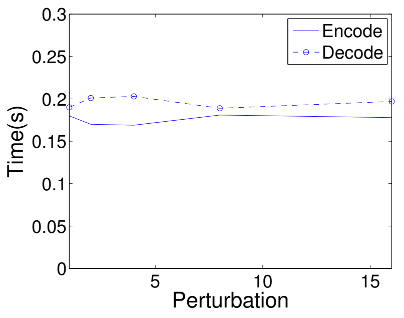
(b) Approx. ratio vs. perturbation



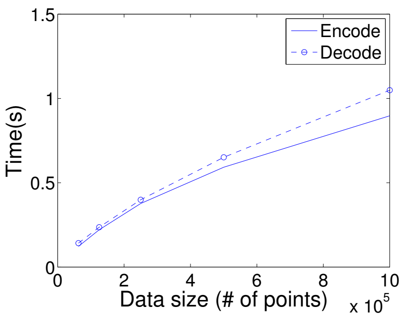
(c) Approx. ratio vs. n



(d) Running time vs. communication



(e) Running time vs. perturbation



(f) Running time vs. n

Figure 4: Experimental results on 1D data.

11 times, and plot the minimum, median and maximum approximation ratios achieved. This is also compared with that of the wavelet method (Figure 4b).

Note that the wavelet compression method does not use any information that Bob possesses, so the decrease in approximation ratio is solely due to the increase in the denominator EMD_k, which is expected to be proportional to ε. For our algorithm, EMD_k increases as ε gets larger, and $\text{EMD}(S_A, S'_B)$ increases not as much, so the approximation ratio decreases as perturbation gets larger.

The running time does not vary much in the size of perturbation (Figure 4e), up and down within a $\pm 10\%$ band.

Varying data size. Finally, we examine how the algorithm scales with the data size n. We took 5 different values for n: 62500, 125000, 250000, 500000, 1000000, and for each n, picked a random subset of size n from the original data set. Again on each subset, we run our algorithm 11 times, and plot the minimum, median and maximum approximation ratios in Figure 4c, while the communication cost is set at 4% of the data. Here we can clearly see that it gets smaller for larger n. This is again mostly due to EMD_k getting larger as n gets larger. On the other hand, the quality of wavelet compression is good over very small data sets, but then deteriorates rapidly. It quickly levels off, after which point the quality seems to be independent of n.

We see that the running time scales nearly linearly with n (Figure 4f), which is expected from our analysis.

4.4 Multi-dimensional experiments

EMD estimation. As reviewed earlier, computing EMD in dimension $d \geq 2$ requires $O(n^3)$ time, which is unbearable even for moderately large data sets. So in our experiments, we used the following rough estimates for EMD and EMD_k. As before, we first injected k true differences in the data, by relocating k points to complete random positions in the grid $[\Delta]^d$. Then we added a random perturbation in the range $[-\varepsilon, \varepsilon]$ to every coordinate of each point. We compute EMD_k by assuming that the optimal solution will identify the k true errors, and match every remaining point to its perturbed location. If perturbation is small and the data is sparse, this should actually be quite close to the true EMD_k.

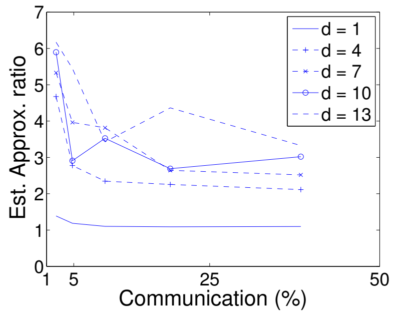
To estimate the EMD of our algorithm, we recall that the algorithm starts from the bottom level of the quadtree, and works its way up until it succeeds at some level ℓ. Let the diagonal distance of a quadtree cell at this level be $A = 2^\ell \sqrt{d}$. Recall from our analysis that if the algorithm relocates a point at this level, then it is guaranteed that its distance to its counterpart is at most $A/2$. Thus, we use the following rather rough and conservative estimate:

$$\widehat{\text{EMD}} = \text{EMD}_k + (\text{number of points moved}) \times A/2.$$

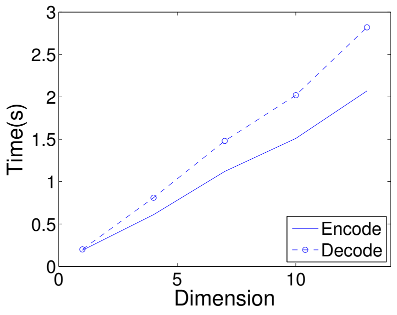
As there is no reasonably efficient way to estimate the EMD for the baseline method, it is omitted from the multi-dimensional experiments. We could have computed it using the $O(n^3)$ method on some small data sets, but from our experience in the 1D case (Figure 4c) its performance on small data sets may not be any indication on larger ones.

Data set. We used the MiniBooNE particle identification data set found in the UCI Machine Learning Repository[7]. It has 130,065 points in 50 dimensions. The coordinates are floating-point numbers, but can be scaled to integers in the interval $[0, 2 \times 10^7]$ without loss of precision. For each of

[7]`http://archive.ics.uci.edu/ml/datasets/MiniBooNE+particle+identification`



(a) Approx. ratio vs. dimension



(b) Running time vs. dimension

Figure 5: Experimental results on multi-dimensional data.

$d = 1, 4, 7, 10, 13$, we picked the first d dimensions and obtained a d-dimensional data set. We set $k = 10, \varepsilon = 1$, and injected true differences and perturbation as before.

Results. We vary the communication cost and compute the approximation ratios $\widehat{\mathrm{EMD}}/\mathrm{EMD}_k$, and the results are plotted in Figure 5a. It shows very reasonable approximation ratios even for d up to 13. On small communication (i.e., small α), the ratios are higher, but it quickly drops as α increases. The approximation ratio gently increases with d, which in general agrees with the $O(d)$ theoretically guaranteed approximation ratio.

One should also keep in mind that the above estimate $\widehat{\mathrm{EMD}}$ is quite conservative; we expect the actual approximation ratio to be better. For instance, we estimate the distance between a relocated pair at the maximum $A/2$; in reality, this may be around $A/4$, which could improve the approximation ratio by a factor of 2.

Finally, we are assured that the running times scale nearly linearly with d (Figure 5b).

4.5 Image Reconciliation

Finally, we apply our algorithm to the reconciliation of two sets of images, which is one of the initial motivations of this work. We took a collection of 10000 high quality JPEG images and gave them to Alice. Bob has the same 10000 images, but after a 95%-quality JPEG re-compression. Then Bob changed k of his images to completely new ones, and our goal is thus to identify these images. Note that this will also give Alice k images that are different from Bob's, there are actually $2k$ different images that we need to find.

We map the images to points in a feature space. For this experiment, we used the simplest feature signature, the color histogram with 6 buckets (the results would only be better if more sophisticated features were used), so each image is mapped to a 6-dimensional point. The coordinates of the points (i.e., number of pixels in the histogram buckets) are 22-bit integers. These points constitute S_A and S_B as input to our algorithm. The JPEG re-compression has the effect of perturbing a point by a small distance in the feature space. This distance is roughly $1/10^4$ of that between two distinct images.

Algorithm modification. Recall that our algorithm will delete and add some points in S_B, so as to reduce the EMD. In particular, when Bob finds a quadtree cell with count larger than in its counterpart at Alice's side, he deletes a point chosen arbitrarily from the cell. If there is more than one point in the cell, this point may not be the true difference. This is not a problem as far as EMD is concerned, as moving any point in the cell to an (approximately) correct location will reduce the EMD, but it is not sufficient for the image reconciliation task, where we want to exactly pinpoint the different images. However, our algorithm can be easily modified to accomplish this task, with one more round of messages, described as follows (the first 2 steps are the same as before).

1. Alice sends Bob the IBLTs of her quadtree.
2. As before, Bob queries the IBLTs level by level, until he reaches a level ℓ where he retrieves all the quadtree differences.
3. Based on the quadtree differences, Bob reports to Alice a set of candidates $\hat{S}_B$, which includes all points in any cell where Bob's point count exceeds Alice's. Bob also sends Alice his IBLT at level ℓ (only).
4. For each point x in $\hat{S}_B$, Alice finds its nearest neighbor in S_A. If the distance between the two is smaller than the perturbation threshold (we used 10 times the average perturbation distance as the threshold), we discard x; otherwise we declare x to be a difference.
5. Likewise, Alice recovers the quadtree differences at level ℓ, and sends Bob a candidate set $\hat{S}_A$.
6. Bob similarly checks each point in $\hat{S}_A$ to see if it matches with one of his point, and declares a difference if no such match is found.

Again, we operate in the fixed communication budget scenario, and the algorithm is unaware of the value of k. While the size of the IBLT can be easily adjusted according to the budget, we don't however have control over the size of $\hat{S}_A$ and $\hat{S}_B$. So we adopt the following strategy. We give half of the communication budget to transmitting the IBLTs, reserving the other half for $\hat{S}_A$ and $\hat{S}_B$ (thus each getting 1/4 of the budget). When the size of either $\hat{S}_A$ or $\hat{S}_B$ exceeds the remaining budget, we simply take a random sample. This may miss out some targets but it is the best effort we can afford under the budget. On the other hand, Alice omits all bottom levels of her quad tree whose cell diameters are dominated by the noise threshold, which reduces space used by IBLTs without affecting recovery; she can also more aggressively remove levels where her cell counts are small enough

		Budget				
		2%	4%	6%	8%	10%
	5	0%	56%	**92%**	**100%**	**100%**
	10	2%	34%	84%	**100%**	**100%**
k	15	0%	28%	80%	**100%**	**100%**
	20	0%	19%	67%	**98%**	**99%**
	25	0%	5%	66%	87%	**99%**

Table 1: Recovery rate for image reconciliation

that she estimates Bob will not report too many points. Finally, if the size of $\hat{S}_B$ is way below the given budget in step 3 above, Bob will go up the quadtree levels above ℓ (although the IBLT already succeeds at level ℓ). This simple heuristic can make better use of the the communication budget, and help to increase the recovery rate.

Results. We experimented with different communication budgets, at $2\%, 4\%, 6\%, 8\%, 10\%$ of the raw data size. Raw data here refers to the 10000 6-dimensional points, as the naive approach would send all of Alice's data to Bob or vice versa. We also varied k from 5 to 25. For each combination of communication budget and k, we ran our algorithm 5 times and report the average recovery rate is Table 1. Recovery rate here is the number of reported differences over $2k$ (the total number of different images). Note that since both Alice can Bob verify for every candidate if it matches an existing image of his/her own, there are no false negatives.

We can see that with communication budget at 2%, there is virtually no recovery. It means that the small size of the IBLTs is not sufficient to accommodate the $2k$ differences, so the retrieval succeeds only at a very high level of the quadtree, which in turns leads to a larger number of candidates. With the remaining 1% communication budget, it is unlikely for the sampling to reach the true differences. As we increase the budget, the recovery rate increases significantly, as an increased budget allows both larger IBLTs as well as higher sampling rates. The bold numbers in the table indicate that all candidates have been sent over with no need to sample[8]. Finally, with 10% communication cost, our algorithm has essentially identified all the different images.

[8]More precisely, it indicates that there is no need to sample in a majority of the runs, as each number is averaged over 5 runs.

5. REFERENCES

[1] A. Andoni, K. Do Ba, P. Indyk, and D. Woodruff. Efficient sketches for earth-mover distance, with applications. In *Proc. IEEE Symposium on Foundations of Computer Science*, 2009.

[2] I. Assent, M. Wichterich, T. Meisen, and T. Seidl. Efficient similarity search using the earth mover's distance for large multimedia databases. In *Proc. IEEE International Conference on Data Engineering*, 2008.

[3] E. Brewer. CAP twelve years later: How the "rules" have changed. *IEEE Computer*, 2012.

[4] E. J. Candes and P. A. Randall. Highly robust error correction byconvex programming. *IEEE Transactions on Information Theory*, 54(7):2829–2840, 2008.

[5] G. Cormode, M. Paterson, S. C. Sahinalp, and U. Vishkin. Communication complexity of document exchange. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 2000.

[6] C. Dwork. Differential privacy: A survey of results. In *Proceedings of The 5th Annual Conference on Theory and Applications of Models of Computation*, 2008.

[7] D. Eppstein, M. T. Goodrich, F. Uyeda, and G. Varghese. What's the difference? Efficient set reconciliation without prior context. In *SIGCOMM*, 2011.

[8] M. Goodrich and M. Mitzenmacher. Invertible Bloom lookup tables. In *Allerton Conference*, 2011.

[9] D. Guo and M. Li. Set reconciliation via counting bloom filters. *IEEE Transactions on Knowledge and Data Engineering*, 25(10):2367–2380, 2013.

[10] H. Guo, P.-Å. Larson, R. Ramakrishnan, and J. Goldstein. Relaxed currency and consistency: How to say "good enough" in SQL. In *Proc. ACM SIGMOD International Conference on Management of Data*, 2004.

[11] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society of Industrial and Applied Mathematics, 1996.

[12] P. Indyk. A near linear time constant factor approximation for euclidean bichromatic matching (cost). In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[13] E. H. Jacox and H. Samet. Spatial join techniques. *ACM Transactions on Database Systems*, 32(1), 2007.

[14] H. Köhler, X. Zhou, S. Sadiq, Y. Shu, and K. Taylor. Sampling dirty data for matching attributes. In *Proc. ACM SIGMOD International Conference on Management of Data*, 2010.

[15] T. Kraska, M. Hentschel, G. Alonso, and D. Kossmann. Consistency rationing in the cloud: Pay only when it matters. In *Proc. International Conference on Very Large Data Bases*, 2009.

[16] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[17] S. Lin and J. Daniel J. Costello. *Error Control Coding: Fundamentals and Applications*. Prentice Hall, 1983.

[18] Y. Minsky, A. Trachtenberg, and R. Zippel. Set reconciliation with nearly optimal communication complexity. *IEEE Transactions on Information Theory*, 49(9):2213–2218, 2003.

[19] R. Sharathkumar and P. K. Agarwal. A near-linear time eps-approximation algorithm for geometric bipartite matching. In *Proc. ACM Symposium on Theory of Computing*, 2012.

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

[21] H. Wada, A. Fekete, L. Zhao, K. Lee, and A. Liu. Data consistency properties and the trade-offs in commercial cloud storages: The consumers' perspective. In *Proc. Conference on Innovative Data Systems Research*, 2011.

[22] M. Wegman and J. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22(3):265–279, 1981.