If it ain't broke, don't fix it: Sparse metric repair*

Anna C. Gilbert[1] and Lalit Jain[2]

Abstract— **Many modern data-intensive computational problems either require, or benefit from distance or similarity data that adhere to a metric. The algorithms run faster or have better performance guarantees. Unfortunately, in real applications, the data are messy and values are noisy. The distances between the data points are far from satisfying a metric. Indeed, there are a number of different algorithms for finding the closest set of distances to the given ones that also satisfy a metric (sometimes with the extra condition of being Euclidean). These algorithms can have unintended consequences; they can change a large number of the original data points, and alter many other features of the data.**

The goal of *sparse metric repair* is to make as few changes as possible to the original data set or underlying distances so as to ensure the resulting distances satisfy the properties of a metric. In other words, we seek to minimize the sparsity (or the ℓ_0 "norm") of the changes we make to the distances subject to the new distances satisfying a metric. We give three different combinatorial algorithms to repair a metric sparsely. In one setting the algorithm is guaranteed to return the sparsest solution and in the other settings, the algorithms repair the metric. Without prior information, the algorithms run in time proportional to the cube of the number of input data points and, with prior information we can reduce the running time considerably.

I. INTRODUCTION

For many data analysis and processing tasks, we assume that the data upon which we run the various algorithms are points in a metric space; that is, the distances between the data points satisfy the properties of a (semi-)metric. We do not necessarily assume that the distances are Euclidean, nor that we have distinct points, but many of these algorithms do require the distances to satisfy the triangle inequality. In machine learning, for example, learning the metric from which a set of points is drawn is critical for tasks such as clustering [1]. For kernelized learning algorithms, we learn the kernel (rather than the more general metric) by learning Euclidean distances [2]. For clustering data, ensuring that all the distances satisfy the triangle inequality constraints guarantees the algorithms run efficiently and produce correct clusters. Indeed, many approximation algorithms for standard theoretical computer science tasks have better approximation guarantees and highly efficient solutions (e.g., sublinear in the input size) when the input is from a metric space. See Indyk's survey of sublinear approximation algorithms [3] for examples such as the Traveling Salesman Problem, Maximum Spanning Tree, and k-median. Finally, in DNA sequence alignment problems, we build amino acid substitution matrices [4] (known as Point Accepted Mutation and Block Substitution Matrices) to speed up pattern matching and alignment tasks, under the assumption that the matrices represent distances (from a metric) amongst amino acids.

Unfortunately, data are messy, physical measurements are noisy, values are missing, and data sets are far from perfect metrics. Before we can employ any of these efficient data analysis algorithms for critical tasks in machine learning, bioinformatics, or graph analysis, we must *repair the metric*. We must adjust the data points so as to ensure their distances satisfy the properties of a metric. In the psychometric literature, there is a long history of work on the implications of data that fail to satisfy a metric, as well as many algorithms to "fix" the data so that the distances are Euclidean. See [5] and the references therein (especially the work of Shepard from the 1960s). For clustering, Baraty, et al. [6] discuss what impact the failure of distances to adhere to a metric has upon clustering algorithms and they use rectifiers to continuously modify all distances to obtain a metric. Brickell, et al. [7] were the first to refer to this problem specifically and to define the metric nearness or repair problem precisely: given a set of points or, equivalently, a matrix of distances, find the closest (in an appropriate norm) matrix of distances that satisfies metric properties. They devise several algorithms based upon convex optimization, as well as several combinatorial algorithms, closely related to all pairs shortest path computations on undirected, weighted graphs.

All of the previously developed algorithms can (and, in practice, do) adjust all of the distances in their repairs. Biswas and Jacobs [8] observe, however, that in many applications, more than 99% of the triangle inequalities are satisfied. In other words, these algorithms make unnecessary and potentially deleterious changes in the original data. While the algorithms may fix the metric, they may break or change in unforeseen ways other aspects of the data set. Which leads us to advise (colloquially), "If it ain't broke, don't fix it." Our precise mathematical goal is to make as few changes as possible to the original data set or underlying distances so as to ensure the resulting distances satisfy the properties of a metric. In other words, we seek to minimize the sparsity (or the ℓ_0 "norm") of the changes we make to the distances among data points subject to the new distances satisfying a metric. We refer to this problem as *sparse metric repair*. Note that we do not insist that the repaired metric be Euclidean. It could be from a manifold or class based from the output of a classifier. Enforcing a triangle inequality is the least we can ask from similarity or distance data.

*ACG was supported by National Science Foundation grants CCF-1161233 and CIF-0910765.

[1]Anna C. Gilbert is with the Department of Mathematics, University of Michigan, Ann Arbor, MI annacg@umich.edu

[2]Lalit Jain is with the Department of Mathematics, University of Michigan, Ann Arbor, MI lalitj@umich.edu

We formulate three different repair scenarios, decrease only metric repair, increase only metric repair, and the general case. Each is structurally different from the others and, as a result, have different algorithmic solutions and guarantees. The decrease only metric repair problem is closely related to the all pairs shortest path problem (first observed by Brickell, et al. [7]) and we use these observations to formulate a combinatorial algorithm that solves the sparse metric repair problem in time that is cubic in the number of data points. We give similar results for the other two scenarios, although the other two are much more difficult to analyze (as we show). The common feature of all of our algorithms is that without prior knowledge of which triangles are broken or which triples of distances fail to satisfy the triangle inequality, the running time is cubic. With such knowledge, the algorithms run considerably faster as they fix only what is broken.

We begin the paper with a precise statement of the sparse metric repair problem and a discussion of several restricted settings in which to study the problem. In Section IV, we present algorithmic results for the three settings, along with some probabilistic analysis of a random data model. We then analyze the experimental performance of our algorithms in Section V on a range of problem types. We also compare our combinatorial algorithms with those from a more standard convex relaxation approach.

II. Preliminaries and Notation

We denote the subspace of positive real symmetric $n \times n$ matrices by $\text{Sym}_n(\mathbb{R}_{\geq 0})$. A matrix $D \in \text{Sym}_n(\mathbb{R}_{\geq 0})$ is said to be *metric*, or equivalently a *distance matrix*, if

$$D_{ii} = 0 \text{ for } 1 \leq i \leq n$$
$$D_{ij} \leq D_{ik} + D_{jk} \text{ for } 1 \leq i, j, k \leq n.$$

Note that our notion of metric is often known as a semi-metric. In particular, we do not insist that $D_{ij} = 0$ iff $i = j$.

In some situations, we wish to indicate that one matrix is element-wise less than another matrix and we write $A \preceq B$ to signify that each element of A is less than or equal to its corresponding element in B. For example, one of the conditions a distance matrix satisfies is $0 \preceq D$.

We will refer to the complete (undirected) weighted graph on n vertices with edge weights given by D as $K_n(D)$. Our triangles are labeled according to their associated triangle inequality. We say that the triangle inequality corresponding to *triangle* ijk for a distance matrix D is $D_{ij} \leq D_{ik} + D_{jk}$. Note that triangle $ijk = jik$, but $ijk \neq ikj \neq jki$. In particular there are $n(n-1)(n-2)/2$ total triangles. We say that D_{ij} occurs on the left hand side of the above triangle inequality, and D_{ik} and D_{jk} occur on the right. The triangle ijk and the associated triangle inequality are used interchangeably.

Finally, we let $\mathcal{T}(D)$ denote the set of all ijk with $D_{ij} > D_{ik} + D_{jk}$ for a $D \in \text{Sym}_n(\mathbb{R}_{\geq 0})$. We refer to $\mathcal{T}(D)$ as the set of *broken triangles*.[1]

[1]In the literature on shortest path algorithms, these are referred to as "negative triangles."

III. Sparse Metric Repair Problem Statement

Given a perturbed or corrupted distance matrix, $D' \in \text{Sym}_n(\mathbb{R}_{\geq 0})$, the goal of *sparse metric repair* is to find a perturbation P so that $D = D' + P$ is metric, P is sparse as possible, and P is contained in a specified set S.

$$\begin{array}{ll} \text{minimize} & \|P\|_0 \\ \text{subject to} & D' + P \text{ is metric} \\ \text{and} & P \in S \subset \mathbb{R}^{n \times n} \end{array} \tag{1}$$

As we will see in the following, assumptions on the constraint set $\mathcal{S}$ play a large role in the types of algorithms, the complexity of the algorithms, and the types of guarantees that we can give for metric repair. We consider three specific cases:

Decrease Only Metric Repair (DOMR): In this case we assume that $P \preceq 0$, i.e. we are interested in only decreasing the distances of D' to ensure metricity.

Increase Only Metric Repair (IOMR): In this case we assume that $0 \preceq P$, i.e. we are interested in only increasing the distances of D' to ensure metricity.

General Metric Repair: We do not place any restrictions on P.

In general, there are many ways to repair D' to assure that $D' + P$ is a distance matrix. For example, as pointed out in the multidimensional scaling literature [9], if $c = \max_{i,j,k} D'_{ik} + D'_{jk} - D'_{ij}$ then $D' + c$ is automatically a metric. Also, note that uniqueness of P is not guaranteed. For example in a triangle with side lengths $1, 2, 7$, we can increase either of the two smaller sides, or decrease the larger side to make it metric. In the first example, the repair is far from sparse as it adds one value to all the distances and in the second example, there are two different repairs, one changes fewer distances than the other. Indeed, there are many cases in which we know that $D' = D + \Delta$ where Δ is sparse and so we seek to change or repair as few distances as possible so as to (approximately) recover D.

We note that this is a challenging problem. Even even if we know the support of Δ exactly, in general, we can not hope to recover D exactly from D'. Indeed, there could be multiple repairs P of the same sparsity level so that $D' - P$ is a distance matrix.

By the nature of the problem, the repair P only needs to be supported on edges appearing in any triangle in $\mathcal{T}(D')$, or in any triangle involving a distance in $\mathcal{T}(D')$. For any pair i, j, there are roughly $2n$ triangle inequalities involving i, j, so the support of P can be as large as $O(|\mathcal{T}(D')|n)$. Depending on the types of assumptions or restrictions on P, this bound can be a very loose upper bound.

In this paper, we focus on two main types of algorithms. In the next section, we describe algorithms for sparse DOMR, sparse IOMR, and the general case that are fundamentally combinatorial in nature and that aim to repair the metric directly. We also present convex relaxation methods based on minimizing the ℓ_1 norm, a technique that is well-established in the sparse analysis literature. Finally, we discuss the

case of random data and provide experiments contrasting combinatorial and convex approaches.

IV. Algorithms for Metric Repair

A. Decrease Only Metric Repair (DOMR)

We begin by considering the case when $P \leq 0$, so $\mathcal{S}$ is the subspace of all positive symmetric matrices. This is also known as the *Decrease Only Metric Repair* problem (DOMR). A few initial remarks are necessary. First, assume that triangle ijk is broken; i.e. $D'_{ij} \geq D'_{ik} + D'_{jk}$. Decreasing D'_{ik} or D'_{jk} will not fix this triangle inequality. Hence, we know that any distances $D'_{i,j}$ that occur on the left hand side of a broken triangle inequality necessarily are in the support of P.

Second, the decrease only metric repair problem is closely related to that of all pairs shortest path (APSP) on a graph and this relationship is easy to see. Because $D = D' + P$ is metric, by definition, every $D_{ij} \leq \min_k(D_{ik} + D_{jk})$. This has a natural interpretation in terms of shortest paths on a graph—the shortest path from i to j is given by the distance D_{ij} from i to j. The relationship between DOMR and APSP was first observed by Brickell et al. [7]. In fact they show that the problems are equivalent; a decrease only solution gives rise to an APSP solution and vice versa.

In particular, this implies that any algorithm for APSP can be reused for DOMR. There are many algorithms for solving APSP; to name a few, the Floyd-Warshall algorithm (presented in Algorithm 1), repeated use of single source shortest path methods such as Dijkstra's algorithm, and recursive methods based on matrix multiplication in the tropical semiring (see Williams and Williams [10] for a full discussion of the relationship amongst these problems). Brickell, et al. [7] also provide a primal-dual algorithm (similar to the method using Dijkstra's algorithm) for APSP. All of the above algorithms have a run time of $O(n^3)$ in the worst case. We will discuss the use of subcubic algorithms for APSP in broken triangle detection further in Section IV-F.

Algorithm 1: Floyd-Warshall for DOMR

```
Input: Corrupted n × n distance matrix D'
Result: Perturbation P
D̂ = D'
for k = 1 to n do
    for i = 1 to n do
        for j = 1 to i − 1 do
            if D̂_ij ≥ D̂_ik + D̂_kj then
                D̂_ij = D̂_ik + D̂_kj ;
        end
    end
end
P = D̂ − D'
```

To summarize Brickell, et al.'s observation:

Lemma 4.1: (**Brickell, et al. [7]**) Let $\widehat{D}$ be any solution to DOMR; i.e. $\widehat{D}$ is element-wise less than D'. In addition, let D^A be the all pairs shortest path solution for $K_n(D')$, the complete graph with edge weights given by D'. Then, $\hat{D}$ is element-wise smaller than D^A, $\hat{D} \preceq D^A$.

This lemma has several immediate implications for finding sparse solutions. Let the perturbation associated to D^A be $P^A := D^A - D'$. Consider a decomposition $D' = \widehat{D} + \Delta$ with $\widehat{D}$ metric and $\Delta \geq 0$. By Lemma 4.1, $\widehat{D} \preceq D^A \preceq D'$, so if $\Delta_{ij} = 0$, then $D^A_{ij} = D'_{ij}$ which implies $(P^A)_{ij} = 0$. In particular, this implies that $\text{supp}(P^A) \subset \text{supp}(\Delta)$. We summarize these observations for (sparse) DOMR in two corollaries.

Corollary 1.1: The perturbation $P^A := D^A - D$ is a sparsest possible decrease only metric repair solution.

Corollary 1.2: The lemma also immediately implies that D^A is the solution to

$$\text{argmin}_{D \preceq D'} \|D - D'\|_p$$

for any ℓ_p norm. In particular, if we set $p = 1$, we see that D^A is the minimal ℓ_1 norm solution.

The algorithmic implication of Corollary 1.1 is that the only distances that must be repaired (or, more accurately, adjusted) are those distances that were corrupted to begin with. In particular, this means that the only triangles that should be modified in the course of the algorithm are triangles that are already broken. The Floyd-Warshall algorithm presented in Algorithm 1 is particularly amenable to including prior information about broken triangles or distances. In particular, we only have to consider triangles ijk that are in $\mathcal{T}(D')$, or triangles that could potentially break if we decrease a distance D_{ij}; i.e. potentially breaking a triangle inequality of the form $D'_{il} \leq D'_{ij} + D'_{jl}$. (We refer to this extension as $\mathcal{T}'$ in the pseudo-code below.) In particular, this implies a $O(|\mathcal{T}(D')|n)$ algorithm for DOMR. In general, if our perturbation P is sparse, we expect that the number of involved triangles will be small as well. If we perturb k, distances, then we will have to consider $O(kn)$ triangles—far fewer from the n^3 required by the naive Floyd-Warshall algorithm. Our modified algorithm is given in Algorithm 2.

Algorithm 2: Floyd-Warshall with Prior Information

```
Input: Corrupted n × n distance matrix D', T(D')
       ordered lexicographically
Result: Perturbation P
D̂ = D'
T' extension of T; all lij for ijk ∈ T
for t = ijk in T' do
    if D̂_ij > D̂_ik + D̂_kj then  D̂_ij = D̂_ik + D̂_kj ;
end
P = D̂ − D'
```

Theorem 4.2: Given a corrupted distance matrix D' and a list $\mathcal{T}(D')$ of broken triangles, ordered lexicographically, Floyd-Warshall with Prior Information returns the sparsest repair P in time $O(|\mathcal{T}(D')|n)$.

B. Increase Only Metric Repair

Next, we consider the Increase Only Metric Repair problem (IOMR); that is, we assume a repair of the form

$0 \preceq P$. Though superficially similar to DOMR, IOMR is significantly more difficult and far fewer guarantees can be given.

Unlike the DOMR case, it is not possible to detect which distances were perturbed from a broken triangle inequality. Indeed, if $D_{ij} \geq D_{ik} + D_{jk}$, then either D_{ik} or D_{jk} could have been initially perturbed (decreasing D_ij would not have broken the triangle inequality). In addition, it is easy to see by an explicit example that the ℓ_1 minimizing IOMR solution does not give any sparsity guarantees. Consider our previous example: a triangle with side lengths $1, 2, 7$, we can increase either of the two smaller sides with a total increase of 4 or decrease the larger side by 4 to make it metric. There are infinite solutions with a total ℓ_1 norm of 4 but only three solution which changes only one distance; i.e., only a few sparse solutions out of multiple solutions with minimal ℓ_1 norm.

We are, however, encouraged by the success of Floyd-Warshall in the DOMR problem and we provide Algorithm 3 that updates D_{ik} with $D_{ij} - D_{jk}$ whenever ijk is broken.

Algorithm 3: Increase Only Metric Repair

```
Input: Corrupted n × n distance matrix D'
Result: Perturbation P
D̂ = D
for k = 1 to n do
    for i = 1 to n do
        for j = 1 to i − 1 do
            if D̂_ij > D̂_ik + D̂_kj then
                D̂_ik = D̂_ij − D̂_kj
            end
        end
    end
end
P = D̂ − D'
```

Lemma 4.3: Given a corrupted distance matrix D', Algorithm 3 for IOMR returns a positive perturbation P so that $D' + P$ is metric in time $O(n^3)$.

Proof: At each step of the inner loop, a triangle is fixed. It suffices to show that these steps are not destructive; i.e. after the end of iteration k of the outer loop, by increasing $\widehat{D}_{ik}$, no triangle inequality ikl, for any triangle with $l < k$, breaks. If $l > k$, we will visit this triangle in the future and repair it.

Choose j to be the largest index so that we update triangle ijk and assume updating $\widehat{D}_{ik}$ breaks triangle ikl as a result. At the end of the inner most loop, $\widehat{D}_{ik} = \widehat{D}_{ij} - \widehat{D}_{kj}$. This implies,

$$\widehat{D}_{ik} > \widehat{D}_{il} + \widehat{D}_{lk} \Rightarrow \widehat{D}_{ij} - \widehat{D}_{kj} > \widehat{D}_{il} + \widehat{D}_{lk}$$
$$\Rightarrow \widehat{D}_{ij} > \widehat{D}_{il} + \widehat{D}_{lk} + \widehat{D}_{kj}.$$

Since $l < k$, triangle ijl is fixed, and so $\widehat{D}_{ij} \leq \widehat{D}_{il} + \widehat{D}_{lj}$. Plugging this in as an upper bound for $\widehat{D}_{ij}$, we see that

$$\widehat{D}_{il} + \widehat{D}_{lj} \geq \widehat{D}_{ij} > \widehat{D}_{il} + \widehat{D}_{lk} + \widehat{D}_{kj},$$

which implies $\widehat{D}_{lj} > \widehat{D}_{lk} + \widehat{D}_{kj}$. In particular, the above manipulation shows if ikl is broken, then ljk must be as well.

Now, if $j < l < i$ or $l < j < i$, then we have already visited triangle ljk (on this iteration of the outside loop) and fixed it. In addition, updating $\widehat{D}_{ik}$ will not affect triangle ljk, or its equivalent jlk in the latter case. This implies that triangle ikl cannot have been broken. So, we can assume that $l > i > j$, and the algorithm has fixed triangle ljk by setting $\widehat{D}_{lk} = \widehat{D}_{lj} - \widehat{D}_{jk}$. In this case,

$$\widehat{D}_{il} + \widehat{D}_{lk} = \widehat{D}_{il} + \widehat{D}_{lj} - \widehat{D}_{jk} \geq \widehat{D}_{ij} - \widehat{D}_{jk} = \widehat{D}_{ik}$$

The last inequality holds since $l < k$, ijl is not broken. Thus, we repair triangle ikl.

■

In general, we do not guarantee that Algorithm 3 provides the sparsest possible perturbation. Indeed, if we increase a distance $\widehat{D}_{ik}$ that was not initally perturbed, many triangles of the form $\widehat{D}_{ik} \leq \widehat{D}_{il} + \widehat{D}_{kl}$ could break as a result and we update many distances $\widehat{D}_{il}$. As demonstrated in Section V, this algorithm, nevertheless, tends to provide sparse solutions on average. As in Algorithm 1, we can reduce the runtime from $O(n^3)$ to to $O(|\mathcal{T}(D')|n)$ by using prior information. We provide Algorithm 3 as a natural counterpoint to Algorithm 1 as it solves the increase only metric repair problem.

Now we turn to a method for IOMR that utilizes prior information. Assume that $D' = D + \Delta$ and we have access to an oracle Q that knows precisely the support of Δ. In theory, we could use a linear program to find the exact values of P using Q. Instead we use a different method that tracks bounds on the distances provided by the triangle inequalities.

The oracle Q is a 0-1 matrix with $Q_{ij} = 1$ if i, j is in P and is 0 otherwise. Our goal is to find an IOMR solution $D' \preceq \widehat{D}$ so that $D' - D$ is supported only on entries where $Q_{ij} = 1$. Algorithm 4 solves this problem, by using Q to turn the IOMR problem into a DOMR problem and then running an APSP routine. The algorithm first finds an upper bound U_{ij} for each D_{ij} by computing

$$\min_{Q_{ik}=0, Q_{jk}=0} (D_{ik} + D_{jk}).$$

It then returns the APSP matrix for U via the Floyd-Warshall algorithm. Note that each distance is replaced with its upper bound, thus guaranteeing that we only increase the distances. Lemma 4.4 proves the validity of this algorithm.

In general we cannot expect to know Q, but there are easy heuristic algorithms for constructing a close approximation. For example:

- We can scan over all broken triangles and set $Q(i, j) = 1$ for distances that occur the most on the right hand side of a triangle inequality. Alternatively, we can scan over all broken triangles and take a subset of distances, so that every triangle contains at least one of these distances. This is similar to the standard set-cover problem which has many fast heuristics.

- Finally, we can run an APSP method on D' and look at the edges that shortest paths tend to be routed through the most. It is very reasonable to assume that such edges have been decreased.

We will demonstrate the efficacy of the first method in Section V.

Lemma 4.4: Assume that the corrupted distance matrix is of the form $D' = D + P$, with D metric and that we are given access to an oracle Q where $Q_{ij} = 1$ iff $P_{ij} \neq 0$. In addition, assume that for each distance ij with $Q_{ij} = 1$, we can find a k so that $Q_{ik} = 0$ and $Q_{jk} = 0$; i.e., each distance ij is contained in at least one triangle ijk that is not broken. Then Algorithm 4 will return an IOMR solution supported only on entries in P and does so in time $O(n^3)$.

Proof: It suffices to show that $\hat{U}_{ij} = D_{ij}$ whenever $Q_{ij} = 0$ by the end of the algorithm. We claim that after the Upper-Bound step and before the APSP step, $D'_{ij} \leq D_{ij} \leq U_{ij}$ for all ij; i.e. U is an upper bound for D. Indeed, if $Q_{ik} = Q_{jk} = 0$, then $D'_{ik} = D_{ik}, D'_{jk} = D_{jk}$. And, by the algorithm,

$$U_{ij} = \min_{k:Q_{ik}=Q_{jk}=0}(D'_{ik}+D'_{jk}) = \min_{k:Q_{ik}=Q_{jk}=0}(D_{ik}+D_{jk}).$$

Since each term in the minimum is an upper bound for D_{ij}, by metricity of D, U_{ij} is also an upper bound for D_{ij}. Let us focus on the case when $Q_{ij} = 0$, i.e. edge ij has not been perturbed, then $U_{ij} = D_{ij} = D'_{ij}$. In this case every triangle ijk in U is correct by the end of the upper bound step. That is,

$$U_{ij} = D_{ij} \leq D_{ik} + D_{jk} \leq U_{ik} + U_{jk}.$$

In particular, we see that this implies that the shortest path from i to j in $K_n(U)$ is the edge ij. Hence after the APSP step, $\hat{U}_{ij} = U_{ij} = D_{ij}$.

■

As with all previous algorithms, if we know the set of broken triangles of U a priori, we can accelerate Algorithm 4 to an $O(|Q|n)$ algorithm.

C. General Metric Repair

Using the ideas in the previous subsections, we present an algorithm for metric repair in the general case. Our algorithm combines the ideas of the update steps for both DOMR and IOMR with the idea of finding an oracle. Essentially, the algorithm first uses the counting heuristic described in Section IV-B to build two dictionaries, one which tracks the number of times that each distance occurs on the right or the left in a broken triangle. This information is then used to decide what distance to fix in a broken triangle. Intuitively, if D'_{ij} occurs on the left of a broken triangle inequality often, this implies that we should decrease it.

In general, the heuristic algorithm is not guaranteed to succeed; i.e., the output may not be a fixed distance matrix. In addition, no guarantee is given for finding the sparsest solution; the heuristic constraints, however, certainly encourages sparsity. Again, the algorithm can be modified to loop over all triangles in $\mathcal{T}(D')$ and triangles containing a distance from $\mathcal{T}'(D')$.

Algorithm 4: Oracle Based Increase Only Metric Repair

Input: Corrupted $n \times n$ distance matrix D', Q oracle
Result: Perturbation P
Let U be an $n \times n$ symmetric matrix initialized to ∞
// Upper-Bound step
for $k = 1$ **to** n **do**
  for $i = 1$ **to** n **do**
    for $j = 1$ **to** $i - 1$ **do**
      if $D'_{ij} \leq D'_{ik} + D'_{kj}$ **then**
        if $Q_{ik} = 0$ ***and*** $Q_{jk} = 0$ **then**
          $U_{ij} = \min(U_{ij}, D'_{ik} + D'_{jk})$
        end
      end
    end
  end
end
// APSP-step
$\widehat{U} = U$
for $k = 1$ **to** n **do**
  for $i = 1$ **to** n **do**
    for $j = 1$ **to** $i - 1$ **do**
      if $\widehat{U}_{ij} \geq \widehat{U}_{ik} + \widehat{U}_{jk}$ **then**
        $\widehat{U}_{ij} = \min(\widehat{U}_{ij}, \widehat{U}_{ik} + \widehat{U}_{jk})$
      end
    end
  end
end
$P = \widehat{U} - D'$

D. Random data

Algorithm 5 applies in situations where we do not have any prior knowledge of the sign of our perturbation. Although it is a heuristic, it is instructive to consider distances that are drawn at random specified distributions and examine the extent to which these random matrices fail to adhere to metricity so that when we demonstrate how the heuristic performs on this type of data in Section V, we have an understanding of its expected performance.

We specifically consider the model where D' is a random symmetric matrix with entries independently drawn from a distribution.

Lemma 4.5: The expected proportion of broken triangles in D' is $1/6$ and $1/4$ when the entries of D' are drawn from a uniform distribution Unif$([0, 1])$, or an exponential distribution Exp(λ).

In particular, this implies that if n is sufficiently large, almost every distance will be involved in a broken triangle! Thus it seems unlikely to find an exceptionally sparse perturbation in all cases that will fix the metric.

E. Convex relaxation methods

Finally, we finish with convex optimization methods (or, more colloquially, ℓ_1 minimization). Using the standard paradigms from compressed sensingy, we relax the ℓ_0 constraint on P to a ℓ_1 constraint:

Algorithm 5: Heuristic for General Metric Repair

Input: Corrupted $n \times n$ distance matrix D'
Result: Perturbation P
$\widehat{D} = D$
Let l, r be $n \times n$ symmetric all-zero matrices.
for $k = 1$ **to** n **do**
  for $i = 1$ **to** n **do**
    for $j = 1$ **to** $i - 1$ **do**
      if $\widehat{D}_{ij} \geq \widehat{D}_{ik} + \widehat{D}_{kj}$ **then**
        $\mathrm{l}_{ij} = \mathrm{l}_{ij} + 1$ $\mathrm{r}_{ik} = \mathrm{r}_{ik} + 1$ $\mathrm{r}_{jk} = \mathrm{r}_{jk} + 1$
      end
    end
  end
end
for $k = 1$ **to** n **do**
  for $i = 1$ **to** n **do**
    for $j = 1$ **to** $i - 1$ **do**
      if $\widehat{D}_{ij} \geq \widehat{D}_{ik} + \widehat{D}_{kj}$ **then**
        if $\mathrm{l}_{ij} > \max(\mathrm{r}_{ik}, \mathrm{r}_{jk})$ **then**
          $\widehat{D}_{ij} = \widehat{D}_{ik} + \widehat{D}_{kj}$;
        else if $\mathrm{r}_{ik} > \mathrm{r}_{jk}$ **then**
          $\widehat{D}_{ik} = \widehat{D}_{ij} - \widehat{D}_{jk}$;
        else
          $\widehat{D}_{jk} = \widehat{D}_{ij} - \widehat{D}_{ik}$
        end
      end
    end
  end
end

Algorithm 6: Iteratively Reweighted ℓ_1 minimization

Input: Corrupted $n \times n$ distance matrix D, iters, tolerance parameter ϵ
Result: Fixed distance matrix $\widehat{D}$
w^1, $n \times n$ weight matrix with every entry initially 1.
for $t = 1$ **to** *iters* **do**
  P^t solution to following optimization problem

$$\begin{aligned} \text{minimize} \quad & \sum_{1 \leq i,j \leq n} w_{ij} |P_{ij}| \\ \text{subject to} \quad & D' + P \text{ is metric}, P \in S \subset \mathbb{R}^{n \times n} \end{aligned}$$

  $w_{ij}^{t+1} = \frac{1}{P_{ij}^{t-1} + \epsilon}$
end

$$\begin{aligned} \text{minimize} \quad & \|P\|_1 \\ \text{subject to} \quad & D' + P \text{ is metric}, P \in S \subset \mathbb{R}^{n \times n} \end{aligned} \tag{2}$$

We can also consider the metric constraints as polyhedral constraints on the set of symmetric $n \times n$ matrices (the resulting polyhedron is known as the metric cone, see for example [11]). So our sparse metric repair problem is asking for the sparsest P so that $D' + P$ lies in this polyhedral cone. A commom technique used for finding the sparsest solution for a set of linear constraints is reweighted ℓ_1 schemes [12].

In general reweighted schemes tend to find the sparsest solutions, however it can take many iterations for it to converge. Both algorithms implemented using standard solvers like CVX are computationally much more expensive compared to the discrete algorithms given above. In particular the best algorithms for checking that $D' + P$ is metric are essentially $O(n^3)$ (see below), so the convex algorithms have complexity at least as bad as the complexity of the discrete methods.

F. Identifying Broken Triangles

All of our algorithms have cubic run time and, with an input list of broken triangles, are much more efficient. To speed up these algorithms, it is sufficient to identify the broken (or negative) triangles efficiently. Indeed, this phenomenon is not a surprise as Williams and Williams [10] show that a number of problems are all equivalent under subcubic reductions, including negative triangle detection, APSP, and detecting metricity. These reductions show that an $O(n^{3-\epsilon_1})$ algorithm for one problem would yield an $O(n^{3-\epsilon_2})$ algorithm for another. As of 2017, there are, however, no truly subcubic algorithms for APSP. Chan and Williams [13] show that APSP on n-node weighted graphs can be solved in $O(n^3/2^{\Omega(\sqrt{\log n})})$ time deterministically. There are subcubic algorithms for graphs with integer weights and for unweighted graphs (for a comprehensive survey, see [14]). This *might* apply in some, but not all, data applications. Finally, we note that Peres, et al. [15] show that on a random data model, APSP is a subcubic (even $O(n^2)$ algorithm) and we use this for inspiration in our experimental analysis.

V. Experimental analysis

Because the algorithms for DOMR are guaranteed to return the sparsest solution (see Section IV-A), we focus on a comparison amongst our combinatorial algorithms, namely IOMR (Algorithm 3), Oracle IOMR (Algorithm 4), Heuristic for General Metric Repair (Algorithm 5) and the convex methods of ℓ_1 minimization (i.e., the algorithmic formulation given by Equation 2), and iteratively reweighted ℓ_1 minimzation which we also denote by IRℓ_1(Algorithm 6).

In general, there are many different factors and models we can consider that impact the initial number of broken triangles and the resulting output sparsity. We could also analyze many aspects of the output, for example, the uniqueness of the sparsest perturbation, or how our solutions compare in magnitude. All of these cases are interesting to consider, but, for succinctness, we focus on sparsity of the output perturbation and the time it took the algorithm to run.

First, consider the case where we observe a broken metric $D' = D + \Delta$, with D metric and Δ a sparse perturbation. We used the various algorithms to compute a perturbation P so that $D' + P$ was again metric. In the top row of Figure 1, D is assumed to be the distance matrix of 50 points in $\mathbb{R}^2$. The x-axis gives the sparsity of Δ as a fraction of the 1225

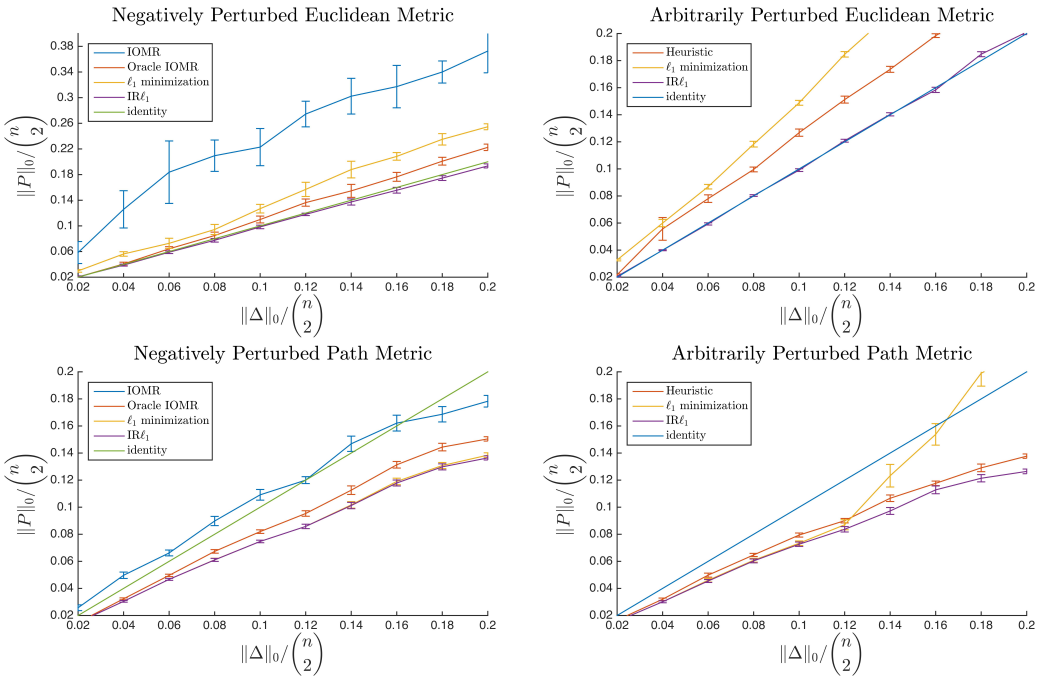

Fig. 1: Experiments on input vs output sparsity for the algorithms in this paper.

distances, and the y-axis is the sparsity level of P. In the left hand plot, Δ is assumed to be negative with values on the support uniformly chosen from $[-\|D\|_\infty/8, 0]$; on the right the values are uniformly chosen from $[-\|D\|_\infty/8, \|D\|_\infty/8]$. These values were chosen to be not too large, to assist finding D's so that $0 \preceq D + \Delta$. In the bottom row, D is the path metric of an Erdös-Renyi random graph on 50 vertices with $p = 2\log 50/50$. Since the diameter of such Erdös-Renyi graphs is less than 3 with high probability, the support of Δ was set to -1 in the negative perturbation case, and arbitrarily chosen from $\{-1, 1\}$ in the arbitrary perturbation case. For each sparsity level, we averaged the sparsity of the output perturbation over ten trials. Finally, we also plot the the identity line as we expect that the sparsity of the output P should not be too much greater than the sparsity of the input perturbation Δ.

All algorithms were implemented in Matlab on a 2014 Macbook Pro running a 2.2 GHz Intel Core i7. We used the GNU Linear Programming Kit [16] for the ℓ_1 minimization based techinques.

In all cases iteratively reweighted ℓ_1 did the best, often finding a solution as sparse as the input perturbation or even sparser. When $\Delta \preceq 0$, IOMROracle tends to do almost as well as IRℓ_1, even matching it for low sparsity levels. In the Euclidean case, we see that IOMROracle and our Heuristic do significantly better than ℓ_1. In the path case, ℓ_1 minization and IRℓ_1 match for low sparsity levels, and both are slightly better than IOMROracle. We expect IOMR to be worse than IOMROracle since it is fixing distances in a predetermined order rather than a data-dependent fashion. It is surprising, however, that it does not seem to produce perturbations with sparsity worse than twice those of IOMROracle, so we conjecture that IOMR is potentially a 2-approximation algorithm for IOMR. For IOMROracle, we counted the broken triangles a distance occurred to decide whether we should fix it (similar to the method in the Heuristic algorithm). As a result, in several cases, IOMROracle and Heuristic both returned perturbations P such that $D' + P$ had broken triangles. However, in each such case, at worst 20 triangles (around 1.6% of all triangles) were broken—a significant improvement over the up to 60% that were initially broken.

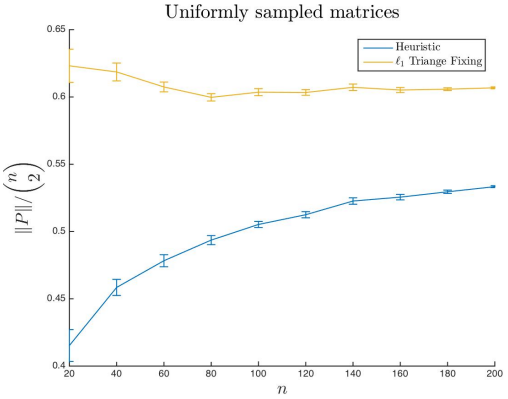

Fig. 2: Output sparsity of metrics with entries uniformly sampled from $[0, 1]$.

The path metric case has several interesting attributes. Firstly, there is a strange divergence of ℓ_1 minimization from IRℓ_1 in the arbitrary perturbation case for large sparsity cases. Secondly, in all the examples we ran, the both ℓ_1 minimization and IRℓ_1 produced perturbations that were integer vectors! In addition the Heuristic almost always repaired the metric. The support of the solutions from the

Heuristic and ℓ_1 algorithms, differed from each other and Δ.

We also considered the case when D' was an arbitrary symmetric matrix whose entries were drawn uniformly from $[0, 1]$. We compared the Heuristic method to the ℓ_1 Triangle Fixing method given in [7]. Figure 2 shows the recovered sparsity as we vary n. As described in Lemma 4.5, we expect around 16% of triangles to be broken, and almost all distances to be involved in a broken triangle. So, repairing or affecting only 50% of distances is a significant improvement over adjusting all of them.

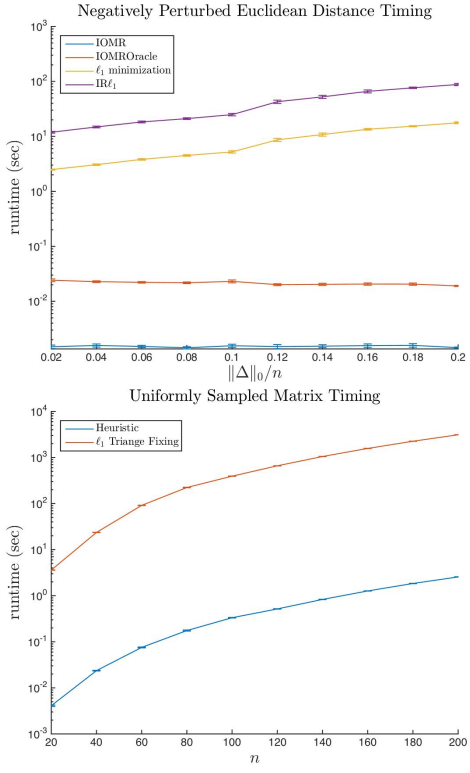

Fig. 3: Runtime comparison of the algorithms.

The real advantage of the combinatorial algorithms over the convex methods is in their runtime; often several orders of magnitude. Figure 3 shows the difference in runtime for the arbitrary perturbation Euclidean case. We also show the difference in timing for the Heuristic method vs Triangle Fixing. In general we found Triangle Fixing difficult to choose parameters for, and we also found that the number of passes over all triangles it required seemed to grow with n, unlike our Heuristic method which requires two. We realize that more specialized LP solvers could lead to large improvements and hope to address this in the future.

VI. Conclusions

Our experiments show that the convex relaxation approaches to these problems have interesting empirical performance (depending on the input sparsity level and instance type). In addition, we can easily construct simple examples that have multiple (even infinite) solutions, all with the same minimal ℓ_1 norm. These two observations suggest a deeper study of convex relaxation based methods. We emphasize that this is in contrast with the efficacy of convex relaxation for more traditional sparse problems.

Many graph and network flow problems, such as Single Source Shortest Path, can be expressed as linear programs. Studying the dual of these programs has been fruitful in developing efficient combinatorial algorithms. In our work, we propose several different combinatorial algorithms and it would be interesting to ascertain what dual problem these algorithms solve. In particular it is likely that the IOMR algorithm is likely to correspond to the dual to a network flow problem extending the correspondence Brickell et al. [7] observe for the DOMR case.

Acknowledgment

We would like to thank Sergey Fomin and Seth Pettie for several useful discussions about this problem and its connections to all pairs shortest path algorithms and tropical matrix multiplication.

References

[1] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell, "Distance Metric Learning , with Application to Clustering with Side-Information," *Advances in Neural Information Processing Systems*, vol. 15, pp. 505–512, 2002.

[2] Y. Chen, M. R. Gupta, and B. Recht, "Learning kernels from indefinite similarities," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 145–152.

[3] P. Indyk, "Sublinear time algorithms for metric space problems," in *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, ser. STOC '99. New York, NY, USA: ACM, 1999, pp. 428–434. [Online]. Available: http://doi.acm.org/10.1145/301250.301366

[4] J. Anfinsen, "Making substitution matrices metric," Master's thesis, Institutt for datateknikk og informasjonsvitenskap, 2005.

[5] J. Laub, K.-R. Müller, F. A. Wichmann, and J. H. Macke, "Inducing metric violations in human similarity judgements," in *Advances in neural information processing systems*, 2007, pp. 777–784.

[6] S. Baraty, D. Simovici, and C. Zara, "The impact of triangular inequality violations on medoid-based clustering," *Foundations of Intelligent Systems*, pp. 280–289, 2011.

[7] J. Brickell, I. S. Dhillon, S. Sra, and J. A. Tropp, "The metric nearness problem," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 1, pp. 375–396, 2008.

[8] A. Biswas and D. W. Jacobs, "An efficient algorithm for learning distances that obey the triangle inequality." in *BMVC*, 2015, pp. 10–1.

[9] J. C. Gower and P. Legendre, "Metric and euclidean properties of dissimilarity coefficients," *Journal of Classification*, no. 3, pp. 5–48, 1986.

[10] V. V. Williams and R. Williams, "Subcubic equivalences between path, matrix and triangle problems," in *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, Oct. 2010, pp. 645–654.

[11] M. M. Deza and E. Deza, "Encyclopedia of distances," in *Encyclopedia of Distances*. Springer, 2009, pp. 1–583.

[12] D. Needell, "Noisy signal recovery via iterative reweighted l1-minimization," in *Signals, Systems and Computers, 2009 Conference Record of the Forty-Third Asilomar Conference on*. IEEE, 2009, pp. 113–117.

[13] T. M. Chan and R. Williams, "Deterministic APSP, Orthogonal Vectors, and More: Quickly Derandomizing Razborov-Smolensky," *SIAM ACM Symposium on Discrete Algorithms (SODA'16)*, pp. 1246–1255, 2016.

[14] A. Madkour, W. G. Aref, F. U. Rehman, M. A. Rahman, and S. Basalamah, "A survey of shortest-path algorithms," *arXiv preprint arXiv:1705.02044*, 2017.

[15] Y. Peres, D. Sotnikov, B. Sudakov, and U. Zwick, "All-pairs shortest paths in $O(n^2)$ time with high probability," in *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science, FOCS*, 2010, pp. 663–672.

[16] "Gnu linear programming kit," https://www.gnu.org/software/glpk.