Triangle Fixing Algorithms for the Metric Nearness Problem

Inderjit S. Dhillon **Suvrit Sra**
Computer Sciences
The University of Texas at Austin
Austin, TX, 78712

Joel A. Tropp
ICES
The University of Texas at Austin
Austin, TX, 78712.

September 10, 2004

Technical Report # TR-04-22
Department of Computer Sciences
University of Texas at Austin
Austin, TX, 78712

Abstract

Numerous problems in machine learning, data mining, databases and statistics involve pairwise dissimilarities amongst a set of objects. These dissimilarities can be represented as edge weights in a complete graph with the objects as the vertices. Often one desires the dissimilarities to satisfy the properties of a metric—especially the triangle inequality. Applications where metric data is important include clustering, metric based indexing, classification, query processing, and approximation algorithms. In this paper we present algorithms for solving the *Metric Nearness Problem:* Given a non-metric graph (dissimilarity matrix), find the "nearest" metric graph whose edge weights satisfy the triangle inequalities. This paper presents algorithms that exploit the innate structure of the problem for solving it efficiently for nearness in ℓ_p norms. Empirically, the algorithms have time and storage requirements linear in the number of triangle constraints. The methods are easily parallelizable enabling the solution of large problems.

1 Introduction

The Metric Nearness (MN) problem, which seeks the "nearest" metric graph to a given input graph, was introduced by Dhillon et al. [2003]. The problm can be modeled as a general convex optimization problem for an important class of nearness measures (the ℓ_p norms with $1 < p < \infty$), and as linear programming for ℓ_1 and ℓ_∞ based error measures. Other than resorting to standard convex optimization softare, Dhillon et al. [2003] did not provide explicit algorithms for solving the problem, though they conjectured the existence of efficient algorithms that could take advantage of the structure of the problem. In this paper we exploit the structure of the problem to give generic triangle fixing algorithms[1] to enforce triangle inequality on the input graph. Our approach, which makes use of specific convex programming techniques, is general enough to encompass a wide variety of nearness measures. We make use of *row action* methods that proceed by optimizing the objective function subject to one constraint at a time, with the inclusion of certain correction terms.

[1]Triangle fixing algorithms proceed by enforcing triangle *equality* for each violated triangle in the input graph. The way equality is enforced is governed by the particular nearness measure employed.

The beauty of such methods is that, by an appropriate choice of the correction term (determined by the nearness measure), the method can be shown to converge to the globally optimal solution.

The MN problem is formally described in Section 3 along with the notation and concepts necessary for understanding the algorithms that form the subject of Section 4. Preliminary experimental results illustrating the performance of our algorithms follow in Section 5. Some discussion about the problem and the algorithms along with a mention of interesting connections to the All Pairs Shortest Paths (APSP) problem can be found in Section 6. Since the MN problem is new and as yet underexplored, we suggest potential future work in Section 7. Important, but somewhat ancillary details are pushed into an appendix trailing this report.

2 Background Material

For solving the MN problem we make use of two well known optimization methods: (i) Bregman's method for inequality constrained problems [Censor and Zenios, 1997, §6.3], and (ii) Dykstra's method for [Deutsch, 2001, Chapter 9]. We could have employed Bregman's method alone for our purpose but to expose other related methods we chose to also include Dykstra's method. Both Bregman's and Dykstra's methods are row action methods as they proceed by enforcing one constraint (one row) of the constraint matrix at a time. Bregman's method is more general in the functions that it can minimize subject to linear inequality constraints whereas Dykstra's method minimizes a quadratic objective allowing the constraint sets to be arbitrary convex sets. We emphasize though that these two methods, used by themselves would be prohibitive, implemented as triangle fixing procedures they become viable.

There also exist methods in the literature [Bauschke and Lewis, Bregman et al., 1999] that allow one to generalize either Bregman's algorithm (or Dykstra's algorithm) so that the method is applicable to a wide range of objective functions along with arbitrary convex constraint sets. We present procedures that efficiently implement Bregman's and Dykstra's methods to take advantange of the structure of the constraint matrix thereby solving the nearness problem efficiently.

Before we proceed to the problem or the algorithms we need to establish some background. The material presented herein can be found in much greater detail in the book by Censor and Zenios [1997]. More details about Dykstra's algorithm and a proof of its convergence can be found in the book by Deutsch [2001].

2.1 Bregman Functions and Generalized Projections

Let S be a nonempty, open convex set such that its closure $\bar{S}$ is contained in the domain of a strictly convex function $\varphi : \Lambda \subseteq \mathbb{R}^n \to \mathbb{R}$. Also assume that $\varphi(x)$ has continuous first partial derivatives for all $x \in S$. From such a function φ we can construct a function $B_\varphi : \bar{S} \times S \to \mathbb{R}$:

$$B_\varphi(x, y) = B_\varphi(x\|y) := \varphi(x) - \varphi(y) - \langle \nabla\varphi(y), x - y\rangle. \tag{2.1}$$

This function $B_\varphi(x\|y)$ is called the Bregman divergence between x and y [Censor and Zenios, 1997] or the E-function [Hestenes, 1975]. It is nonnegative, convex in the first argument and zero if and only if $x = y$. Under certain conditions (strictly convexity, twice continuously differentiability and co-finiteness when $\varphi : \mathbb{R}^n \to \mathbb{R}$) φ is called a Bregman function [See Censor and Zenios, 1997, Def. 2.1.1]. For our purposes, we will always work with functions that are Bregman functions and following Censor and Zenios [1997] we denote the family of Bregman functions $\mathscr{B}(S)$, where S is as described above.

Since we aim to solve nearness problems where the measure of nearness is a Bregman divergence, we must first look at Bregman projections (or simply projections where the distance is measured by a Bregman divergence).

Definition 2.1 (Bregman projection [Censor and Zenios, 1997, Def 2.1.2]). Given $\Omega \subseteq \mathbb{R}^n$, $\varphi \in \mathscr{B}(S)$ and $y \in S$, a point $x^* \in \Omega \cap \bar{S}$ for which

$$P_\Omega^\varphi(y) \equiv \min_{z \in \Omega \cap \bar{S}} B_\varphi(z\|y) = B_\varphi(x^*\|y),$$

is called the Bregman projection of the point y onto the set Ω.

In the absence of a superscript, we adopt the convention that $P_\Omega(\boldsymbol{y})$ refers to the orthogonal projection of $\boldsymbol{y}$ onto Ω.

Lemma 2.2 ([Censor and Zenios, 1997, Lemma 2.1.2]). *If $\varphi \in \mathscr{B}(S)$, then for any closed convex set $\Omega \subseteq \mathbb{R}^n$, such that the intersection of Ω with the closure of S is nonempty, i.e., $\Omega \cap \bar{S} \neq \emptyset$, and for any $y \in S$, there exists a unique Bregman projection $x^* \equiv P_\Omega^\varphi(y)$.*

We now state the most important lemma for our algorithms. This lemma shows how to find the Bregman projection of a given point onto a hyperplane (or onto a half-space).

Lemma 2.3 (Projection onto Hyperplane [Censor and Zenios, 1997, Lemma 2.2.1]). *Let $\varphi \in \mathscr{B}(S)$, $H = \{\boldsymbol{x} | \langle \boldsymbol{a}, \boldsymbol{x} \rangle = b\}$, and assume that for every $y \in S$, the projection $P_H^\varphi(y) \in S$. Then for any given $y \in S$, the system*

$$\begin{aligned} \nabla\varphi(x^*) &= \nabla\varphi(\boldsymbol{y}) + \mu\boldsymbol{a} \\ \langle \boldsymbol{a}, x^* \rangle &= b, \end{aligned} \tag{2.2}$$

uniquely determines the point x^* that is the Bregman projection of y onto H. For a fixed H, the system also determines the parameter μ.

Note that the KKT optimality conditions mandate,

$$\begin{aligned} \mu(b - \langle \boldsymbol{a}, \boldsymbol{y} \rangle) > 0, &\quad \text{if } \boldsymbol{y} \notin H, \\ \mu = 0, &\quad \text{if } \boldsymbol{y} \in H. \end{aligned} \tag{2.3}$$

If the point $\boldsymbol{y}$ already satisfies $\boldsymbol{a}^T\boldsymbol{y} \leq b$, then $\mu = 0$. This observation allows us to use the same lemmas for half-spaces as for hyperplanes. As a simple corollary, let us derive the orthogonal projection onto a hyperplane.

Corollary 2.4 (Orthogonal Projection). *Let $\varphi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}_0 - \boldsymbol{x}\|^2$, where $\boldsymbol{x} \in \mathbb{R}^n$. Let $H = \{\boldsymbol{x} | \langle \boldsymbol{a}, \boldsymbol{x} \rangle = b\}$. Then $P_H(\boldsymbol{x})$ is given by,*

$$P_H(\boldsymbol{x}) = \boldsymbol{x} + \frac{1}{\|\boldsymbol{a}\|^2}[b - \langle \boldsymbol{a}, \boldsymbol{x} \rangle]^+ \boldsymbol{a}, \tag{2.4}$$

where $[\alpha]^+ = \alpha$ if $\alpha \geq 0$ and 0 otherwise.

Proof. From Lemma 2.3 and (2.3) we know that $\boldsymbol{x}^* = P_H(\boldsymbol{x})$ must satisfy

$$\begin{aligned} \boldsymbol{x}^* - \boldsymbol{x}_0 &= \boldsymbol{x} - \boldsymbol{x}_0 + \mu\boldsymbol{a} \\ \langle \boldsymbol{a}, \boldsymbol{x}^* \rangle &= b. \end{aligned}$$

Solving for μ and substituting, we immediately obtain (2.4). □

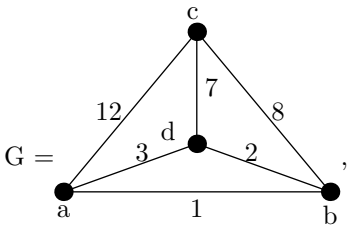
3 Metric Nearness

Let us begin with some basic definitions.

Definition 3.1 (Triangle Fixing). The procedure of enforcing triangle *equality* for one violating triangle (when an edge in the triangle has weight greater than the sum of the other two) is called a *triangle fixing* procedure. This procedure involves optimally adjusting the weights of the edges of the given triangle where the notion of optimality depends upon the measure of nearness employed.

Definition 3.2 (Metric Graph). We define a *metric graph* to be a complete graph G' whose edge weights satisfy the triangle inequality. If G' is a metric graph then $e_{ik} \leq e_{ij} + e_{jk}$ for every triple of distinct vertices (i, j, k). We denote the set of all n vertex metric graphs by $\mathscr{M}_n$.

We use G to denote the input graph that is assumed to be a complete undirected edge weighted graph on n vertices. Assuming the edge weights represent inter-vertex dissimilarities, we use the matrix $\boldsymbol{D}$ to encode this information. That is, $d_{ij} = d_{ji}$ gives the dissimilarity between the vertices i and j. Further assume that self dissimilarity is zero, i.e., $d_{ii} = 0$. Thus, $\boldsymbol{D}$ is a symmetric, nonnegative matrix with zero diagonal. We call the matrix corresponding to a metric graph a *distance matrix* and we denote it as $\boldsymbol{M}$. We associate a vector $\boldsymbol{d}$ corresponding to each dissimilarity matrix $\boldsymbol{D}$ and we obtain $\boldsymbol{d}$ by stacking the strict upper triangle of $\boldsymbol{D}$ taken row-wise. Figure 1 illustrates a graph, its corresponding dissimilarity matrix and the dissimilarity vector.



$$\boldsymbol{D} = \begin{bmatrix} 0 & 1 & 12 & 3 \\ 1 & 0 & 8 & 2 \\ 12 & 8 & 0 & 7 \\ 3 & 2 & 7 & 0 \end{bmatrix}, \quad \boldsymbol{d} = [1, 12, 3, 8, 2, 7]$$

Figure 1: Input Graph and its representations

We use the convention that the triple (i, j, k) corresponds to the triangle inequality

$$d_{ij} \leq d_{jk} + d_{ki}.$$

Similarly (j, k, i) and (k, i, j) correspond to $d_{jk} \leq d_{ki} + d_{ij}$ and $d_{ki} \leq d_{ij} + d_{jk}$; thus there are three ordered triples for each triangle in the graph. The $\binom{n}{2}$ edges of G result in $3\binom{n}{3}$ such ordered triples. The set of all these triples is denoted by T and we define

$$T = \{(i, j, k), (j, k, i), (k, i, j) : 1 \leq i < j < k \leq n\}.$$

Our convention allows us to index the set of triples T. This indexing proves useful for efficient implementation of a triangle fixing algorithm. For example, for a 4-vertex complete graph the set of triples (triangle inequalities) can be listed as

$$T = \{(1, 2, 3), (2, 3, 1), (3, 1, 2), \ldots, (2, 3, 4), (3, 4, 2), (4, 2, 3)\}. \tag{3.1}$$

Each triple in set T can also be represented by a vector that describes which inequality is in effect. For the triangle inequality $d_{ij} - d_{jk} - d_{ki} \leq 0$ we define a vector $\boldsymbol{a}$ with entries $+1, -1, -1$ in positions corresponding to edges $i \to j$, $j \to k$ and $k \to i$ respectively and zeros elsewhere. We

collect all the triangle inequality vectors and put them in a matrix $\boldsymbol{A}$. The number of rows in $\boldsymbol{A}$ equals the size of T ($3\binom{n}{3}$) and the number of columns equals the size of $\boldsymbol{d}$ ($\binom{n}{2}$). The rows of $\boldsymbol{A}$ have a 1-1 correspondence with the triples in T [2] in T, and the columns correspond to the edges of the graph G. For a complete graph on 4 vertices (Figure 1) the triangle constraint matrix $\boldsymbol{A}$ is given by

$$\boldsymbol{A} = \begin{array}{c} \text{Triple} \\ \text{abc} \\ \text{bca} \\ \text{cab} \\ \text{abd} \\ \text{bda} \\ \text{dab} \\ \text{acd} \\ \text{cda} \\ \text{dac} \\ \text{bcd} \\ \text{cdb} \\ \text{dbc} \end{array} \begin{array}{c} \begin{array}{cccccc} \text{ab} & \text{ac} & \text{ad} & \text{bc} & \text{bd} & \text{cd} \end{array} \\ \left[\begin{array}{cccccc} 1 & -1 & 0 & -1 & 0 & 0 \\ -1 & -1 & 0 & 1 & 0 & 0 \\ -1 & 1 & 0 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 & -1 & 0 \\ -1 & 0 & -1 & 0 & 1 & 0 \\ -1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 & 0 & -1 \\ 0 & -1 & -1 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 & -1 \\ 0 & 0 & 0 & -1 & -1 & 1 \\ 0 & 0 & 0 & -1 & 1 & -1 \end{array}\right] \end{array}. \tag{3.2}$$

In the triangle constraint matrix $\boldsymbol{A}$ above, ab, ac, bc, etc., represent the edges that form the triangle. The row numbers on the left are used to index the inequalities and correspond to the triples in an ordered listing of T (see 3.1). It is easy to see that whenever a triangle inequality is violated, the corresponding component of $\boldsymbol{Ad}$ is greater than 0.

3.1 The Problem

The metric nearness problem requests a metric graph G' (with distance matrix $\boldsymbol{M}$) that is closest to a given nonmetric graph G (with dissimilarity matrix $\boldsymbol{D}$) with respect to some measure of divergence. Specifically we seek a distance matrix $\boldsymbol{M}$ so that,

$$B_\varphi(\boldsymbol{M}\|\boldsymbol{D}) \equiv B_\varphi(\boldsymbol{m}\|\boldsymbol{d}) = \sum_{ij} B_\varphi(m_{ij}\|d_{ij}), \tag{3.3}$$

is minimized subject to the restriction that $\boldsymbol{M}$ correspond to a metric graph (i.e., $\boldsymbol{Am} \leq 0$), $B_\varphi(x\|y)$ be a Bregman divergence and φ be a Bregman function.

Remark 3.3. If $\varphi(x)$ is a Bregman function such that $\nabla\varphi(x_0) = 0$ then

$$\operatorname*{argmin}_{x\in\Omega} B_\varphi(x\|x_0) = \operatorname*{argmin}_{x\in\Omega} \varphi(x),$$

where Ω is the domain of $\varphi(x)$. This fact can be easily proved by using the definition of a Bregman divergence.

Bregman's optimization algorithm works for Bregman functions. We first explore how to solve the nearness problem for *vector* ℓ_p norms ($1 < p < \infty$) that are Bregman functions also. For these norms the problem is to find $\boldsymbol{m}$ to

$$\begin{aligned} \text{minimize} \quad & \frac{1}{p}\|\boldsymbol{m} - \boldsymbol{d}\|_p^p, \\ \text{subject to} \quad & \boldsymbol{Am} \leq 0. \end{aligned} \tag{3.4}$$

[2] We do not need to construct the matrix $\boldsymbol{A}$ in an implementation and we also do not need to construct T. The triples can be generated in some desired order and the corresponding constraint vector $\boldsymbol{a}$ determined at runtime.

For simplicity we introduce an auxiliary variable $\boldsymbol{x} = \boldsymbol{m} - \boldsymbol{d}$ that we call the error vector. Then our problem is to minimize $\frac{1}{p}\|\boldsymbol{x}\|_p^p$, subject to the constraints $\boldsymbol{A}(\boldsymbol{d}+\boldsymbol{x}) \leq 0 \equiv \boldsymbol{A}\boldsymbol{x} \leq -\boldsymbol{A}\boldsymbol{d}$. Thus we set $\varphi(\boldsymbol{x}) = \frac{1}{p}\|\boldsymbol{x}\|_p^p$. Trivially $\varphi(\boldsymbol{x})$ has global minimizer at $\boldsymbol{m} = \boldsymbol{d}$, i.e., $\boldsymbol{x} = 0$, hence minimizing $\varphi(\boldsymbol{x})$ is the same as minimizing $B_\varphi(\boldsymbol{x}\|\boldsymbol{0})$.

3.2 Handling ℓ_1 and ℓ_∞

Our approach is not restricted to Bregman functions. The most notable exceptions that we can tackle in our triangle fixing framework are the problems arising from measuring the nearness using the ℓ_1 and ℓ_∞ vector norms. Both these norms are important because the MN problem modeling their minimization yields linear programs. The Linear Programming (LP) formulation for these norms was originally presented in the MN Technical Report [Dhillon et al., 2003].

For the ℓ_1 and ℓ_∞ cases, the objective functions are,

$$\begin{aligned} & \min \mathbf{1}^T|\boldsymbol{x}|, \quad \text{for } \ell_1, \\ \text{and,} \quad & \min \max_i |x_i|, \quad \text{for } \ell_\infty. \end{aligned}$$

Observe that these two measures are not strictly convex, whereby triangle fixing based on Bregman's algorithm cannot be applied directly. Using an LP solver turns out to be computationally expensive both in time and storage. Fortunately, as is shown in the next section, we may model the linear programs for these two cases by corresponding quadratic programs that offer strict convexity that we may exploit.

4 Algorithms

This section contains triangle fixing operations that solve the MN problem for a vast assortment of Bregman divergence measures and also for the ℓ_1 and ℓ_∞ norms. We begin by illustrating the triangle fixing operation for the ℓ_2 case that is not only the simplest, but also pivotal in solving the ℓ_1 and ℓ_∞ cases.

4.1 The ℓ_2 case

For the ℓ_2 case we can use either Bregman's algorithm based triangle fixing (after appropriate simplifications of the general procedure given in the next section) or use Dykstra's algorithm (see Appendix A for details). Here we provide the triangle fixing procedure based on Dykstra's algorithm for the ℓ_2 case. The main difference between our implementation of Bregman's algorithm and Dykstra's algorithm for the ℓ_2 case is that the former minimizes $\frac{1}{2}\|\boldsymbol{x}\|^2$ subject to $\boldsymbol{A}\boldsymbol{x} \leq -\boldsymbol{A}\boldsymbol{d}$, whereas the latter minimizes $\frac{1}{2}\|\boldsymbol{m} - \boldsymbol{d}\|^2$ subject to $\boldsymbol{A}\boldsymbol{m} \leq 0$. The correctness and convergence of our procedures follow from the corresponding properties of Bregman's and Dykstra's procedures.

We see in the algorithm that we work by fixing one triangle at a time instead of fixing the entire graph. This fact is a simple consequence of the structure of the triangle constraint vector $\boldsymbol{a}$ that is 0 for all edges of the graph that are not a part of the triangle to which it refers. Consequently, we call the process of performing a projection onto the half-space defined by the constraint vector $\boldsymbol{a}$ to be the process of triangle fixing.

For each triangle inequality in the input graph, Algorithm 4.1 performs the projection as given by Corollary 2.4. We see that $\|\boldsymbol{a}\|^2 = 3$, and we are projecting onto the hyperspace $\boldsymbol{a}^T\boldsymbol{x} \leq 0$. Therefore we set $\boldsymbol{x} \leftarrow \boldsymbol{x} - \frac{1}{\|\boldsymbol{a}\|^2}\delta^+\boldsymbol{a}$. For an inequality, say (a, b, c), that corresponds to $\boldsymbol{a}$ we have

$\delta = [\langle \boldsymbol{a}, \boldsymbol{x} \rangle - 0] = ab - bc - ca$. Since the vector $\boldsymbol{a}$ has a $+1$ for the edge whose weight (say ab) could be possibly larger than the sum of the other two edge weights, we subtract $\delta/3$ from ab. Also, $\boldsymbol{a}$ has -1 in the positions corresponding to the edges bc and ca hence we add δ to bc and ca. We observe that there exists a δ for each triangle. Therefore we need to store $\binom{n}{3}$ floating point numbers.

ALGORITHM 4.1: Triangle fixing for ℓ_2 norm.

L2_METRIC_NEARNESS(G, T)
Input: G the input graph, T list of triples (inequalities to fix)
Output: G after metrizing.

{Initialization}
foreach $(i, j, k) \in T$
  $\delta_{ijk} = 0$ {Correction terms}
while *not converged*
  foreach $(a, b, c) \in T$
    $(oa, ob, oc) \leftarrow (ab, bc, ca)$
    {Apply correction}
    $ab \leftarrow ab + \delta_{abc}$; $bc \leftarrow bc - \delta_{abc}$; $ca \leftarrow ca - \delta_{abc}$
    $\delta = ab - bc - ca$
    if $(\delta > 0)$
      $ab \leftarrow ab - \delta/3$
      $bc \leftarrow bc + \delta/3$
      $ca \leftarrow ca + \delta/3$
    endif
    $\delta_{abc} = \delta_{abc} - ab + oa$
  endfor
end.

4.2 General ℓ_p norms ($1 < p < \ell_\infty$)

Now we present the triangle-fixing procedure when the error measure is an arbitrary ℓ_p norm. This case becomes immediately computationally more difficult than the quadratic (ℓ_2) case because of the need to solve nonlinear equations to determine the projection parameter (see 2.2).

Coming back to the ℓ_p problem we see that we have to minimize $\varphi(\boldsymbol{x}) = \frac{1}{p}\|\boldsymbol{x}\|_p^p$, where $\|\cdot\|_p$ is the vector p-norm ($\|\boldsymbol{x}\|_p = (\sum_i |x_i|^p)^{1/p}$). Using (2.2) we find that the projection must satisfy

$$\nabla\varphi(\boldsymbol{x}^*) \equiv \operatorname{sgn}(\boldsymbol{x}^*)|\boldsymbol{x}^*|^{p-1} = \operatorname{sgn}(\boldsymbol{x})|\boldsymbol{x}|^{p-1} + \mu\boldsymbol{a}. \tag{4.1}$$

If the Legendre conjugate is well defined we may write (2.2) as

$$\boldsymbol{x}^* = (\nabla\varphi^*)\big(\nabla\varphi(\boldsymbol{x}) + \mu\boldsymbol{a}\big). \tag{4.2}$$

Equation (4.2) allows us to solve for the updated vector once the parameter μ is known. Observe that for a given triangle constraint vector $\boldsymbol{a}$, if $a_i = 0$, then $x_i^* = (\nabla\varphi^*)\big(\nabla\varphi(x_i)\big) = x_i$. Thus the triangle fixing once again becomes evident as we only have to modify the components corresponding to the edges of the triangle described by $\boldsymbol{a}$. This special structure of $\boldsymbol{a}$ is essential for the efficient implementation of Bregman's algorithm. Since we have to consider only three edges, we need

to store at most three numbers per triangle under consideration, leading to decreased storage requirements.

Note that $\boldsymbol{x}$ denotes the vector of additive changes to $\boldsymbol{d}$ that are needed to take $\boldsymbol{d}$ to the nearest metric graph. Consider the triple (a, b, c) for which which we wish to enforce $\boldsymbol{a}^T\boldsymbol{x} \leq b$. On applying (4.2) to (4.1) we obtain,

$$x^*_{ab} = \operatorname{sgn}\big((\operatorname{sgn}(x_{ab})|x_{ab}|^{p-1} + \mu)\big)\left|\operatorname{sgn}(x_{ab})|x_{ab}|^{p-1} + \mu\right|^{\frac{1}{p-1}} \tag{4.3a}$$

$$x^*_{bc} = \operatorname{sgn}\big((\operatorname{sgn}(x_{bc})|x_{bc}|^{p-1} - \mu)\big)\left|\operatorname{sgn}(x_{bc})|x_{bc}|^{p-1} - \mu\right|^{\frac{1}{p-1}} \tag{4.3b}$$

$$x^*_{ca} = \operatorname{sgn}\big((\operatorname{sgn}(x_{ca})|x_{ca}|^{p-1} - \mu)\big)\left|\operatorname{sgn}(x_{ca})|x_{ca}|^{p-1} - \mu\right|^{\frac{1}{p-1}} \tag{4.3c}$$

$$x^*_{ab} - x^*_{bc} - x^*_{ca} - b = 0, \quad \text{on using } \boldsymbol{a}^T\boldsymbol{x} \leq b. \tag{4.3d}$$

ALGORITHM 4.2: Bregman Projection for ℓ_p norm ($1 < p < \infty$).

FINDMU($\boldsymbol{x}_{abc}$, (a, b, c))
Input: $\boldsymbol{x}_{abc}$ the correction vector for the triple (a, b, c)
Output: μ as per (4.3).
{Enforces the constraint $\boldsymbol{a}^T\boldsymbol{x} \leq b$}
begin
  $(x_{ab}, x_{bc}, x_{ca}) \leftarrow \boldsymbol{x}_{abc}$
  $b \leftarrow -(ab - bc - ca)$ $\quad \{b = -\boldsymbol{a}^T\boldsymbol{d}\}$
  $\delta \leftarrow x_{ab} - x_{bc} - x_{ca} - b$ $\quad \{[\boldsymbol{a}^T\boldsymbol{x} - b]^+\}$
  if $(\delta > 0)$
    $\alpha \leftarrow \operatorname{sgn}(x_{ab})|x_{ab}|^{p-1}$
    $\beta \leftarrow \operatorname{sgn}(x_{bc})|x_{bc}|^{p-1}$
    $\gamma \leftarrow \operatorname{sgn}(x_{ca})|x_{ca}|^{p-1}$
    $s \leftarrow 1/(p-1)$
    $f(\mu) := \operatorname{sgn}(\alpha + \mu)|\alpha + \mu|^s - \operatorname{sgn}(\beta - \mu)|\beta - \mu|^s - \operatorname{sgn}(\gamma - \mu)|\gamma - \mu|^s$
    $\mu \leftarrow f^{-1}(\alpha, \beta, \gamma)$ $\quad$ {Found numerically}
  endif
return μ

We solve the set of equations (4.3) for μ (using some nonlinear root finding method) and then back substitute into (4.3a)–(4.3c) to obtain the values for x^*_{ab}, x^*_{bc} and x^*_{ca}. Algorithm 4.2 computes μ. The equations (4.3) are explicit enough so we do not show the procedure FIXTRIANGLE that implements them. Since these equations need to be solved multiple times, if not solved efficiently they could easily become a bottleneck of the triangle fixing algorithm.

ALGORITHM 4.3: Generic Triangle Fixing For ℓ_p $(1 < p < \infty)$.

TRIANGLE_FIXING(G, T)
Input: G: Input graph, T set of triples
Output: G' nearest metric graph in specified distortion

foreach $(a, b, c) \in T$
  $z_{abc} \leftarrow 0$
foreach $ab \in G$
  $x_{ab} \leftarrow ab$ {Error values for each edge}
while not converged
  foreach $(a, b, c) \in T$
    $\boldsymbol{x}_{abc} \leftarrow (x_{ab}, x_{bc}, x_{ca})$
    $\mu \leftarrow \text{FINDMU}(\boldsymbol{x}_{abc}, (a, b, c))$
    $\theta \leftarrow \min(\mu, z_{abc})$
    $(x^*_{ab}, x^*_{bc}, x^*_{ca}) \leftarrow \text{FIXTRIANGLE}(x_{abc}, (a, b, c), \theta)$ Fix using (4.3)
    $z_\Delta \leftarrow z_{abc} - \theta$
  end *foreach*
end *while*

At this point we may also remark that solving MN for general Bregman divergences will involve performing a Bregman projection which could be computationally intensive. Sometimes one can use first order approximations without performing the projections very accurately and still converge to the optimum. More information regarding this remark may be found in the paper by Bauschke and Borwein [1997].

4.3 MN for the ℓ_1 and ℓ_∞ norms

The basic triangle fixing algorithm succeeds only when the nearness measure used in is strictly convex. Hence, it cannot be applied directly to the ℓ_1 and ℓ_∞ cases. These require a more sophisticated approach.

First, observe that the problem of minimizing the ℓ_1 norm of the changes can be written as an LP:

$$\begin{aligned} &\min_{\boldsymbol{x},\boldsymbol{z}} \ \mathbf{1}^T\boldsymbol{z} \\ &\text{subject to } \boldsymbol{A}\boldsymbol{x} \leq -\boldsymbol{A}\boldsymbol{d},\ -\boldsymbol{x} - \boldsymbol{z} \leq \mathbf{0},\ \boldsymbol{x} - \boldsymbol{z} \leq \mathbf{0}. \end{aligned} \tag{4.4}$$

The auxiliary variable $\boldsymbol{z}$ can be interpreted as the absolute value of $\boldsymbol{x}$. Similarly, minimizing the ℓ_∞ norm of the changes can be accomplished with the LP

$$\begin{aligned} &\min_{\boldsymbol{x},\zeta} \ \zeta \\ &\text{subject to } \boldsymbol{A}\boldsymbol{x} \leq -\boldsymbol{A}\boldsymbol{d},\ -\boldsymbol{x} - \zeta\mathbf{1} \leq \mathbf{0},\ \boldsymbol{x} - \zeta\mathbf{1} \leq \mathbf{0}. \end{aligned} \tag{4.5}$$

We interpret $\zeta = \|\boldsymbol{x}\|_\infty$.

Solving these linear programs using standard software is prohibitively expensive because of the large number of constraints. Moreover, the solutions are not unique because the ℓ_1 and ℓ_∞ norms are not strictly convex. Instead, we replace the LP by a quadratic program (QP) that is strictly convex and returns the solution of the LP that has minimum ℓ_2-norm. For the ℓ_1 case, we have the following result.

Theorem 4.1 (ℓ_1 Metric Nearness). *There exists a constant $\lambda > 0$ such that*

$$\operatorname*{argmin}_{\boldsymbol{z} \in Z} \ \|\lambda \mathbf{1} + \boldsymbol{z}\|_2 \quad = \quad \operatorname*{argmin}_{\boldsymbol{z} \in Z^\star} \ \|\boldsymbol{z}\|_2, \tag{4.6}$$

where Z is the feasible set for (4.4) *and $Z^\star$ is the set of optimal solutions to* (4.4)*. The minimizer is unique.*

Theorem 4.1 follows from an application of a result of Mangasarian [1984, Theorem 2.1-a-i]. A similar theorem may be stated for the ℓ_∞ case.

The QP (4.6) can be solved using an augmented triangle-fixing algorithm since the majority of the constraints in (4.6) are triangle inequalities. As in the ℓ_2 case, the triangle constraints are enforced using (3.4). Each remaining constraint can be enforced by computing an orthogonal projection. Algorithm 4.4 gives a sample implementation of the augmented triangle fixing procedure for solving (4.6).

We have found empirically that for the ℓ_∞ case a small value of λ yields good answers and for the ℓ_1 problem, λ must be usually larger than the maximum edge weight in the graph.

ALGORITHM 4.4: Triangle fixing for ℓ_1 norm.

```
L1_METRIC_NEARNESS(d, T, λ)
Input: d input dissimilarity vector, T list of triples, λ parameter
Output: d after metrizing.
begin
   {Initialization}
   z ← −λ1
   for i = 1 to length(d)
      ε_x(i) = 0; ε_z(i) = 0;        {Correction terms}
      γ_x(i) = 0; γ_z(i) = 0;        {Correction terms}
   while not converged
      {Perform inner loop triangle fixing as in Algorithm 4.1}
      {Now we enforce −x − z ≤ 0}
      for i = 1 to length(d)
         t_x ← d_i; t_z ← z_i
         δ ← −d_i − ε_x(i) − z_i − ε_z(i)
         if δ > 0
            d_i ← d_i + ε_x(i) + 0.5 * δ
            z_i ← z_i + ε_z(i) + 0.5 * δ
         ε_x(i) ← ε_x(i) + t_x − d_i
         ε_z(i) ← ε_z(i) + t_z − z_i
         {Now we enforce x − z ≤ 0}
         t_x ← d_i; t_z = z_i
         δ ← d_i + γ_x(i) − z_i − γ_z(i)
         if δ > 0
            d_i ← d_i + γ_x(i) − 0.5 * δ
            z_i ← z_i + γ_z(i) + 0.5 * δ
         γ_x(i) ← γ_x(i) + t_x − d_i
         γ_z(i) ← γ_z(i) + t_z − z_i
      end for
   end while
end.
```

4.4 Variations

One can think of numerous variations of the MN problem where new constraints are added to the problem or some constraints are removed. A few simple ideas are considered here and they can all be easily handled within our current framework. For example,

- Consider MN with any element-wise monotonic distance measure with the stipulation that we allow only decreases in the original dissimilarity values. Plaxton and Clement [2003–2004] brought to our attention the result that for such a decrease only MN problem, the element-wise maximal solution is given by the All Pairs Shortest Path (APSP) solution. This interesting observation led us to explore the connection between Linear Programming and APSP and we expand a little bit on that in Section 6.1.

- In a similar vein, one can also consider the increase only variation of MN, in which one desires the nearest metric graph to have edge weights greater than or equal to the input non-metric graph.

- Plaxton and Clement [2003–2004] also suggested a non-strictly polynomial time binary search procedure for finding the nearest metric graph where nearness is measured by the ℓ_∞ norm. The binary search procedure starts by guessing the ℓ_∞ error, adding that to all edge weights, and then doing APSP on the graph to see if the reduced edge weights are still within the guessed range. If so, it decreases its guess and searches again, else it increases its guess and repeats the search. They also seem to have some research on solving the MN problem for the ℓ_1 norm.

- One could enforce ordering constraints on values, i.e., if in the input $d_{ij} < d_{pq}$ then we also want $m_{ij} < m_{pq}$.

- One might also require box constraints on the metric solution, i.e., $\boldsymbol{l} \leq \boldsymbol{m} \leq \boldsymbol{u}$. Note that a non-zero $\boldsymbol{l}$ will guarantee a metric instead of a possibly quasi-metric solution (in a quasi-metric $m_{ij} = 0$ is possible even if $i \neq j$).

- Finding Approximate metrics (where all the triangle constraints need not be satisfied) can also be tackled.

- Certain applications that make use λ-triangle inequalities also fall under the framework. In this case a given triangle vector $\boldsymbol{a}$ has $+\lambda_1$, $-\lambda_2$, $-\lambda_3$ instead of the usual $+1$, -1 and -1 components. The problem is still triangle fixing because the structure is determined by the presence of the non-zeros.

5 Examples and Experiments

The MN problem has an input of size N, and the number of constraints is roughly $N^{3/2}$. We ran experiments to ascertain the empirical behavior of the algorithm. Figure 2 shows log–log plots of the running time of our algorithms for solving the ℓ_1 and ℓ_2 Metric Nearness Problems. The time cost appears to be $O(N^{3/2})$, which is *linear* in the number of constraints. The results plotted in the figure were obtained by executing the algorithms on random dissimilarity matrices. The procedure was halted when the distance values changed less than 10^{-3} from one iteration to the next. For both problems, the results were obtained with a simple MATLAB implementation. Nevertheless, this basic version outperforms MATLAB's optimization package by one or two orders of magnitude

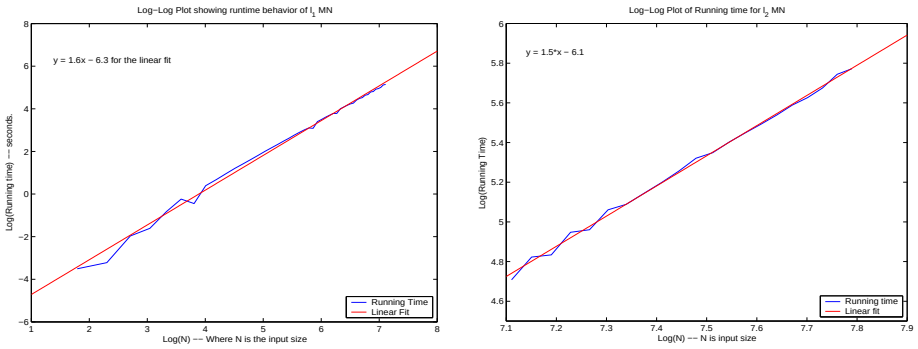


Figure 2: Running time for ℓ_1 and ℓ_2 norm solutions (plots have different scales).

(depending on the problem). A more sophisticated (C or parallel) implementation could improve the running time even more, which would allow us to study larger problems.

6 Discussion

Our iterative procedures proceed by fixing triangles one by one. This operation forms the common core for all metric nearness problems. It would be interesting to see if one can obtain faster methods for solving the nearness problem at least for ℓ_1 and ℓ_∞ cases. Because the running time of APSP (without allowing arithmetic on edge weights) is lower bounded by $\Omega(N^{3/2})$, the metric nearness problem is also lower bounded by the same. Since our methods proceed by fixing triangles, it is not evident how one could perform sub-cubic computation and still achieve a low error of approximation. We make a passing observation that if one requires to enforce both ordering and metricity, one can trivially achieve that by (with high error) adding $\max_{ij} d_{ij}$ to each edge of the graph. In this case the error is as high as possible, but the running time is just $O(N)$.

6.1 Linear Programming and APSP

As previously noted, if just one sided decreasing changes are allowed to the edge weights of the graph, then the graph yielded by APSP solves the problem for any *monotonic* error measure. Since we originally formulated MN as as a linear program, the APSP solution lends to the curiosity of the problem because it can be turned around and one can ask: What is the LP interpretation of APSP? The LP interpretation of Dijkstra's SSSP (see [Papadimitriou and Steiglitz, 2000]) has long been known, and to our knowledge there has been a gaping hole for an LP view of the APSP problem (other than an LP arising out of Multi-commodity network flows). Since both APSP and the decreasing metric nearness essentially proceed by triangle fixing operations, we can build an explicit connection between an LP solution and an APSP solution. We do not give an explicit mapping between the moves made by a primal-dual simplex procedure for solving the decreasing MN and APSP.

7 Summary and Future Work

In this paper we presented generic triangle fixing algorithms for solving the metric nearness problem [Dhillon et al., 2003]. All our algorithms share their structure, differing only in the particular Bregman projection involved to fix a given triangle. We saw that the algorithms are quite efficient and they allow us to obtain fairly accurate approximations.

There are various issues however that would merit further investigation. Some possibilities are:

- The decrease only case that performs arithmetic on edge weights, to obtain metricity in $O(APSP)$ time.
- Decreasing storage requirements for large graphs or even performing out of core metric nearness. For example,
  1. Since all we need to do is to perform cyclic fixes to the triangles we can fix a large number of triangles and then swap their data (error vectors) out to disk and fix the next set and proceed henceforth.
  2. Is it possible to recompute the errors instead of storing them? Can one store $O(N)$ or lesser information and construct the error vectors for each triangle so that the overall iteration still remains $O(N^{3/2})$. It seems that the storage requirements are $\Omega(N^{3/2})$ because there are $\Omega(N^{3/2})$ constraints and we need to store error values for each constraint.
- Algorithms that seek to explore graphs theoretic procedures for the ℓ_1 MN problem based on a network simplex approach could be investigated.

We plan to further investigate the application of MN to other problems in data mining, machine learning and database query retrieval.

Acknowledgements

We would like to acknowledge Prof. C. G. Plaxton and his student A. Clement for exposing the connection of MN to APSP and also for the binary search procedure for the ℓ_∞ case. This research was supported by NSF CAREER Award No. 0093404, Texas Advanced Research Program Grant 003658-0431-2001 and NSF ITR Award No. IIS-0325116.

References

H. H. Bauschke and J. M. Borwein. Legendre functions and the method of random Bregman projections. *Journal of Convex Analysis*, 4:27–67, 1997.

H. H. Bauschke and A. S. Lewis. Dykstra's algorithm with Bregman projections: a convergence proof.

L. M. Bregman, Y. Censor, and S. Reich. Dykstra's Algorithm as the Nonlinear Extension of Bregman's Optimization Method. *Journal of Convex Analysis*, 6(2):319–333, 1999.

Y. Censor and S. A. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Numerical Mathematics and Scientific Computation. Oxford University Press, 1997.

F. R. Deutsch. *Best Approximation in Inner Product Spaces*. Springer Verlag, first edition, 2001. ISBN 0387951563.

I. S. Dhillon, S. Sra, and J. A. Tropp. The Metric Nearness Problems with Applications. Technical Report TR-03-23, Computer Sciences, University of Texas at Austin, 2003.

M. R. Hestenes. *Optimization Theory: The Finite Dimensional Case*. Wiley Interscience, New York, 1975. ISBN 0471374717.

O. L. Mangasarian. Normal solutions of linear programs. *Mathematical Programming Study*, 22: 206–216, 1984.

C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover, 2000.

C. G. Plaxton and A. Clement. Personal Communication, 2003–2004.

A Bregman's and Dykstra's Algorithms

A quick summary of the algorithms central to our approach is provided here.

A.1 Bregman's Algorithm for Inequality Constrained Problems

Consider the problem,

$$\begin{aligned} \text{minimize} \quad & \varphi(x) \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x} \in \bar{S}, \varphi \in \mathscr{B}(S). \end{aligned} \tag{A.1}$$

Bregman's algorithm (as described below) cycles through the constraints one by one performing an appropriate projection with correction to converge towards the optimal solution. The constraints are visited in a cyclic fashion and thus if the number of constraints is r, the $[kr] = r, k \in \mathbb{N}$ and $[n] = n \bmod r$ otherwise.

Algorithm A.5: Bregman's algorithm for Inequality Constrained Problems

Bregman(φ, $\boldsymbol{A}$, $\boldsymbol{b}$)
Input: φ, $\boldsymbol{A}$, $\boldsymbol{b}$ as in (A.1)
Output: $\operatorname{argmin} \varphi(x)$ s.t. constraints are satisfied

{Initialization}
$\boldsymbol{x}^0 \in \{\boldsymbol{x} \in S | \exists \boldsymbol{z} \in \mathbb{R}^m_+ \ni \nabla\varphi(\boldsymbol{x}) = -\boldsymbol{A}^T\boldsymbol{z}\}$ and $\boldsymbol{z}^0$ s.t. $\nabla\varphi(\boldsymbol{x}^0) = -\boldsymbol{A}^T\boldsymbol{z}^0$

{Iterative step} Given $\boldsymbol{x}^n$ and $\boldsymbol{z}^n$ perform the following updates
repeat
  $c_n := \min(\boldsymbol{z}^n_{[n]}, \mu_n)$
  $\nabla\varphi(\boldsymbol{x}^{n+1}) = \nabla\varphi(\boldsymbol{x}^n) + c_n\boldsymbol{a}_{[n]}$
  $\boldsymbol{z}^{n+1} = \boldsymbol{z}^n - c_n\boldsymbol{e}_{[n]}$
  {μ_n is calculated by solving (2.2) with $\boldsymbol{y} = \boldsymbol{x}^n$, $\boldsymbol{a} = \boldsymbol{a}_{[n]}$ and $b = \boldsymbol{b}_{[n]}$.}
until *convergence*.

A.2 Dykstra's Algorithm

Dykstra's algorithm can be viewed as a special case of Bregman's method by appealing to the generalization of Bregman's method when the constraint sets are arbitrary convex sets instead of just half-spaces. The interested reader is pointed to the work by Bauschke and Lewis, who describe a Dysktra style algorithm for a restricted class of strictly convex functions. Bregman et al. [1999] show how one can consider Dykstra's Algorithm as the non-linear extension of Bregman's cylic projection procedure.

We describe Dykstra's Algorithm it here for the sake of completeness and since we directly use it for solving the nearness problem for ℓ_1, ℓ_2 and ℓ_∞ based nearness measures.

Dykstra's algorithm minimizes,

$$\begin{aligned} &\frac{1}{2}\|\boldsymbol{x}_0 - \boldsymbol{y}\|^2 \\ &\text{s.t. } \boldsymbol{y} \in \cap_{i=1}^{r} C_i, \quad \text{each } C_i \text{ is a convex set} \end{aligned} \tag{A.2}$$

For further details and a proof of convergence the interested reader is referred to Deutsch [2001].

ALGORITHM A.6: Projection onto the intersection of convex sets

DYKSTRA($\boldsymbol{x}_0$, C_1,...,C_r)
Input: $\boldsymbol{x}_0$, C_i convex sets
Output: $\text{argmin}_{\boldsymbol{y}} \|\boldsymbol{x}_0 - \boldsymbol{y}\|^2$ s.t. constraints are satisfied

{Initialization}
$\boldsymbol{x}^0 = \boldsymbol{x}_0$
{Iterative step} Given $\boldsymbol{x}^n$ and $\boldsymbol{z}^n$ perform the following updates
repeat
  $\boldsymbol{x}^{n+1} \leftarrow P_{[n]}(\boldsymbol{x}^n + \boldsymbol{z}^{n-r})$
  $\boldsymbol{z}^n \leftarrow \boldsymbol{x}^n - \boldsymbol{x}^{n+1} + \boldsymbol{z}^{n-r}$
  $\{P_{[n]}$ is the orthogonal projection onto the set $C_{[n]}\}$
until *convergence.*