Solving the Metric Nearness Problem: Methods and Results

Júlia Guizardi

DISSERTATION PRESENTED TO
THE INSTITUTE OF MATHEMATICS AND STATISTICS
OF THE UNIVERSITY OF SÃO PAULO
TO OBTAIN THE TITLE OF
MASTER OF SCIENCE

Program: Applied Mathematics
Advisor: Prof. Dr. Ernesto G. Birgin

This work was carried out with the support of the Coordination for the Improvement of Higher Education Personnel – Brazil (CAPES) – Funding Code 001.

São Paulo, Brazil
May, 2025

Resolução do Problema da Aproximação Métrica: Métodos e Resultados

Júlia Guizardi

Dissertação apresentada ao Programa de
Pós-Graduação em Matemática Aplicada
do Instituto de Matemática e Estatística
da Universidade de São Paulo como parte
dos requisitos para obtenção do título
de Mestre em Ciências.

Ficha catalográfica elaborada com dados inseridos pelo(a) autor(a)
Biblioteca Carlos Benjamin de Lyra
Instituto de Matemática e Estatística
Universidade de São Paulo

Guizardi, Júlia
Solving the Metric Nearness Problem: Methods and Results / Júlia Guizardi; orientador, Ernesto G. Birgin. - São Paulo, 2025.
59 p.: il.

Dissertação (Mestrado) - Programa de Pós-Graduação em Matemática Aplicada / Instituto de Matemática e Estatística / Universidade de São Paulo.
Bibliografia
Versão original

1. Aproximação. 2. Algoritmos. I. G. Birgin, Ernesto. II. Título.

Bibliotecárias do Serviço de Informação e Biblioteca
Carlos Benjamin de Lyra do IME-USP, responsáveis pela
estrutura de catalogação da publicação de acordo com a AACR2:
Maria Lúcia Ribeiro CRB-8/2766; Stela do Nascimento Madruga CRB 8/7534.

Abstract

Júlia Guizardi. **Solving the Metric Nearness Problem: Methods and Results**. Master's Thesis (Master) - Institute of Matemathics and Statistics, University of São Paulo, São Paulo, 2025.

The Metric Nearness Problem consists of finding the closest distance matrix, using the ℓ_p norm, to a given dissimilarity matrix such that metric properties, particularly the triangle inequalities, are satisfied. This problem can be applied in various contexts, including image processing, data clustering, and sensor location (when additional constraints are imposed), where noisy or incomplete distance data must be corrected. This work provides both theoretical and practical studies of the problem. The minimization formulation is presented under the ℓ_1, ℓ_2 and ℓ_∞ norms. Two numerical optimization methods are studied and applied: the Augmented Lagrangian method via Algencan, and the Dykstra projection method, which was specifically programmed for this problem. Both are applied to the three norms. Optimizations were implemented to handle the large number of constraints by dealing with patterns in the algorithm. Extensive experiments were conducted on synthetic and real-world datasets to evaluate computational performance and numerical accuracy.

The ℓ_2 norm problem was especially suitable for Algencan given its nonlinear structure, while Dykstra's method showed superior performance in large-scale settings due to the specific implementation that generates low memory requirements. In linear cases, such as the ℓ_1, ℓ_∞ norms, a comparison with the Simplex method was also performed. Results reveal that the Dykstra method, when properly applied, can solve problems up to 10^{11} constraints.

Keyword: Metric Nearness Problem, Algencan, Dykstra.

Resumo

Júlia Guizardi. **Resolução do Problema da Aproximação Métrica: Métodos e Resultados**. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2025.

O Problema da Aproximação Métrica consiste em encontrar a matriz de distâncias mais próxima, usando a norma ℓ_p, para uma matriz de dissimilaridade dada de forma que as propriedades métricas, particularmente as desigualdades triangulares, sejam satisfeitas. Este problema pode ser aplicado em diversos contextos, incluindo processamento de imagens, agrupamento de dados e localização de sensores (quando restrições adicionais são acrescentadas), onde os dados fornecidos possuem ruídos ou estão incompletos e precisam ser corrigidos.

Este trabalho oferece estudos tanto teóricos quanto práticos do problema. A formulação do problema é apresentada sob as normas ℓ_1, ℓ_2 e ℓ_∞. Dois métodos de otimização numérica são estudados e aplicados: o método de Lagrangiano Aumentado via Algencan e o método de projeção de Dykstra, que pode ser especificamente programado para este problema. Ambos são aplicados às três normas. Otimizações foram implementadas para lidar com o grande número de restrições, aproveitando de padrões no algoritmo. Diversos experimentos foram conduzidos em conjuntos de dados sintéticos e reais para avaliar o desempenho computacional e a precisão numérica.

O problema com a norma ℓ_2 foi especialmente adaptado ao Algencan devido à sua estrutura não linear, enquanto o método de Dykstra mostrou desempenho superior em conjuntos de grande escala devido à implementação específica que gera baixos requisitos de memória. Em casos lineares, com as normas ℓ_1 e ℓ_∞, também foi realizada uma comparação com o método Simplex. Os resultados revelam que o método de Dykstra, quando aplicado corretamente, pode resolver problemas com até 10^{11} restrições.

Palavras-chave: Problema da Aproximação Métrica, Algencan, Dykstra.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

When analyzing data, matrices are an articulated solution to store and structure information. If a matrix holds the purpose of measuring relations in a dataset, it is called an *association matrix.* One common use for association matrices is to store distances of a set of points.

Many real-world problems involve matrices that should store distances that satisfy metric properties, such as image processing [3], computed tomography reconstruction [28], and GPS-free positioning [9], [17]. In practice, these problems have data imperfections and estimated datasets that cause violations of the metric properties. The metric nearness problem consists in, given a set of approximated distances $D = (d_{ij}) \in \mathbb{R}^{n\times n}$, find the nearest distance matrix $X = (x_{ij}) \in \mathbb{R}^{n\times n}$ satisfying the triangle inequalities in a high-dimensional space, that is

$$\underset{X\in\Omega}{\text{Minimize}} \left(\sum_{1\leq i<j\leq n} |w_{ij}(x_{ij} - d_{ij})|^p \right)^{\frac{1}{p}}, \tag{1.1}$$

where Ω is the set of all distance matrices in $\mathbb{R}^{n\times n}$, $W = (w_{ij}) \in \mathbb{R}^{n\times n}$ is the weight matrix and $p = 1, 2, \ldots, \infty$. With this formulation, the metric nearness problem annihilates the noisy factor and restores values that were not given, addressing the problem in many applications that require more precise distance values. For example, sensor location [17] that, along with other constraints, uses real-world measurements to determine the positions of sensors in a distributed network. Sensor locations are also used for wireless sensor networks, robotics, and geolocation systems that require precise positioning.

One of the most renowned applications is data clustering [12], which consists of approximating nonmetric data with metric data, to take advantage of fast clustering algorithms and still guarantee the quality of the results. Data clustering has many applications, for example, approximate mutation probability matrices from protein sequencing in biology [11]. Another application is reconstructing neural structures in electron microscopy images [32] by clustering two segmentations of closely correlated images.

There are many methods to solve optimization problems. What makes this particular problem interesting is the large number of constraints that the entries in X must satisfy. For instance, considering three points in space, there are three triangle inequalities satisfied by the distances between them. Adding a fourth point increases the number of triangle inequalities to 12. In other words, the number of constraints grows cubically, and when the number of points reaches one hundred, the number of triangle inequalities is of the order of 10^6. Since this problem has many applications, some works have been developed to focus on different approaches and adjustments to optimize the execution, specifically of the metric nearness problem dealing with the high volume of constraints with a fast approach. PALM [30], for example, is a proximal augmented Lagrangian method that solves each subproblem using semismooth Newton method. A different approach can be seen in [27], where it is used a parallel projection method without locking variables.

The first explored approach in this work consists of using a software called Algencan [1], which uses the Augmented Lagrangian method to solve minimization problems. Algencan is a Fortran subroutine that gives the user the capability of controlling parameters, programming derivatives, and smartly storing them. In addition to that, it is well known that Lagrangian methods are effective in solving problems with a large number of inequality constraints, which means they can be used to work with enormous quantities of data and achieve an optimal solution in most cases.

For the second approach, the problem (1.1) is reformulated as a projection of the matrix D onto a convex set given by the constraints. Then, one can use Dykstra projection method [13]. Dykstra method solves (1.1) by finding a matrix in the intersection of all semi-planes defined by the inequalities. Then the matrix found is the closest one from D, yet in the feasible set. This method has also proven its value since it has been explored in many works ([30], [31], [27]) to solve this particular problem due to its simple implementation and small computational time as it adapts in a robust way to this problem. The problem (1.1) takes a linear form when $p = 1$ or $p = \infty$ then, for comparison purposes, the Simplex method will also be used for small datasets.

Each method has specific implementation details when applied to (1.1). This study aims to explore the performance of Dykstra's algorithm and Algencan individually by evaluating from small to large-scale problems and analyzing the results under different problem settings, providing data about their convergence speed, memory requirements, and numerical accuracy. Additionally, compare how different norms influence the difficulty of the problem.

In this work the main problem is presented in Chapter 2, along with the adaptations for the norm-1, norm-2 and norm-∞. The first approach to solve it, Algencan, together with some Augemented Lagrangian theory and Dykstra projection method are described in Chapter 3. In Chapter 4 these methods are applied to the metric nearness problem. The advantages of using the two considered methods are presented with all the three previously introduced norms and the experiments of both methods, including some Simplex results when pertinent. The experiments involve synthetic data as well as real data from benchmark problems. Lastly, it is presented the final conclusions of this work in Chapter 5.

Chapter 2

The Metric Nearness Problem

A matrix $D \in \mathbb{R}^{n \times n}$ is called a *dissimilarity matrix* if it contains information associating objects by a measure, being non-negative and symmetric with zero diagonal. When a dissimilarity matrix is based on a metric, that is, it satisfies the triangle inequalities, it is called a *distance matrix* $X = (x_{ij})$.

Example 2.1. *Considering a cube in $\mathbb{R}^3$ centered at $(0, 0, 0)$ with edge length two and five randomly generated points, as shown in Figure 2.1. The distance matrix X is given by:*

$$X = \begin{bmatrix} 0.0 & 0.93767 & 0.280287 & 0.539395 & 1.49874 \\ 0.93767 & 0.0 & 1.08257 & 1.3909 & 2.11248 \\ 0.280287 & 1.08257 & 0.0 & 0.609951 & 1.22746 \\ 0.539395 & 1.3909 & 0.609951 & 0.0 & 1.61963 \\ 1.49874 & 2.11248 & 1.22746 & 1.61963 & 0.0 \end{bmatrix}.$$

Note, however, that not every distance matrix corresponds to distances between points in a Euclidean space.

Given a symmetric matrix $D = (d_{ij})$ with approximate distances, and $W = (w_{ij})$, a weight matrix whose values reflect the confidence in the entries of D, the objective is to find a matrix X that is the distance matrix closest to D. This matrix must satisfy certain constraints. First, only the upper triangular part of the matrix is considered, as the matrix X must be symmetric, then the variables of the problem are

$$x_{ij}, 1 \leq i < j \leq n.$$

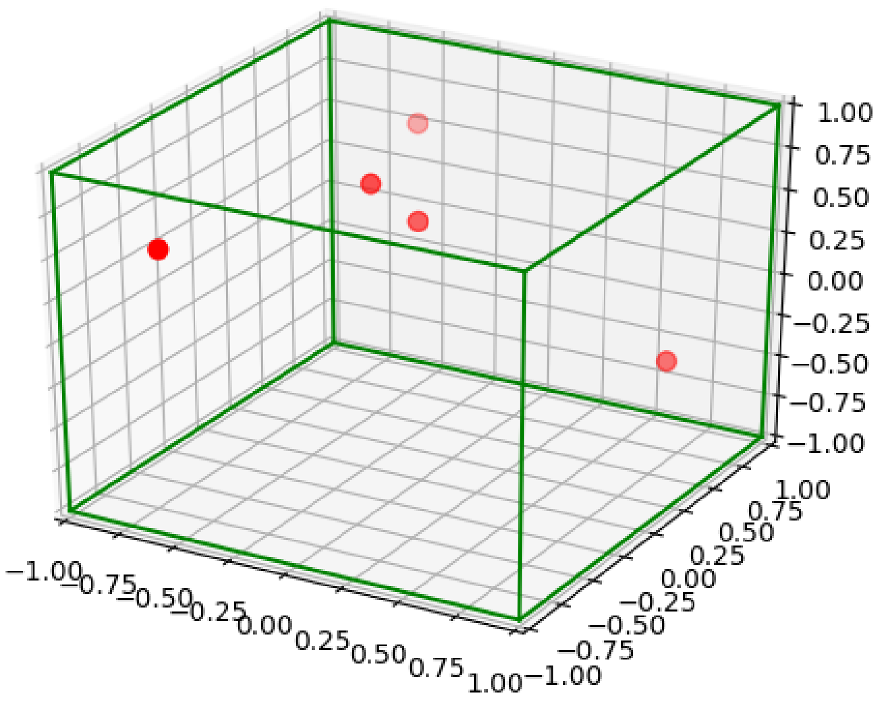

Figure 2.1: Cube centered at $(0, 0, 0)$ with five random points: (-0.460486, 0.403884, 0.429174), (-0.948317, -0.382667, 0.278899), (-0.285782, 0.471504, 0.220687), (-0.550106, 0.926224, 0.529560), (0.715726, 0.693007, -0.453529).

On top of that, all off-diagonal entries must be nonnegative:

$$x_{ij} \geq 0, \quad 1 \leq i < j \leq n.$$

The entries must also satisfy the triangle inequality. For that, let p_i and p_j be points in $\mathbb{R}^q$, and x_{ij} be the distance between them. Since only the strict upper triangular matrix is considered, one can assume that $i < j$. By fixing these two points, it is necessary to go over the remaining $n-2$ points, let's say p_k with $k = 1, \ldots, n$ and $k \neq i, j$, to establish the $m = \frac{n(n-1)(n-2)}{2}$ inequalities. Then, when $k < i$,

$$x_{ij} \leq x_{ki} + x_{kj}, \quad 1 \leq k < i < j \leq n.$$

When $i < k < j$, the inequality becomes

$$x_{ij} \leq x_{ik} + x_{kj}, \quad 1 \leq i < k < j \leq n.$$

Finally, when $j < k$,

$$x_{ij} \leq x_{ik} + x_{jk}, \quad 1 \leq i < j < k \leq n.$$

Assembling the minimization problem with its constraints, the ℓ_p norm-based metric nearness problem takes the following form:

$$\begin{array}{rll}\underset{x_{ij},1\le i<j\le n}{\text{Minimize}} & \left(\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}|w_{ij}(x_{ij}-d_{ij})|^p\right)^{\frac{1}{p}} & \\ \text{subject to} & x_{ij}\le x_{ki}+x_{kj}, & 1\le k<i<j\le n \\ & x_{ij}\le x_{ik}+x_{kj}, & 1\le i<k<j\le n \\ & x_{ij}\le x_{ik}+x_{jk}, & 1\le i<j<k\le n \\ & x_{ij}\ge 0, & 1\le i<j\le n.\end{array} \tag{2.1}$$

The matrix D may also be nonsymmetric. In this case, there are two possibilities. The first is to reformulate $D=\frac{1}{2}(D+D^T)$ and consider only the upper part. The second is to consider all the entries of D and reformulate the objective function as

$$\left(\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}|w_{ij}(x_{ij}-d_{ij})|^p+\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}|w_{ji}(x_{ij}-d_{ji})|^p\right)^{\frac{1}{p}}.$$

Assuming that the object to be minimized, the matrix X, is a zero diagonal symmetric matrix, the strict upper triangle part contains all the information needed and it can be stored as a vector. Naturally, the strict upper part of each column is followed by the strict upper part of the next one. For example, the matrix X :

$$X=\begin{pmatrix}0 & x_{12} & x_{13} & x_{14}\\ x_{21} & 0 & x_{23} & x_{24}\\ x_{31} & x_{32} & 0 & x_{34}\\ x_{41} & x_{42} & x_{43} & 0\end{pmatrix}$$

it is rewritten as the vector x :

$$x=\begin{pmatrix}x_{12} & x_{13} & x_{23} & x_{14} & x_{24} & x_{34}\end{pmatrix}.$$

This secures less storage space, since the matrix X needs n^2 entries and the vector x only needs $d=\frac{1}{2}n(n-1)$, as well as streamlines its use. Given an element x_{ij} in the matrix X, by defining the function

$$\operatorname{ind}(i,j)=\frac{1}{2}(j-1)j-j+i+1, \tag{2.2}$$

it is achieved the corresponding index of the element in the vector, or $x_{ij}=x_{\operatorname{ind}(i,j)}$. Rewriting

(2.1),

$$\begin{aligned}
\text{Minimize} \quad & \left(\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} |w_{ij}(x_{\text{ind}(i,j)} - d_{ij})|^p \right)^{\frac{1}{p}} \\
\text{subject to} \quad & x_{\text{ind}(i,j)} \le x_{\text{ind}(k,i)} + x_{\text{ind}(k,j)}, \quad 1 \le k < i < j \le n \\
& x_{\text{ind}(i,j)} \le x_{\text{ind}(i,k)} + x_{\text{ind}(k,j)}, \quad 1 \le i < k < j \le n \\
& x_{\text{ind}(i,j)} \le x_{\text{ind}(i,k)} + x_{\text{ind}(j,k)}, \quad 1 \le i < j < k \le n \\
& x_{\text{ind}(i,j)} \ge 0, \quad 1 \le i < j \le n.
\end{aligned} \tag{2.3}$$

The following theorem ensures the existence of a solution and provides conditions for its uniqueness.

Theorem 2.1. *[29] The function $X \mapsto ||W \odot (X - D)||$ always attains its minimum on the convex cone defined by the distance matrices of the corresponding size. Moreover, every local minimum is a global minimum. If, in addition, the norm is strictly convex and the weight matrix has no zeros or infinities off its diagonal, then there is a unique global minimum.*

It is important to note that the ℓ_p norms are strictly convex for $1 < p < \infty$, which, following Theorem 2.1, ensures that the solution to (2.1) is unique. An example where using the ℓ_1 norm gives more than one solution will be seen later on Example 4.1.

Although it is possible to use any norm in the problem (2.1), in this work it is used the most commons, that is $p = 1, 2, \infty$. The problem can be interpreted as a linear program when $p = 1$, and it measures the total weighted change in the input matrix. It is a quadratic problem when $p = 2$. Lastly, $p = \infty$ also defines a linear problem that reflects the maximum absolute values over all pairs. The other ℓ_p norms serve as intermediates between these two extremes. Consequently, choosing a small p typically results in a solution with a few significant modifications to the original data, while a large p tends to produce a solution characterized by numerous minor adjustments. The advantages of using each common p will be seen in the following sections briefly, and more differences will be seen later on.

2.1 ℓ_1 norm

As it is used in many applications, for example with sensor locations in Gentile [17], problem (2.1) can be rewritten as a linear programming problem using the ℓ_1 norm by introducing slack variables. This modifies the objective function to be a linear function, which ensures the convexity of the problem.

Adapting the matrix X and the slack variables to their equivalent vectors, the problem can be written as follows:

$$\begin{array}{lll}
\underset{x\in\mathbb{R}^d, z\in\mathbb{R}^d}{\text{Minimize}} & \displaystyle\sum_{k=1}^{d} z_k & \\
\text{subject to} & w_{ij}(x_{\text{ind}(i,j)} - d_{ij}) \leq z_{\text{ind}(i,j)}, & 1 \leq i < j \leq n \\
& -w_{ij}(x_{\text{ind}(i,j)} - d_{ij}) \leq z_{\text{ind}(i,j)}, & 1 \leq i < j \leq n \\
& x_{\text{ind}(i,j)} \leq x_{\text{ind}(k,i)} + x_{\text{ind}(k,j)}, & 1 \leq k < i < j \leq n \\
& x_{\text{ind}(i,j)} \leq x_{\text{ind}(i,k)} + x_{\text{ind}(k,j)}, & 1 \leq i < k < j \leq n \\
& x_{\text{ind}(i,j)} \leq x_{\text{ind}(i,k)} + x_{\text{ind}(j,k)}, & 1 \leq i < j < k \leq n \\
& x_k \geq 0, & 1 \leq k \leq d.
\end{array} \tag{2.4}$$

It is noticeable that new constraints are added to the problem, resulting in

$$m = \frac{n(n-1)(n-2)}{2} + n(n-1).$$

2.2 ℓ_2 norm

Problem (2.1) may be used considering $p = 2$. This formulation of the problem is also known as the Euclidean norm.

$$\begin{array}{lll}
\underset{x\in\mathbb{R}^d}{\text{Minimize}} & \displaystyle\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} w_{ij}^2 (x_{\text{ind}(i,j)} - d_{ij})^2 & \\
\text{subject to} & x_{\text{ind}(i,j)} \leq x_{\text{ind}(k,i)} + x_{\text{ind}(k,j)}, & 1 \leq k < i < j \leq n \\
& x_{\text{ind}(i,j)} \leq x_{\text{ind}(i,k)} + x_{\text{ind}(k,j)}, & 1 \leq i < k < j \leq n \\
& x_{\text{ind}(i,j)} \leq x_{\text{ind}(i,k)} + x_{\text{ind}(j,k)}, & 1 \leq i < j < k \leq n \\
& x_k \geq 0, & 1 \leq k \leq d.
\end{array} \tag{2.5}$$

The objective function of (2.5), f_2, assumes a non-linear form. However, the number of constraints is maintained. The gradient of f_2 is

$$\nabla f_2(x) = \begin{bmatrix} w_{12}^2(2x_1 - 2a_{12}) \\ \vdots \\ w_{(n-1)n}^2(2x_d - 2a_{(n-1)n}) \end{bmatrix}$$

and the Hessian

$$\nabla^2 f_2(x) = diag(2w_{12}^2, \dots, 2w_{(n-1)n}^2).$$

2.3 ℓ_∞ norm

Finally, another interesting norm to study is the ℓ_∞ norm. The problem can be rewritten as

$$\begin{array}{lll} \underset{x\in\mathbb{R}^d, z\in\mathbb{R}}{\text{Minimize}} & z & \\ \text{subject to} & w_{ij}(x_{\text{ind}(i,j)} - d_{ij}) \le z, & 1 \le i < j \le n \\ & -w_{ij}(x_{\text{ind}(i,j)} - d_{ij}) \le z, & 1 \le i < j \le n \\ & x_{\text{ind}(i,j)} \le x_{\text{ind}(k,i)} + x_{\text{ind}(k,j)}, & 1 \le k < i < j \le n \\ & x_{\text{ind}(i,j)} \le x_{\text{ind}(i,k)} + x_{\text{ind}(k,j)}, & 1 \le i < k < j \le n \\ & x_{\text{ind}(i,j)} \le x_{\text{ind}(i,k)} + x_{\text{ind}(j,k)}, & 1 \le i < j < k \le n \\ & x_i \ge 0, & 1 \le k \le d. \end{array} \tag{2.6}$$

The problem is greatly simplified due to the objective function compared to the ℓ_2 norm, although again, the number of constraints is increased by $2d$ while the number of variables is increased only by one.

To illustrate the metric nearness problem and the differences when considering each norm, the following example is given.

Example 2.2. *Consider the dissimilarity matrix*

$$D = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 10 \\ 2 & 10 & 0 \end{bmatrix}.$$

It is clear that $10 \not\le 1 + 2$, *in other words, the values do not satisfy the triangle inequality. To find a distance matrix* X *that is the closest one to* D, *one can apply the* ℓ_1 *norm. One of the results* *expected would be matrix*

$$X_{\ell_1} = \begin{bmatrix} 0 & 8 & 2 \\ 8 & 0 & 10 \\ 2 & 10 & 0 \end{bmatrix}.$$

This result shows that the use of ℓ_1 norm changes drastically a single value and fixes the others since the objective function will be the sum of these changes.

On the other side, when using ℓ_∞ the result is the matrix

$$X_{\ell_\infty} = \begin{bmatrix} 0 & 3.33333 & 4.33333 \\ 3.33333 & 0 & 7.66666 \\ 4.33333 & 7.66666 & 0 \end{bmatrix}.$$

It demonstrates that the use of the ℓ_∞ norm distributes the change in all entries to decrease the objective function.

Between these two norms, the ℓ_2 norm gives the following result that balances the other two approaches,

$$X_{\ell_2} = \begin{bmatrix} 0 & 1.66666 & 2.16666 \\ 1.66666 & 0 & 3.83333 \\ 2.16666 & 3.83333 & 0 \end{bmatrix}.$$

Chapter 3

Mathematical Methods for Optimization

Before the experiments can be executed, some theory in ALGENCAN and Dykstra will be explored.

3.1 Algencan

Consider a general constrained optimization problem

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{Minimize}} & f(x) \\ \text{subject to} & h(x) = 0, \\ & g(x) \leq 0, \\ & \ell \leq x \leq u, \end{array} \tag{3.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^n \to \mathbb{R}^m$, and $g : \mathbb{R}^n \to \mathbb{R}^p$ are continuously differentiable functions. One can assume $\ell, u \in \mathbb{R}^n$, that is, $-\infty < \ell_i$ and $u_i < +\infty$ for $i = 1, \ldots, n$.

The following content was extracted from Birgin and Martínez [6] and it will be considered the Augmented Lagrangian method in the way analyzed in [1] and [5].

The method exhibits interesting global theoretical characteristics. First, any limit point corresponds to a stationary point of the problem of minimizing the infeasibility measure $\|h(x)\|^2 + \|g(x)_+\|^2$, subject to the bound constraints $\ell \leq x \leq u$. Conversely, any feasible limit point fulfills a sequential optimality condition. As a result, every feasible limit point is KKT-stationary, assuming only very mild constraint qualifications.

The augmented Lagrangian corresponding to problem (3.1) is given by

$$L_\rho(x,\lambda,\mu) = f(x) + \frac{\rho}{2}\left(\sum_{i=1}^{m}\left(h_i(x)+\frac{\lambda_i}{\rho}\right)^2 + \sum_{i=1}^{p}\left(\max\left\{0, g_i(x)+\frac{\mu_i}{\rho}\right\}\right)^2\right) \tag{3.2}$$

for all $x \in [\ell, u]$, $\rho > 0$, $\lambda \in \mathbb{R}^m$, and $\mu \in \mathbb{R}^p_+$.

Algorithm 1 presented below is a safeguarded Augmented Lagrangian approach, meaning that estimates of the Lagrange multipliers are computed at each iteration, but are ignored when updating the iterate if their sizes exceed user defined values denoted by $\lambda_{\min}, \lambda_{\max}$, and $\mu_{\max}$.

Algorithm 1 - [6]

Assume that $x^0 \in \mathbb{R}^n$, $\lambda_{\min} < \lambda_{\max}$, $\bar\lambda^1 \in [\lambda_{\min}, \lambda_{\max}]^m$, $\mu_{\max} > 0$, $\bar\mu^1 \in [0, \mu_{\max}]^p$, $\rho_1 > 0$, $\gamma > 1$, $0 < \tau < 1$, and a sequence $\{\varepsilon_k\}_{k=1}^{\infty}$ are given. Initialize $k \leftarrow 1$.

Step 1. Find $x^k \in [\ell, u]$ approximately solving:

$$\min_{x\in\mathbb{R}^n} \; L_{\rho_k}(x, \bar\lambda^k, \bar\mu^k) \quad \text{subject to } \ell \le x \le u, \tag{3.3}$$

satisfying

$$\left\| \; P_{[\ell,u]}\left(x^k - \nabla\, L_{\rho_k}(x^k, \bar\lambda^k, \bar\mu^k)\right) - x^k \right\| \le \varepsilon_k. \tag{3.4}$$

Step 2. Define

$$V^k = \min\left\{-g(x^k), \frac{\bar\mu^k}{\rho_k}\right\}.$$

If $k = 1$ or

$$\max\left\{\|h(x^k)\|, \|V^k\|\right\} \le \tau \max\left\{\|h(x^{k-1})\|, \|V^{k-1}\|\right\}, \tag{3.5}$$

chose $\rho_{k+1} = \rho_k$. Otherwise, define $\rho_{k+1} = \gamma\rho_k$.

Step 3. Compute

$$\lambda^{k+1} = \bar\lambda^k + \rho_k h(x^k), \qquad \mu^{k+1} = \left[\bar\mu^k + \rho_k g(x^k)\right]_+. \tag{3.6}$$

Computing $\bar\lambda^{k+1} \in [\lambda_{\min}, \lambda_{\max}]^m$ and $\bar\mu^{k+1} \in [0, \mu_{\max}]^p$. Set $k \leftarrow k+1$ and go to Step 1.

Algorithm 1 proceeds by approximately minimizing the Augmented Lagrangian function under the bound constraints, followed by updates to the penalty parameter and the Lagrange multipliers.

Test (3.5) evaluates progress in terms of feasibility and complementarity. If both metrics show improvement, the penalty parameter is considered sufficiently large and remains unchanged. Otherwise, it is increased by a factor $\gamma > 1$. The Lagrange multipliers λ^{k+1} and μ^{k+1}, corresponding to the current approximation x^{k+1}, are computed using (3.6) in Step 3. In the same step, the safeguarded values $\bar{\lambda}^{k+1}$ and $\bar{\mu}^{k+1}$ are also determined. Although theoretically these values do not need to correspond to λ^{k+1} and μ^{k+1}, in practice the procedure is as follows: if $\lambda^{k+1} \in [\lambda_{\min}, \lambda_{\max}]^m$ and $\mu^{k+1} \in [0, \mu_{\max}]^p$, then $\bar{\lambda}^{k+1} = \lambda^{k+1}$ and $\bar{\mu}^{k+1} = \mu^{k+1}$. Otherwise, λ^{k+1} and μ^{k+1} may be given by an arbitrary choice, such as projecting λ^{k+1} and μ^{k+1} onto the respective boxes or setting $\bar{\lambda}^{k+1} = 0$ and $\bar{\mu}^{k+1} = 0$, if $0 \geq \lambda_{\min}$.

The task of approximately minimizing $L_{\rho_k}(x, \bar{\lambda}^k, \bar{\mu}^k)$ over the set $[\ell, u]$, as described in (3.4), can always be solved. Since the set $[\ell, u]$ is compact, a global minimizer that satisfies (3.4) is guaranteed to exist. Furthermore, local optimization methods are capable of finding an approximate stationary point satisfying (3.4) in a finite number of iterations. Consequently, for any iterate x^k, the next iterate x^{k+1} is well-defined. Hence, Algorithm 1 produces an infinite sequence $\{x^k\}$, whose characteristics are described below.

1. Any limit point $x^* = \lim_{k \in K} x^k$ of the sequence satisfies the complementarity condition:

$$\mu_i^{k+1} = 0 \quad \text{whenever } g_i(x^*) < 0$$

for all sufficiently large $k \in K$.

2. Each accumulation point x^* of the sequence satisfies the first-order optimality conditions of the problem of minimizing the infeasibility metric under box constraints:

$$\min \|h(x)\|^2 + \|g(x)^+\|^2 \quad \text{subject to } \ell \leq x \leq u.$$

3. If for all $k \in \{1, 2, \dots\}$, x^k is an approximate global minimizer of $L_{\rho_k}(x, \bar{\lambda}^k, \bar{\mu}^k)$ over $[\ell, u]$ with a fixed tolerance $\eta > 0$, then every limit point of $\{x^k\}$ is a global minimizer of the infeasibility measure, subject to $\ell \leq x \leq u$. In this case, condition (3.4) is not required.

4. If for all $k \in \{1, 2, \dots\}$, x^k is an approximate global minimizer of $L_{\rho_k}(x, \bar{\lambda}^k, \bar{\mu}^k)$ over $[\ell, u]$ with a sequence of tolerances $\eta_k \to 0$, such that

$$L_{\rho_k}(x^k, \bar{\lambda}^k, \bar{\mu}^k) \leq L_{\rho_k}(x, \bar{\lambda}^k, \bar{\mu}^k) + \eta_k$$

for all $x \in [\ell, u]$, then each feasible limit point of $\{x^k\}$ is a global solution to the original constrained problem (3.1). As with the previous case, condition (3.4) is unnecessary.

5. If $\varepsilon_k \to 0$, then any feasible limit point $x^* = \lim_{k\in K} x^k$ generated by the algorithm satisfies the Approximate KKT (AKKT) condition:

$$\lim_{k\in K} \left\| P_{[\ell,u]} \left(x^k - \nabla f(x^k) - \nabla h(x^k)\lambda^{k+1} - \nabla g(x^k)\mu^{k+1} \right) - x^k \right\| = 0,$$

and

$$\lim_{k\in K} \max \left\{ \|h(x^k)\|_\infty, \| \min\{-g(x^k), \mu^{k+1}\}\|_\infty \right\} = 0.$$

All these properties proofs and additional information can be seen in [5]. These properties demonstrate that even when ε_k does not go to zero, Algorithm 1 converges to stationary points of the infeasibility measure

$$\|h(x)\|^2 + \|[g(x)]_+\|^2,$$

subject to the bounds $\ell \leq x \leq u$. Moreover, if $\varepsilon_k \to 0$, feasible limit points satisfy a sequential optimality condition. Therefore, under minimal constraint qualifications, feasible points satisfy the Karush-Kuhn-Tucker conditions.

Algencan is an implementation of the Augmented Lagrangian method designed to solve medium to large-scale constrained optimization problems with an optimized computer time. It is made aiming to solve nonlinear programming problems with a large number of variables and box constraints. The software is based on the theoretical foundations incorporated with computational strategies by Birgin and Martínez [5] using FORTRAN 90. The application also has many parameters that can be used to enhance performance as well as give some autonomy to the user.

Although it is particularly effective, Algencan comes from the Augmented Lagrangian theory, therefore, it uses first and second derivatives from the objective function and from the constraints. The program allows sparse Jacobian and Hessian matrices and stores them in an effective form. When requested by the user, the Algencan 3.1.1 also approximates the first and second derivatives using finite differences [4].

3.2 Dykstra Projection Method

If a set can be written as $\Omega = \cap_{i=1}^{p} \Omega_i$, where each Ω_i is a closed convex set in $\mathbb{R}^d$ and $\Omega \neq \emptyset$, then one can consider the optimization problem

$$\underset{x \in \Omega}{\text{Minimize}} \, ||x_0 - x||, \tag{3.7}$$

where x_0 is a given point and the inner product in the space is given by $||a||^2 = \langle a, a \rangle$. Therefore, the solution x^* is the projection of x_0 in Ω and can be denoted by $P_\Omega(x_0)$.

Dykstra's algorithm [13] solves the problem (3.7) in an effective way. It projects the iterate onto each convex set individually, completing a cycle of projections at every iteration. The process is repeated until a stop criteria is met. Besides the iterate, the algorithm generates another sequence with the increments $\{y_i^k\}_{k \in \mathbb{N}}$ with $i = 1, \ldots, p$.

Summarizing the Dykstra alternating projection method, setting $x_0^0 = x_0$ and $y_0^0 = 0$, the sequences are defined by

$$\begin{aligned} x_0^k &= x_p^{k-1}, \\ x_i^k &= P_{\Omega_i}(x_{i-1}^k - y_{i-1}^{k-1}), \quad i = 1, 2, \ldots, p, \\ y_i^k &= x_i^k - (x_{i-1}^k - y_{i-1}^{k-1}), \quad i = 1, 2, \ldots, p. \end{aligned} \tag{3.8}$$

It is important to remark that the increment y_i^{k-1} associated with Ω_i in the previous cycle is subtracted from x_i^{k-1} before projecting in Ω_i. Therefore, only the last y_i^k needs to be stored.

This projection method is related to Hildreth [20] and Han [19]. The Hildreth method can be interpreted as a projection method in some cases. More specifically, Hildreth's method solves quadratic programming problems with linear inequality constraints by iteratively dealing with the feasible region defined by each constraint.

On the other hand, the method described by Han is explicitly a projection method. It extends the idea of cyclic projection onto convex sets, using iterative schemes to solve a general optimization problem with convex constraints. Unlike Hildreth's method, which is specifically for quadratic programming.

Although there are similarities, the sequence generated by most alternating projection methods does not achieve the closest point in the intersection of the sets to the initial point. The advantage of Dykstra projection method relies in that the convergence to the closest point is guaranteed, as will be seen in Theorem 3.1. More about projection methods can be found in [14].

Theorem 3.1. ***(Boyle and Dykstra [8])*** *Let $\Omega_1, \ldots, \Omega_p$ be closed and convex sets of $\mathbb{R}^d$ such that*

$$\Omega = \bigcap_{i=1}^{p} \Omega_i \neq \emptyset.$$

For any $i = 1, 2, \ldots, p$ and any $x^0 \in \mathbb{R}^d$, the sequence $\{x_k\}$ generated by (3.8) converges to $x^* = P_\Omega(x^0)$, that is $\|x_k - x^*\| \to 0$ as $k \to \infty$.

The algorithm keeps iterating till a stopping criteria is achieved. Many implementations used the decreasing change of x, that is, the algorithm stops when

$$||x^{k-1} - x^k|| \leq \epsilon, \tag{3.9}$$

where ϵ is a small tolerance rate, expecting that there would not be any meaningful turn in the value of x anymore. However, accordingly to [7], there are some cases which (3.9) can occur even when the iterate is far from convergence, so the algorithm could stop the process prematurely, not solving the problem. Therefore, Birgin and Raydan [7] propose a robust stopping criteria monitoring the increments,

$$\sum_{i=1}^{p} \|y_i^{k-1} - y_i^k\|^2 \leq \epsilon. \tag{3.10}$$

The robust stop criteria guarantees better results without adding computational costs, and as a remark, it is known that Dykstra projection method has a linear convergence rate [14].

Chapter 4

Implementation and Results

4.1 Algencan on the Metric Nearness Problem

Although Algencan does not require dense matrices, it was crucial to create an index matrix containing triples of indices for every triangle inequality, specifically, a matrix with m columns, where each column represents a distinct combination of i, j, k. Later on, this triples matrix poses a challenge when executing Algencan as the size of m grows cubically.

The matrices D and W are read as vectors using the same set as the function *ind*(2.2). The objective function, along with its gradient and Hessian, is provided to Algencan according to each norm. As well as the constraints, its Jacobian and Hessian are given using the triples matrix, accounting the absolute value constraints when applicable. Algencan already stores these matrices, taking advantage of their sparsity.

4.1.1 Experiments

To conduct these experiments, it was used Algencan 3.1.1 obtained from TANGO[4]. The datasets used were generated with distance values between random points with a random noisy factor of the order of 10^{-2} created using Julia programming, a similar approach is made in [29]. This allows an initial analysis to understand the size of problems the method can solve and make computational changes as necessary.

Since there are other methods specifically designed for linear programming problems, it is natural to use Algencan for the ℓ_2-norm. The results can be seen in Table 4.1 where n represents the size of $D \in \mathbb{R}^{n \times n}$ and m is the number of restrictions. The graphic of time over the number of constraints is in Figure 4.1.

n	m	Iterations	Time(s)	Objective function
100	485100	11	0.80	0.678180
200	3940200	10	22.35	1.496289
300	13365300	13	129.23	2.344606
400	31760400	13	339.57	3.217657
500	62125500	23	981.21	4.052214
600	107460600	15	2677.58	4.925976
700	170765700	17	3011.04	5.824276
800	255040800	14	5252.16	6.693881
900	363285900	17	10547.64	7.544801
1000	498501000	19	14624.61	8.435415
1100	663686100	18	24759.46	9.321070
1128	715717128	44	551908.70	9.548081

Table 4.1: Experiments using Algencan applied to the metric nearness problem using ℓ_2-norm.

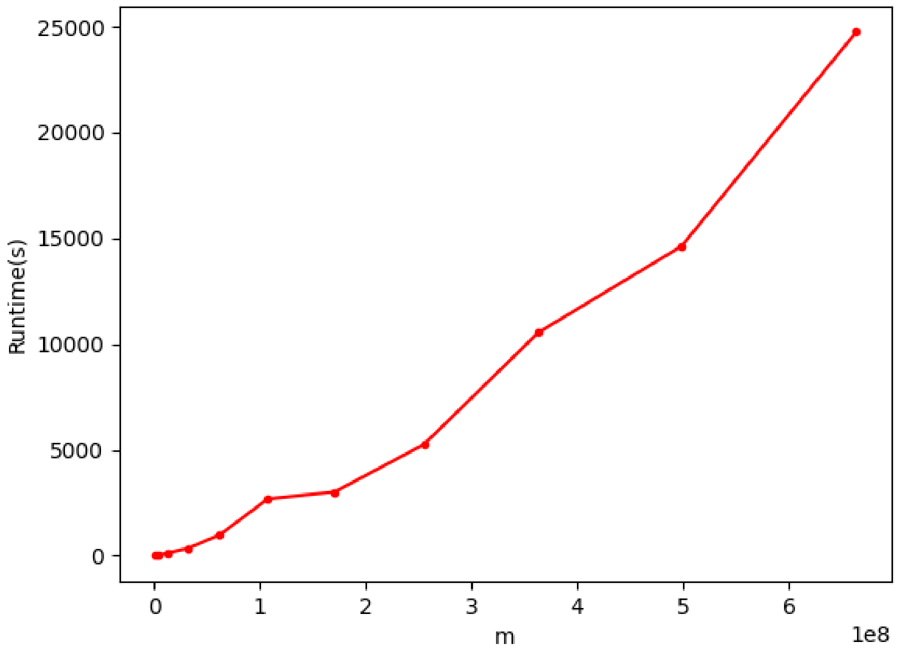

Figure 4.1: Runtime for problems with 100 to 1128 entry values using Algencan with ℓ_2-norm.

A question arises: why stop at $n = 1128$?

In Fortran 90, as well as in most programming languages, the largest representable integer using the default 4-byte (32-bit) `INTEGER` type must be less than 2^{31} = 2,147,483,648, since one bit is reserved for the sign. This means that integer values are typically limited to the range from −2,147,483,648 to 2,147,483,647. To work with larger integers, one must explicitly use a *kind* parameter that specifies a wider integer type, such as 64-bit integers.

In its internal architecture, Algencan defines all vectors and numerical values using default settings. In addition, Algencan requires the number of nonzero entries in the Jacobian matrix of the constraints. In the case of triangular inequalities, this number is three times m. When $n = 1128$, the problem has 715,717,128 constraints, resulting in 2,147,151,384 entries in the Jacobian matrix. Increasing n just by one more unit would exceed the computational limit.

Although Algencan has the capability of solving many types of problems, it is not made especially for linear cases since, as said before, there are already many algorithms that only solve that type of problem. This, combined with the large set of variables and constraints, can lead the program to no convergence. This happens because the norm of the gradient of the Augmented Lagrangian function has difficulty keeping a small value in a problem of this size. In order to use Algencan with some linear problems, it was needed to relax the optimality tolerance without losing the quality of the results by increasing the optimality tolerance in the parameters. The results for the ℓ_∞-norm can be seen in Table 4.2 and a graphic with the time on Figure 4.2. The use of the ℓ_1-norm resulted in slower performance due to the addition of $\frac{n(n-1)}{2}$ slack variables. Therefore this test was not conducted, given the excessive computational time required.

n	**m**	**Iterations**	**Time(s)**	**Objective function**
100	495000	9	0.74	0.049201
200	3980000	13	16.33	0.056041
300	13455000	10	34.19	0.088322
400	31920000	19	126.06	0.102599
500	62375000	15	197.31	0.101767

Table 4.2: Experiments using Algencan applied to the metric nearness problem using ℓ_∞-norm.

To find a method that would solve larger problems, Dykstra's method was studied.

4.2 Dykstra on the Metric Nearness Problem

4.2.1 Implementation details

One might think how to use the alternating projection method onto the metric nearness problem with ℓ_1 or ℓ_∞, however this will be described in the following sections, since it is only natural to begin with the ℓ_2 norm, given that the algorithm described so far is especifically for these problems.

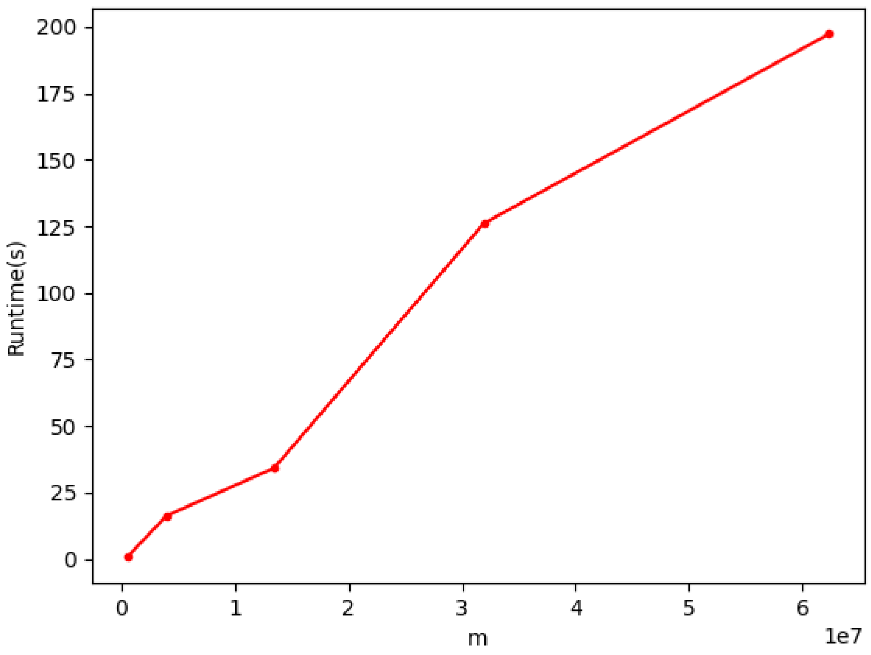

Figure 4.2: Runtime for problems with 100 to 500 entry values using Algencan to ℓ_∞-norm.

The objective function of the problem (2.5) is rewritten as

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (w_{ij} x_{\text{ind}(i,j)} - w_{ij} d_{ij})^2$$

and one can make a change of variables setting $x_{\text{ind}(i,j)} = w_{ij} x_{\text{ind}(i,j)}$, then also adapt $d_{ij} = w_{ij} d_{ij}$. The problem can be rewritten as

$$\begin{array}{lll}
\underset{x \in \mathbb{R}^d}{\text{Minimize}} & \displaystyle\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (x_{\text{ind}(i,j)} - d_{ij})^2 & \\
\text{subject to} & \dfrac{x_{\text{ind}(i,j)}}{w_{(i,j)}} \leq \dfrac{x_{\text{ind}(k,i)}}{w_{(k,i)}} + \dfrac{x_{\text{ind}(k,j)}}{w_{(k,j)}}, & 1 \leq k < i < j \leq n \\
& \dfrac{x_{\text{ind}(i,j)}}{w_{(i,j)}} \leq \dfrac{x_{\text{ind}(i,k)}}{w_{(i,k)}} + \dfrac{x_{\text{ind}(k,j)}}{w_{(k,j)}}, & 1 \leq i < k < j \leq n \\
& \dfrac{x_{\text{ind}(i,j)}}{w_{(i,j)}} \leq \dfrac{x_{\text{ind}(k,i)}}{w_{(k,i)}} + \dfrac{x_{\text{ind}(j,k)}}{w_{(j,k)}}, & 1 \leq i < j < k \leq n \\
& x_k \geq 0, & 1 \leq k \leq d,
\end{array}$$

as long as $w_{(j,k)} \neq 0$. With the intention of applying Dykstra's method, the problem is summarized to

$$\underset{x \in \Omega}{\text{Minimize}} \quad ||x - d||, \tag{4.1}$$

where

$$\Omega = \cap_{i=1}^{p} \Omega_i = \{x \in \mathbb{R}^n, Ax \leq b\},$$

and each Ω_i is a closed convex set in $\mathbb{R}^d$, defined by

$$\Omega_i = \{x \in \mathbb{R}^n, a_i x \leq b_i\}.$$

Since all the triangle inequalities constraints can be expressed as $a_i x \leq b_i$, with $A \in \mathbb{R}^{m\times d}$, $b \in \mathbb{R}^m$ and $m = \frac{1}{2}n(n-1)(n-2)$.

Assuming that the algorithm is on its $k-th$ iteration at the $i-th$ constraint, setting $r_{i-1}^k = x_{i-1}^k - y_{i-1}^{k-1}$, to perform the corection step. Then the projection of r_{i-1}^k in the halfspace Ω_i defined by $a_i x \leq b_i$ is

$$x_i^k = P_{\Omega_i}(x_{i-1}^k - y_{i-1}^{k-1}) = r_{i-1}^k - max\left\{\frac{\langle a_i, r_{i-1}^k\rangle - b_i}{||a_i||^2}, 0\right\} a_i. \tag{4.2}$$

Since

$$\langle a_i, r_{i-1}^k + \lambda a_i\rangle = b_i \Rightarrow \langle a_i, r_{i-1}^k\rangle + \lambda \|a_i\|^2 = b_i \Rightarrow \lambda = \frac{b_i - \langle a_i, r_{i-1}^k\rangle}{\|a_i\|^2}.$$

Using (3.8), one can deduce that $y_i^k = \max\left\{\frac{\langle a_i, r_{i-1}^k\rangle - b_i}{||a_i||^2}, 0\right\} a_i$, then update the increment and be ready to visit the next constraint. This process can be seen at the Algorithm 2.

Algorithm 2 - Dykstra projection method

Let $A \in \mathbb{R}^{m\times d}$, $b \in \mathbb{R}^m$ and $D = \mathbb{R}^{n\times n}$. Initialize $i \leftarrow 0$.

Step1. Define $y := 0 \in \mathbb{R}^m$ and $x_{ind(i,j)} := d_{i,j}$.

While not converged do

Step2. Visit constraints cyclically $i = i + 1$.

Step3. Perform the correction

$$x = x + y_i.$$

Step4. Perform the projection

$$x = x - max\left\{\frac{\langle a_i, r_{i-1}^k\rangle - b_i}{||a_i||^2}, 0\right\} a_i$$

Step5. Update the increment

$$y_i = max\left\{ \frac{\langle a_i, r_{i-1}^k \rangle - b_i}{||a_i||^2}, 0 \right\} a_i.$$

Although useful, the Algorithm 2 lacks in some aspects that will be addressed shortly.

The sparsity of A

One of the most important things to realize when implementing the Algorithm 2 is the sparsity of the constraint matrix A. Since each line corresponds to a triangle inequality, there are only three nonzero entries, $1, -1, -1$, at the entries $ind(i,j), ind(i,k)$ and $ind(k,j)$ respectively.

When operating the projection step, the only values of x that are altered are those of the nonzero entries of A. Afterwards, the correction step, the projection step, and even the increment update at the $i-th$ projection can be done only using these three entries.

Therefore, there is no need to store the matrix A. It is only necessary to acknowledge the positions $ind(i,j), ind(i,k)$ and $ind(k,j)$, as well as their signs.

The value of b

An improvement can be made by noticing that all the values of b, when considering just the triangle inequalities, are equal to zero.

The pattern of y

Initially, the sequence $\{y_k\}$ was viewed as a matrix $m \times 3$, where each row represents the increment of a constraint in that iteration.

$$Y = \begin{bmatrix} y_{11} & y_{12} & y_{13} \\ & \vdots & \\ y_{m1} & y_{m2} & y_{m3} \end{bmatrix}.$$

Therefore, the computational limit was given by the number of rows of Y, or, in other words, by the number of constraints in the problem:

$$m \leq 2^{31} - 1 \Rightarrow \frac{n(n-1)(n-2)}{2} \leq 2^{31} - 1 \Rightarrow n \leq 1626.5.$$

A pattern was noticed in the matrix Y. Each row was composed of a θ multiplied by $[1, -1, -1]$, that is, the non-zero elements of a row of the constraint matrix A. Then:

$$Y = \begin{bmatrix} 1\theta_1 & -1\theta_1 & -1\theta_1 \\ & \vdots & \\ 1\theta_m & -1\theta_m & -1\theta_m \end{bmatrix} \Rightarrow \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_m \end{bmatrix} \begin{bmatrix} 1 - 1 - 1 \end{bmatrix},$$

so that Y could be seen as a vector y.

Algorithm 3 - Dykstra projection method

Let $A \in \mathbb{R}^{m \times d}$, $b \in \mathbb{R}^m$ and $D = \mathbb{R}^{n \times n}$. Initialize $i \leftarrow 0$.

Step1. Define $y := 0 \in \mathbb{R}^M$, $x_{ind(i,j)} := d_{i,j}$ and $p := (i, j, k) \in \mathbb{N}^3$.

While not converged do

Step2. Visit constraints cyclically $i = i + 1$.

Step3. Perform the correction

$$x_p = x_p + y_i(1, -1, -1).$$

Step4. Perform the projection

$$x_p = x_p - max\left\{\frac{\langle(1,-1,-1), r_{i-1}^k\rangle}{3}, 0\right\}(1, -1, -1)$$

Step5. Update the increment

$$y_i = max\left\{\frac{\langle(1,-1,-1), r_{i-1}^k\rangle}{3}, 0\right\}.$$

The same random problems used in Algencan were applied to Algorithm 3 implemented in Fortran 90 using the tolerance criteria equals to 10^{-8}. The results can be seen in Table 4.3 and a comparison in Figure 4.3. The details of this implementation, along with all other codes described here, can be found in https://github.com/JuliaGuizardi/Metric-Nearness-Problem.

n	**m**	**Iterations**	**Time(s)**	**Objective function**
100	485100	28	0.05	0.678180
200	3940200	55	0.70	1.496289
300	13365300	86	3.68	2.344605
400	31760400	121	12.37	3.217656
500	62125500	140	28.21	4.052213
600	107460600	188	65.84	4.925975
700	170765700	239	133.14	5.824276
800	255040800	303	247.77	6.693881
900	363285900	347	404.48	7.544800
1000	498501000	422	686.03	8.435415

Table 4.3: Experiments using Dykstra applied to the metric nearness problem using ℓ_2-norm.

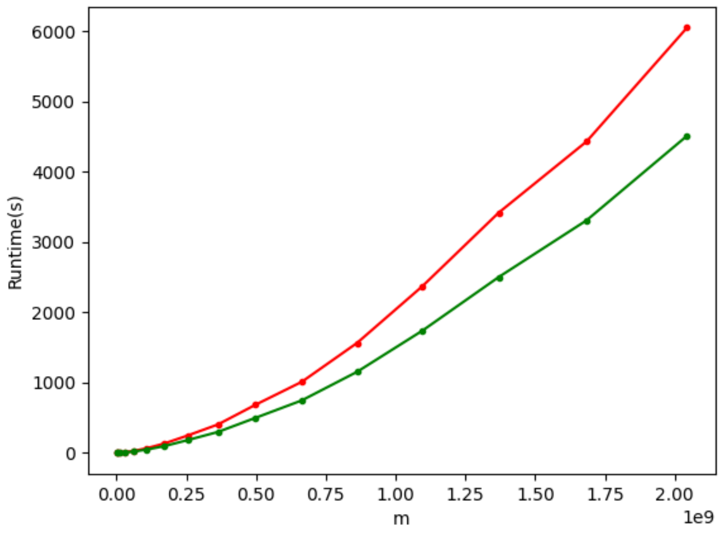

Figure 4.3: Graph comparing the runtime of the algorithm before (red) and after (green) the improvements described.

Considering all those upgrades in the Algorithm 2, the Algorithm 3 is developed.

Although the previous Algorithm 3 seems robust, there is space for one more improvement. At the update of the increment, there is room for zero entries in the vector y. To check the percentage of non-zero values of y, the experiments seen in Table 4.3 were recreated but calculating the average percent of non-zero entries in y. The results can be seen in Table 4.4.

This shows that the amount of zero entries in y must be noticed in order to solve bigger problems, considering that the size of y is of the order of n^3 and Table 4.4 shows that this number can be drastically reduced.

To solve larger problems, the vector y began to be seen sparsely. For this purpose, a vector

n	**m**	**Average of non-zero y(%)**
100	485100	0.294
200	3940200	0.176
300	13365300	0.126
400	31760400	0.100
500	62125500	0.082
600	107460600	0.070
700	170765700	0.061
800	255040800	0.054
900	363285900	0.048
1000	498501000	0.044

Table 4.4: Experiments using Dykstra applied to the metric nearness problem using ℓ_2-norm considering the average of non-zero entries in y.

ind_y was created with the indices of the non-zero increments. So,

$$y = \begin{bmatrix} \theta_i \\ \theta_j \\ \vdots \end{bmatrix} \quad \text{e} \quad ind_y = \begin{bmatrix} i \\ j \\ \vdots \end{bmatrix}.$$

Another problem is created: How to recognize and access the values of y to make the correction step? At this moment, some strategies were approached.

The first technique tried was to use Binary Search. Binary search is a fundamental algorithm for locating an element in a sorted array by repeatedly dividing the search space in half. At each step, the algorithm compares the target value with the middle element of the array. If the target is smaller, the search continues in the left half, otherwise, it proceeds in the right half. This process repeats until the element is found or there is no more room to search. The Binary Search has a time complexity of $O(\log n)$, and, it is significantly faster than linear search for large datasets.

Unfortunately, since at each iteration the binary search would be used m times, and most of them would go to the end and not find a match, this resulted in a slower version of the program.

Another solution that was considered briefly was using a text file to store the values and the corresponding indices. As expected, it also increased the time considerably.

To truly solve larger problems, two more vectors were created: z and ind_z. At each iteration, y and ind_y store the non-zero entries of y from the previous iteration and their respective indices. While z and ind_z store the new values of y and their respective indices.

At iteration k, when projection i is executed, the algorithm simply check if the next element in ind_y is equal to i. At the end of the iteration, the new values of the increment, together with the indices, are copied to the vectors storing the old values. A minor problem had to be overcome: although the vector y is smaller in size, the number of indices within the vectors needed to be larger. To achieve this, integer vectors were created using the ISO_FORTRAN_ENV package, which enables the storage of larger numbers.

This method, combined with the Algorithm 3, creates a fast and efficient method to solve the problem (4.1). The results of applying this method can be seen in Table 4.5 and a comparison with previous Dykstra methods in Figure 4.4. With this modification, the number of constraints is no longer a limitation without compromising runtime.

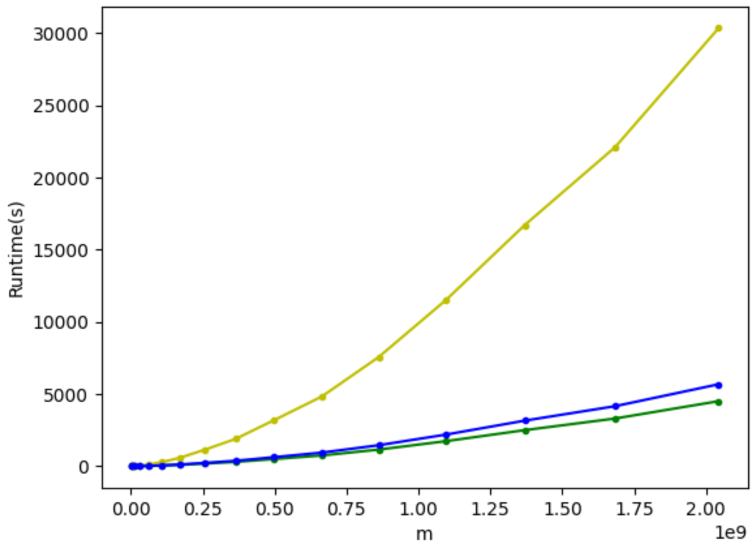

Figure 4.4: Graph comparing the runtime of the algorithm with binary search (yellow), y as a dense vector (green), and y as a sparse vector (blue).

n	**m**	**Iterations**	**Time(s)**	**Objective function**
100	485100	28	0.03	0.678180
200	3940200	55	0.60	1.496289
300	13365300	86	3.18	2.344605
400	31760400	121	10.80	3.217656
500	62125500	140	24.76	4.052213
600	107460600	188	58.65	4.925975
700	170765700	239	119.87	5.824276
800	255040800	303	228.88	6.693881
900	363285900	347	375.75	7.544800
1000	498501000	422	628.73	8.435415

Table 4.5: Experiments using Dykstra applied to the metric nearness problem using ℓ_2-norm.

The greatest advantage of using just the non-zero y is when compared to Algencan. Since y is the only object that stores the constraints in Dykstra method, by reducing its size, the size of problems that the algorithm solves increases compared to Algencan, once the number of constraints is not a limitation anymore. More experiments were made with up to 10^{10} constraints, the results can be seen in Table 4.6. A comparison of the value of n against the time and the average size of y can be seen on Figures 4.5 and 4.6.

n	**m**	**Iterations**	**Time(s)**	**Objective function**	**Avg non-zero y(%)**
1100	663686100	474	943.06	9.321069	0.040
1200	861841200	563	1456.37	10.219121	0.037
1300	1095966300	666	2193.51	11.087712	0.034
1400	1369061400	766	3158.67	11.990787	0.032
1500	1684126500	819	4162.08	12.853625	0.030
1600	2044161600	916	5682.09	13.721075	0.028
1700	2452166700	1052	7895.20	14.621010	0.027
1800	2911141800	1093	9815.24	15.476243	0.025
1900	3424086900	1242	13210.28	16.337093	0.024
2000	3994002000	1438	18037.19	17.250960	0.023
2100	4623887100	1509	22246.14	18.092716	0.022
2200	5316742200	1676	28450.17	18.996772	0.021
2300	6075567300	1695	33179.05	19.872769	0.020
2400	6903362400	1854	41727.74	20.718868	0.019
2500	7803127500	2135	55100.22	21.645585	0.018
2600	8777862600	2209	64883.44	22.512863	0.018
2700	9830567700	2464	82518.99	23.389737	0.017
2800	10964242800	2431	92079.40	24.241190	0.016
2900	12181887900	2735	117435.47	25.134888	0.016
3000	13486503000	2897	140165.40	25.983862	0.015

Table 4.6: Experiments using Dykstra applied to the metric nearness problem using ℓ_2-norm.

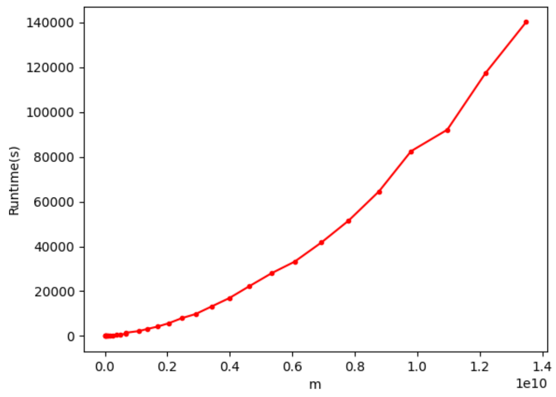

Figure 4.5: Runtime for problems with n =100 to 3000 using Dykstra to ℓ_2-norm.

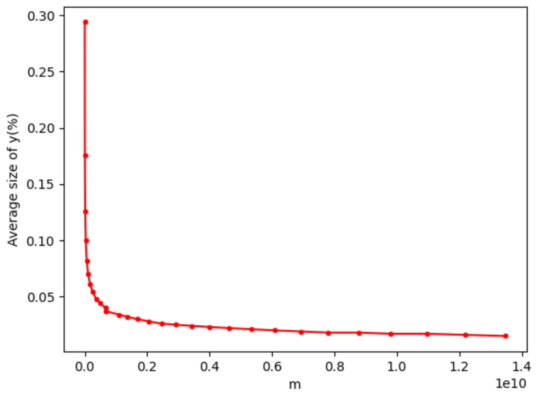

Figure 4.6: Average size of y for problems with n =100 to 3000 using Dykstra to ℓ_2-norm.

Even on the experiment with the largest n, the average size of the vector y was 2.125.332, which is still a thousand times smaller than the previous computer limit. This occurs because, although effective, Algencan is not specifically designed for this problem as the Dykstra projection method was implemented here exclusively for the metric nearness problem.

4.2.2 Dykstra's method applied to generalized quadratic problems

Let $A \in \mathbb{R}^{m\times d}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^d$, $\gamma > 0$, and $\Gamma \in \mathbb{R}^{d\times d}$ be positive definite diagonal matrix. Consider the following quadratic optimization problem:

$$\begin{aligned} \underset{x\in\mathbb{R}^d}{\text{Minimize}} \quad & Q(x) = \frac{1}{2}x^T\Gamma x + c^T x \\ \text{subject to} \quad & Ax \le b. \end{aligned} \tag{4.3}$$

The corresponding dual problem is formulated as [31]:

$$\begin{aligned} \underset{y\in\mathbb{R}^m}{\text{Maximize}} \quad & D(y) = -b^T y - \frac{1}{2}\left\|A^T y + c\right\|^2_{\Gamma^{-1}} \\ \text{subject to} \quad & y \le 0. \end{aligned} \tag{4.4}$$

The optimal value is found when the Karush-Kuhn-Tucker (KKT) conditions are met. That is, the solutions are the vectors x^* and y^* such that:

$$\textbf{(1)}\ \Gamma x^* = -A^T y^* - c \tag{4.5}$$

$$\textbf{(2)}\ Ax^* \le b \tag{4.6}$$

$$\textbf{(3)}\ y^{*T}(Ax^* - b) = 0 \tag{4.7}$$

$$\textbf{(4)}\ y^* \ge 0 \tag{4.8}$$

If these four conditions hold, then $y^* = \arg\max D(y)$ and $x^* = \arg\min Q(x)$ with $D(y^*) = Q(x^*)$. In a more general case, if a vector x satisfies $Ax \le b$ and $y \ge 0$, then the inequality

$$D(y) \le D(y^*) = Q(x^*) \le Q(x)$$

holds.

To apply Dykstra's method to the problem (4.3), first, it is considered a weighted norm, so given arbitrary vectors $f, g \in \mathbb{R}^d$,

$$\langle f, g\rangle_\Gamma = f^T \Gamma g$$

$$\|f\|_\Gamma^2 = \langle f, f\rangle_\Gamma = f^T \Gamma f.$$

Then, one can produce the following equivalence

$$\frac{1}{2}\|x + \Gamma^{-1}c\|_\Gamma^2 = \frac{1}{2}x^T\Gamma x + x^T\Gamma\Gamma^{-1}c + \frac{1}{2}c^T\Gamma^{-1}c$$

$$\Leftrightarrow \frac{1}{2}\|x + \Gamma^{-1}c\|_\Gamma^2 = \frac{1}{2}x^T\Gamma x + c^T x.$$

Taking $z = -\Gamma^{-1}c$, the problem (4.3) can be approximate by the problem:

$$x^* = \arg\min_{x\in C} \|x - z\|_\Gamma^2, \tag{4.9}$$

where $C = \cap_{i=1}^m C_i$ are the half-space constraints:

$$C_i = \{x \in \mathbb{R}^d : a_i^T x \leq b_i\} = \{x \in \mathbb{R}^d : \langle \hat{a}_i, x\rangle_\Gamma \leq b_i\},$$

considering that $\hat{a}_i = \Gamma^{-1}a_i$, consequently

$$\langle \hat{a}_i, x\rangle_\Gamma = \hat{a}_i^T\Gamma x = a_i^T\Gamma^{-1}\Gamma x = a_i^T x.$$

An important remark is that, according to equation (4.9), the initial value for x must be z, meaning $x_0 = -\Gamma^{-1}c$. Based on this, Algorithm 4 is constructed. This method is also equivalent to the previously described Hildreth method [20] when applied to quadratic programming.

Algorithm 4 - Dykstra projection method to Quadratic Programming

Let $A \in \mathbb{R}^{m\times d}$, $b \in \mathbb{R}^m, c \in \mathbb{R}^d$ and $\Gamma = \mathbb{R}^{d\times d}$. Initialize $i \leftarrow 0$.

Step1. Define $y := 0 \in \mathbb{R}^m$, $x_{ind(i,j)} := -\Gamma^{-1}c$.

While not converged do

Step2. Visit constraints cyclically $i = i + 1$.

Step3. Perform the correction

$$x = x + y_i(\Gamma^{-1}a_i).$$

Step4. Perform the projection

$$x = x - \frac{\max\{a_i^{-1}x - b_i, 0\}}{a_i^T\Gamma^{-1}a_i}(\Gamma^{-1}a_i)$$

Step5. Update the increment

$$y_i = \frac{\max\{a_i^{-1}x - b_i, 0\}}{a_i^T\Gamma^{-1}a_i}.$$

Optimality conditions

Assuming that the Algorithm 4 is in the $k-th$ iteration, that is after k sions, it is obtained x_k and y_k, the primal and dual, respectively. The sequences are converging to the optimal solutions $\hat{x}$ and $\hat{y}$ such that $D(\hat{y}) = Q(\hat{x})$. The KKT optimality conditions for quadratic programming seen in equations (4.5) - (4.8) affirm that x^* and y^* are optimal solutions for the primal (4.3) and the dual (4.4) if and only if:

$$(1) Ax \leq b \qquad (2) y^T(Ax - b) = 0 \qquad (3) \Gamma x = -A^T y - c \qquad (4) y \geq 0.$$

Step 5 shows that the fourth condition is automatically satisfied for any y_k.

Now observe that the third condition holds for the initial $x_0 = -\Gamma^{-1}c$ and $y_0 = 0$. Let's assume that it also holds for a random iteration k, that is

$$\Gamma x_k = -A^T y_k - c,$$

then

$$x_k = \Gamma^{-1}(-A^T y_k - c),$$

since Γ is inversible. According to Step 4 and defining $\delta_k - \dfrac{\max\{a_i^{-1}x_k - b_i, 0\}}{a_i^T\Gamma^{-1}a_i}$,

$$x_{k+1} = x_k - \delta_k(\Gamma^{-1}a_i)$$

$$\Rightarrow x_{k+1} = \Gamma^{-1}(-A^T y_k - c) - \delta_k(\Gamma^{-1}a_i)$$

$$\Rightarrow x_{k+1} = \Gamma^{-1}(-A^T y_k - c - \delta_k a_i).$$

Note that

$$-A^T y_k - c - \delta_k a_i = -A^T(y_k + e_i\delta_k) - c = -A^T y_{k+1} - c.$$

Consequently, the third condition is also always satisfied. This means that y_k is always feasible for the dual function (4.4) and

$$D(y_k) = -b^T y_k - \frac{1}{2}(A^T y_k + c)^T\Gamma^{-1}(A^T y_k + c) = -b^T y_k - \frac{1}{2}x_k^T\Gamma x_k.$$

Following the projections of all constraints set, the primal-dual gap ω_k is checked, providing information on how close the algorithm is to convergence.

$$\omega_k = \frac{D(y_k) - Q(x_k)}{D(y_k)}. \tag{4.10}$$

4.2.3 Dykstra's method applied to linear problems

The ℓ_1 and ℓ_∞ problems, respectively (2.4) and (2.6) can be simplified as following:

$$\begin{aligned} \underset{x}{\text{Minimize}} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b, \end{aligned} \tag{4.11}$$

where x is the vector x attached with the slack variables, A is a sparse matrix and c will have zeros or ones, as requested by the position. The problem (4.11) can be approximated by the quadratic function,

$$\begin{aligned} \underset{x}{\text{Minimize}} \quad & c^T x + \frac{1}{2\gamma} x^T \Gamma x \\ \text{subject to} \quad & Ax \leq b, \end{aligned} \tag{4.12}$$

with Γ being a diagonal matrix with positive diagonal entries and $\gamma > 0$. Following [31] and [24], when Γ is the identity matrix there exists one $\gamma_0 > 0$ such that for every $\gamma > \gamma_0$, the optimal solution of (4.12) is equivalent to optimal solution with the minimum Euclidean norm of (4.11).

The dual of (4.12) is

$$\begin{aligned} \underset{y \in \mathbb{R}^m}{\text{Maximize}} \quad & -b^T y - \frac{1}{2\gamma}(A^T y + c)^T \Gamma^{-1} (A^T y + c) \\ \text{subject to} \quad & y \leq 0. \end{aligned} \tag{4.13}$$

One might notice the difference between equations (4.12) and (4.13) with the primal and dual equations displayed in the previous section is $\gamma > 0$. This is added to minimize the effect that the quadratic term in (4.12) has in the approximation the the linear form (4.11), since as γ increases, the approximation improves. The value of γ is added to the Algorithm 4.

Example 4.1. *In order to try the method, an example was created. A convex set $\Omega \in \mathbb{R}^2$ was defined using the intersection of semi planes, as can be seen in Figure 4.7. A point outside Ω can be projected using the ℓ_1 norm onto Ω, by applying the method described in the problem (4.14). One example can be seen in Figure 4.8.*

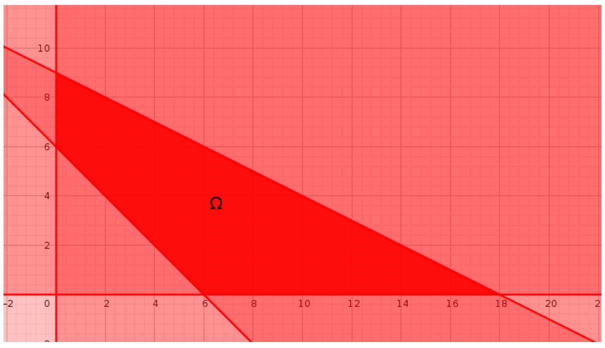

Figure 4.7: Convex set Ω.

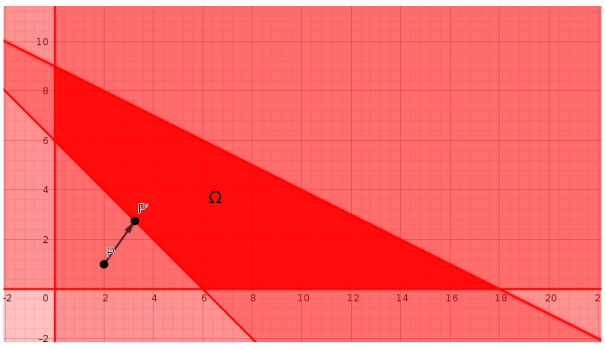

Figure 4.8: Projection of $P = (2, 1)$ onto Ω.

$$\begin{aligned} \underset{x \in \mathbb{R}^2}{\text{Minimize}} \quad & |x_1 - p_1| + |x_2 - p_2| \\ \textit{subject to} \quad & x_1 + x_2 \leq -6 \\ & x_1 + 2x_2 \leq 18 \\ & x_1, x_2 \geq 0 \end{aligned} \quad . \tag{4.14}$$

The problem (4.14) is converted to a linear problem by adding slack variables,

$$\begin{aligned} \underset{x\in\mathbb{R}^2, z\in\mathbb{R}^2}{\text{Minimize}} \quad & z_1 + z_2 \\ \textit{subject to} \quad & x_1 + x_2 \leq -6 \\ & x_1 + 2x_2 \leq 18 \\ & x_1 - p_1 \leq z_1 \\ & -x_1 + p_1 \leq z_1 \\ & x_2 - p_2 \leq z_2 \\ & -x_2 + p_2 \leq z_2 \\ & x_1, x_2 \geq 0 \end{aligned} \quad . \tag{4.15}$$

Using $\gamma = 100$ and $\Gamma = I_4$, the results on Table 4.7 were obtained. It is important to remember that the projected points are in $\mathbb{R}^2$ but as seen in problem (4.15), when the slack variables are added, the variables of the problem became x and z, then the Euclidean norm said on Table 4.7 is consistent with that.

Point	**Projection of x**	z	**Objective function**	**Euclidean norm**
(0,0)	(3,3)	(3,3)	6	6
(2,1)	(3.25,2.75)	(1.25,1.75)	3	4.77
(3,1)	(3.5,2.5)	(0.5,1.5)	2	4.58
(10,20)	(10,4)	(0,16)	16	19.28

Table 4.7: Projected points in Ω using ℓ_1-norm by Dykstra projection method.

The same problem was applied to Simplex using the package JuMP in Julia Language in order to compare the results. The outcome can be seen in Table 4.8. One can notice that the values of the projected point, in most cases, are different from Dykstra to Simplex, although the value of the objective function stays the same. This happens because the norm ℓ_1 is not strictly convex, so according to Theorem 2.1, there is no guarantee of a unique solution. In addition to that, while the Simplex method searches by the corners of the feasible set, Dykstra, applied to linear cases, finds the solution with the minimum Euclidean norm.

Point	**Projection of x**	z	**Objective function**	**Euclidean norm**
(0,0)	(6,0)	(6,0)	6	8.48
(2,1)	(5,1)	(3,0)	3	5.91
(3,1)	(5,1)	(2,0)	2	5.47
(10,20)	(10,4)	(0,16)	16	19.28

Table 4.8: Projected points in Ω using ℓ_1-norm by Simplex.

Even though this example shows a good usage of the Dykstra projection method applied to linear function one remark need to be made, when γ is set equals two the result obtained by the projection of $p = (10, 20)$ is $x = (4, 7)$ with an object function equals to 19, which is not a minimum of the linear problem, it only achieves a real solution when γ is big enough.

Choosing γ

A new question emerges: how to define γ? If one sets γ too high the convergence rate will be slow, on the contrary, a small γ might converge to a non-optimal solution of the quadratic problem 4.12. In [31], the following Theorem is given:

Theorem 4.1. *Let $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^d$, and $\mathscr{A} = \{x \in \mathbb{R}^d : Ax \leq b\}$. Denote $\hat{x} = \arg\min_x c^T x$ and assume that all entries of $\hat{x}$ are between 0 and $B > 0$. Let Γ be a diagonal matrix with entries $\Gamma_{ii} = c_i > 0$ and let $x^* = \arg\min_x [c^T x + (1/2\gamma)x^T \Gamma x]$. Then*

$$c^T\hat{x} \leq c^T x^* \leq c^T \hat{x}\left(1 + \frac{B}{2\gamma}\right).$$

Although Theorem 4.1 creates a boundary for the optimal solution of the quadratic problem, it is still dependent on the size of γ. In order to understand how the value of γ affects the final solution of the metric nearness problem, some experiments were made using small random problems and comparing the results with the Simplex method. The problems were generated as before.

In the Table 4.9, the problem has only 24 constraints and the value of γ used varies from 0.1 to 100. From $\gamma = 2$, the result of the linear objective function is the same as the value given by Simplex, being 0.00 in just two iterations.

n=4 and m=24					
γ	**Iterations**	**Time(s)**	**Infeasibility**	**Linear Function**	**Dual Gap (ω)**
0.1	1	0.00	1.1×10^{-16}	1.703170	3.3×10^{1}
1.0	2	0.00	1.1×10^{-16}	0.068199	0.0
2.0	2	0.00	1.1×10^{-16}	0.000000	-5.8×10^{-16}
10.0	2	0.00	1.7×10^{-15}	0.000000	6.7×10^{-15}
20.0	2	0.00	2.1×10^{-15}	0.000000	2.2×10^{-14}
100.0	2	0.00	1.4×10^{-14}	0.000000	9.2×10^{-13}

Table 4.9: Experiments with $n = 4$ and $m = 24$ using Dykstra applied to the metric nearness problem with the ℓ_1-norm.

Using $n = 8$, the same tests were made. The solution to the objective function given by Simplex was 0.000842. This value was achieved by $\gamma = 10, 20, 100$ as it is seen in Table 4.10. It is also possible to see an increase in the maximum infeasibility when $\gamma = 100$.

n=8 and m=224					
γ	**Iterations**	**Time(s)**	**Infeasibility**	**Linear Function**	**Dual Gap (ω)**
0.1	2	0.00	2.2×10^{-16}	5.882090	0.0
1.0	2	0.00	2.2×10^{-16}	0.093019	-5.7×10^{-16}
2.0	2962	0.01	8.4×10^{-9}	0.000843	1.5×10^{-9}
10.0	31443	0.12	8.6×10^{-9}	0.000842	1.6×10^{-8}
20.0	67044	0.28	9.3×10^{-9}	0.000842	3.8×10^{-8}
100.0	351855	1.46	1.0×10^{-8}	0.000842	2.1×10^{-7}

Table 4.10: Experiments with $n = 8$ and $m = 224$ using Dykstra applied to the metric nearness problem with the ℓ_1-norm.

When $n = 16$, the Simplex solution was 0.250751. The results seen in Table 4.11 show that this value was only achieved by $\gamma = 10$ and $\gamma = 20$, it is possible to see a reduction of the linear function when $\gamma = 100$, but the maximum infeasibility increases significantly.

n=16 and m=1920					
γ	**Iterations**	**Time(s)**	**Infeasibility**	**Linear Function**	**Dual Gap (ω)**
0.1	2	0.00	2.2×10^{-16}	33.146066	-1.5×10^{-16}
1.0	2491	0.03	8.4×10^{-9}	0.825414	5.8×10^{-11}
2.0	14303	0.17	8.7×10^{-9}	0.251842	3.5×10^{-10}
10.0	108800	1.35	1.0×10^{-8}	0.250751	2.8×10^{-9}
20.0	226923	2.84	8.6×10^{-9}	0.250751	4.7×10^{-9}
100.0	1000001	12.63	2.5×10^{-4}	0.250497	3.9×10^{-4}

Table 4.11: Experiments with $n = 16$ and $m = 1920$ using Dykstra applied to the metric nearness problem with the ℓ_1-norm.

Between the values of γ experimented with $n = 32$, the only one that achieved the same result as Simplex was $\gamma = 20$, with this value being 1.282093, as viewed in Table 4.12.

n=32 and m=15872					
γ	**Iterations**	**Time(s)**	**Infeasibility**	**Linear Function**	**Dual Gap (ω)**
0.1	40	0.00	1.4×10^{-9}	110.793939	2.5×10^{-13}
1.0	6234	0.58	8.5×10^{-9}	2.383258	4.1×10^{-11}
2.0	9670	0.89	9.9×10^{-9}	1.350396	9.5×10^{-11}
10.0	102495	9.38	9.9×10^{-9}	1.282093	1.3×10^{-9}
20.0	218996	20.07	1.0×10^{-8}	1.282093	3.0×10^{-9}
100.0	1000001	94.59	1.1×10^{-4}	1.281902	7.8×10^{-5}

Table 4.12: Experiments with $n = 32$ and $m = 15872$ using Dykstra applied to the metric nearness problem with the ℓ_1-norm.

With $n = 64$, the result given by Simplex was 7.643207 would be somewhere between $\gamma = 20$ and $\gamma = 100$, although an important remark at this moment is that when $\gamma = 100$ the code achieved the maximum iterations, 1000000. The results are in Table 4.13.

n=64 and m=129024					
γ	**Iterations**	**Time(s)**	**Infeasibility**	**Linear Function**	**Dual Gap (ω)**
1.0	14836	9.70	1.0×10^{-8}	13.916846	1.9×10^{-11}
2.0	18559	12.49	1.0×10^{-8}	8.232122	-3.9×10^{-11}
10.0	130915	86.75	1.0×10^{-8}	7.646193	-1.4×10^{-10}
20.0	129768	83.93	1.0×10^{-8}	7.643494	-1.3×10^{-9}
100.0	1000001	674.38	2.4×10^{-4}	7.640805	1.6×10^{-4}
200.0	1000001	672.51	2.4×10^{-4}	7.640805	1.6×10^{-4}

Table 4.13: Experiments with $n = 64$ and $m = 129024$ using Dykstra applied to the metric nearness problem with the ℓ_1-norm.

Again, no values of γ tested returned the same value for the linear function as Simplex, 23.456935, when $n = 100$ as seen in Table 4.14. Unfortunately, for all $\gamma \geq 10$ the algorithm reached the maximum iterations without convergence.

n=100 and m=495000					
γ	**Iterations**	**Time(s)**	**Infeasibility**	**Linear Function**	**Dual Gap (ω)**
1.0	256897	614.65	1.0×10^{-8}	37.674170	1.7×10^{-11}
2.0	490980	1168.09	1.0×10^{-8}	25.681395	8.4×10^{-12}
10.0	1000001	2405.04	4.0×10^{-7}	23.486744	-1.3×10^{-8}
20.0	1000001	2340.36	3.5×10^{-6}	23.459550	4.5×10^{-7}
100.0	1000001	2362.39	8.1×10^{-5}	23.453494	5.9×10^{-5}
200.0	1000001	2349.36	4.8×10^{-4}	23.449053	1.6×10^{-4}

Table 4.14: Experiments with $n = 100$ and $m = 495000$ using Dykstra applied to the metric nearness problem with the ℓ_1-norm.

n=200 and m=3980000					
γ	**Iterations**	**Time(s)**	**Infeasibility**	**Linear Function**	**Dual Gap (ω)**
1.0	115596	2291.51	1.0×10^{-8}	188.807398	2.6×10^{-12}
2.0	567334	9719.63	1.0×10^{-8}	116.823596	-1.7×10^{-12}
10.0	1000001	19618.20	4.5×10^{-6}	106.649968	3×10^{-8}
20.0	1000001	31030.83	1.1×10^{-5}	106.460323	1.5×10^{-7}
100.0	1000001	20088.46	6.3×10^{-5}	106.392289	1.4×10^{-5}
200.0	1000001	12039.82	2.1×10^{-4}	106.378501	7.7×10^{-5}

Table 4.15: Experiments with $n = 200$ and $m = 3980000$ using Dykstra applied to the metric nearness problem with the ℓ_1-norm.

A resume of the ℓ_1 norm can be seen on Figure 4.9 and 4.10 considering time and number of iterations with different values of γ and $n = 32, 64, 100$ and 200.

The same tests were made using the ℓ_∞ norm. The results can be seen in Tables 4.16-4.22. The optimal solution to the linear function according to Simplex method is displayed with the other values. A conclusion with the ℓ_∞ norm can be seen in Figure 4.11 and 4.12 again considering the time and number of iterations with different values of γ and $n = 32, 64, 100$ and 200.

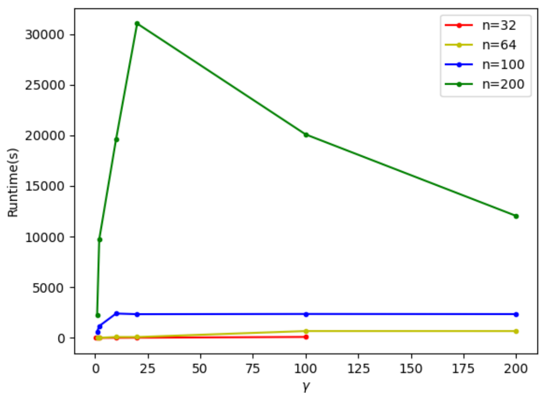

Figure 4.9: Runtime for problems with 32 to 200 entry values using Dykstra to ℓ_1-norm with various γ values.

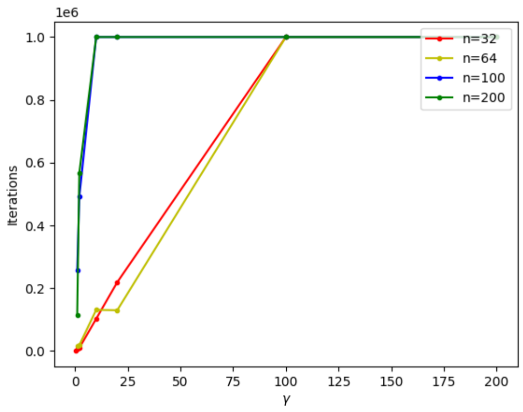

Figure 4.10: Average size of y for problems with 32 to 200 entry values using Dykstra to ℓ_1-norm with various γ values.

n=4 and m=24					
Simplex Objective Function: 0.0					
γ	**Iterations**	**Time(s)**	**Infeasibility**	**Linear Function**	**Dual Gap (ω)**
0.1	29	0.00	2.9×10^{-9}	0.724265	-1.9×10^{-9}
1.0	38	0.00	3.9×10^{-9}	0.424265	-2.2×10^{-9}
2.0	73	0.00	9.3×10^{-9}	0.213217	-1.6×10^{-9}
10.0	111	0.00	5.6×10^{-17}	0.000000	-4.1×10^{-1}
20.0	110	0.00	5.6×10^{-17}	0.000000	-2.2×10^{-6}
100.0	108	0.00	0.0	0.000001	-2.6×10^{-1}

Table 4.16: Experiments with $n = 4$ and $m = 24$ using Dykstra applied to the metric nearness problem with the ℓ_∞-norm.

In most cases, the results with the ℓ_∞ norm required bigger values of γ to achieve the correct solution, once compared to ℓ_1 norm. Regardless of some distinctions, in general, some conclusions can be made regarding γ :

1. The number of iterations, time, and difficulty of convergence increase with the value of γ;

2. When γ is small, although faster, the results are not precise;

3. A large γ can lead to a loss of feasibility.

Based on these conclusions, one might question the reliability of the results obtained by applying Dykstra's method to the linear problem, especially when accuracy is required.

n=8 and m=224					
Simplex Objective Function: 0.000280					
γ	**Iterations**	**Time(s)**	**Infeasibility**	**Linear Function**	**Dual Gap** (ω)
0.1	77	0.00	7.1×10^{-9}	0.928830	1.3
1.0	84	0.00	7.4×10^{-9}	0.748830	6.1×10^{-10}
2.0	263	0.00	9.6×10^{-9}	0.578167	1.7×10^{-9}
10.0	690	0.00	9.7×10^{-9}	0.088111	1.4×10^{-10}
20.0	18354	0.07	6.4×10^{-9}	0.000281	-1.1×10^{-7}
100.0	303165	1.20	8.6×10^{-9}	0.000281	-2.1×10^{-7}

Table 4.17: Experiments with $n = 8$ and $m = 224$ using Dykstra applied to the metric nearness problem with the ℓ_∞-norm.

n=16 and m=1920					
Simplex Objective Function: 0.027837					
γ	**Iterations**	**Time(s)**	**Infeasibility**	**Linear Function**	**Dual Gap** (ω)
1.0	420	0.00	2.3×10^{-9}	1.087888	8.9×10^{-9}
2.0	558	0.00	9.3×10^{-10}	1.049070	6.0×10^{-9}
10.0	714	0.00	3.8×10^{-10}	0.781501	-4.2×10^{-9}
20.0	3255	0.04	2.5×10^{-10}	0.579963	-1.5×10^{-9}
100.0	3156	0.00	2.5×10^{-11}	0.027838	-4.1×10^{-8}
200.0	8274	0.09	1.2×10^{-11}	0.027838	-5.7×10^{-8}

Table 4.18: Experiments with $n = 16$ and $m = 1920$ using Dykstra applied to the metric nearness problem with the ℓ_∞-norm.

4.3 Real Data

In this part of the experiments it was used real-world networks obtained from SuiteSparse Matrix Collection [21] and SNAP repository [23]. The data was extracted from the previous repositories, the distance between each pair of vertices was calculated based on the number of vertices between them in the graph. Then the weights were removed, the edges were made undirected, and the largest connected set was found, so there would be no loose vertices. After that, some noise was generated on the data based on the size of the graph. Finally, the dissimilarity matrix created was converted into a file to be used in the code. All these procedures were made using Julia.

The Jazz repository [18] comes from musicians who have played with each other and associates 198 musicians. The graph can be seen in Figure 4.13. The SmallW [16], [10] has originally 396 vertices, but since some of them are not connected with others, it results in a fully connected graph with 233 vertices, Figure 4.14. The set represents citations in papers, and the values correspond to how close papers are to each other, the name comes from ”Small World”.

The Erdos991 dataset [26] is a representation of co-authorship relationships among mathematicians. It has 446 vertices, and the graph is shown in Figure 4.15. The USAir97 [2] dataset has the

n=32 and m=15872					
Simplex Objective Function: 0.025243					
γ	**Iterations**	**Time(s)**	**Infeasibility**	**Linear Function**	**Dual Gap (ω)**
1.0	1049	0.10	4.1×10^{-9}	1.099882	2.6
2.0	1327	0.13	1.2×10^{-9}	1.063927	2.9
10.0	3585	0.34	1.6×10^{-10}	0.931078	5.7
20.0	2769	0.27	8.4×10^{-11}	0.831191	1.5
100.0	6590	0.57	3.1×10^{-11}	0.369272	-3.1×10^{-8}
200.0	7850	0.68	4.4×10^{-10}	0.086277	-1.5×10^{-7}

Table 4.19: Experiments with $n = 32$ and $m = 15872$ using Dykstra applied to the metric nearness problem with the ℓ_∞-norm.

n=64 and m=129024					
Simplex Objective Function: 0.028121					
γ	**Iterations**	**Time(s)**	**Infeasibility**	**Linear Function**	**Dual Gap (ω)**
10.0	6774	4.45	3.3×10^{-10}	1.091170	1.3
20.0	6922	4.45	1.1×10^{-10}	1.034378	1.1
100.0	36248	22.911	9.3×10^{-12}	0.818612	1.4
200.0	130429	79.83	1.7×10^{-11}	0.650184	-1.7×10^{-9}
1000.0	126982	77.57	6.8×10^{-12}	0.035858	-3.6×10^{-7}
2000.0	1000001	609.31	5.2×10^{-4}	-1.0×10^{-4}	0.028007

Table 4.20: Experiments with $n = 64$ and $m = 129024$ using Dykstra applied to the metric nearness problem with the ℓ_∞-norm.

connection flights between airports in the United States of America in the year of 1997, it contains 332 vertices and the graph can be seen in Figure 4.16.

The SmaGri dataset [15] is based on citations to works by Small and Griffith and their subsequent scholarly descendants. Originally, it had 1059 vertices, but some of them are not connected with the others, so this value is reduced to 1024. The graph is shown in Figure 4.17. The Netscience dataset [25] has each edge representing a coauthorship between two scientists. The initial data has 1589 vertices, but again, this value was reduced to the largest component connected, in this case, 379. The graph can be seen in Figure 4.18.

A bigger problem was also extracted from the matrix collections. The CaGrQc dataset [22] covers scientific collaborations between authors with papers submitted to the General Relativity and Quantum Cosmology category in the period from January 1993 to April 2003. This dataset had 5242 vertices, but the largest component had 4158. The graph is in Figure 4.19.

These first six datasets were applied to the code previously described using ℓ_1 norm in Table 4.23, ℓ_2 norm in Table 4.24 and ℓ_∞ norm in Table 4.25. Although not representing accuracy, the value of γ was set equal to two to help the code achieve convergence even in bigger problems. The relation between the size of the graph and the time and number of iterations for each norm is seen

n=100 and m=495000					
Simplex Objective Function: 0.031334					
γ	**Iterations**	**Time(s)**	**Infeasibility**	**Linear Function**	**Dual Gap (ω)**
10.0	17043	39.10	5.2×10^{-10}	1.123661	-1.4×10^{-2}
20.0	57258	133.66	3.8×10^{-11}	1.091328	1.4
100.0	23138	110.41	1.4×10^{-11}	0.941132	1.1
200.0	21119	88.97	4.1×10^{-12}	0.829263	1.2
1000.0	229987	509.59	1.9×10^{-12}	0.346061	-2.2×10^{-8}
2000.0	512267	1122.96	1.7×10^{-12}	0.059447	1.2×10^{-9}

Table 4.21: Experiments with $n = 100$ and $m = 495000$ using Dykstra applied to the metric nearness problem with the ℓ_∞-norm.

n=200 and m=3980000					
Simplex did not compile.					
γ	**Iterations**	**Time(s)**	**Infeasibility**	**Linear Function**	**Dual Gap (ω)**
10.0	96136	1667.51	1.1×10^{-10}	1.345952	1.1
20.0	140259	2661.16	6.0×10^{-11}	1.271511	1.0
100.0	163741	2777.04	1.7×10^{-11}	1.158664	1.0
200.0	145056	2498.99	1.0×10^{-8}	1.086647	1.1
1000.0	350307	8367.67	1.2×10^{-12}	0.843285	1.1
2000.0	371906	6621.14	2.9×10^{-13}	0.658717	1.1

Table 4.22: Experiments with $n = 200$ and $m = 3980000$ using Dykstra applied to the metric nearness problem with the ℓ_∞-norm.

in Figures 4.20 -4.25.

The last dataset, CaGrQc, which contains over 10^{11} triangle inequality constraints, was only applied to the ℓ_2 norm due to the size of the problem. The result is also shown in Table 4.24.

Graph	**n**	**m**	**Iterations**	**Time(s)**	**Obj function**	**Infeasibility**
Jazz	198	3822978	9314	154.83	4308.475495	1×10^{-8}
SmallW	233	6243934	9593	256.13	6048.079188	1×10^{-8}
Erdos991	446	44061232	2149	404.35	98085.996005	1×10^{-8}
USAir97	332	18132844	1000001	78298.54	2495.955156	4×10^{-6}
SmaGri	1024	535301120	8942	21916.47	90021.067838	1×10^{-8}
NetScience	379	27005645	12128	1402.41	48510.532619	1×10^{-8}

Table 4.23: Experiments using Dykstra applied to real data using ℓ_1-norm and $\gamma = 2$.

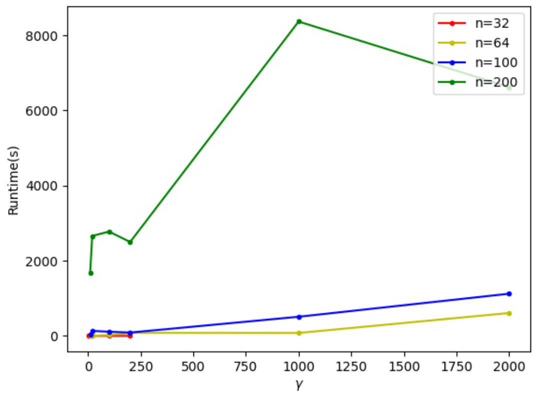

Figure 4.11: Runtime for problems with 32 to 200 entry values using Dykstra to ℓ_∞-norm with various γ values.

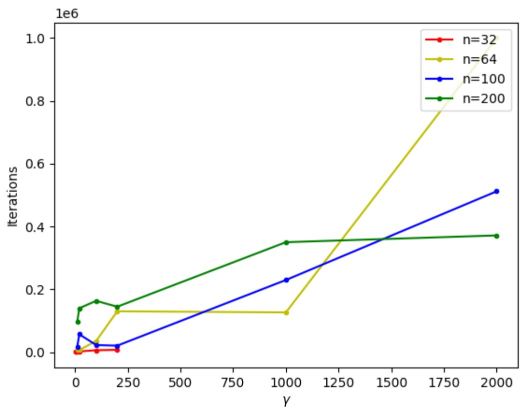

Figure 4.12: Average size of y for problems with 32 to 200 entry values using Dykstra to ℓ_∞-norm with various γ values.

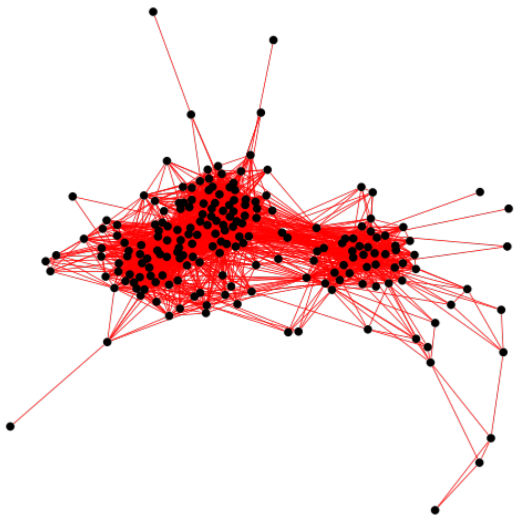

Figure 4.13: Jazz

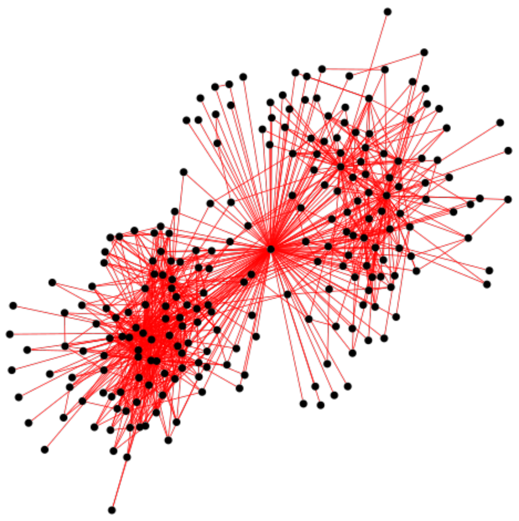

Figure 4.14: SmallW

Graph	**n**	**m**	**Iterations**	**Time(s)**	**Obj function**	**Avg non-zero y(%)**
Jazz	198	3822588	97	4.79	14.213416	0.133
SmallW	233	6243468	66	2.80	11.879831	0.044
Erdos991	446	44060340	211	56.89	27.722387	0.038
USAir97	332	18132180	934	204.41	29.617875	0.227
SmaGri	1024	535299072	257	533.23	47.031099	0.011
NetScience	379	27004887	220	31.51	24.976131	0.060
CaQrGc	4158	35917826868	638	207865.74	195.320973	0.001

Table 4.24: Experiments using Dykstra applied to real data using ℓ_2-norm.

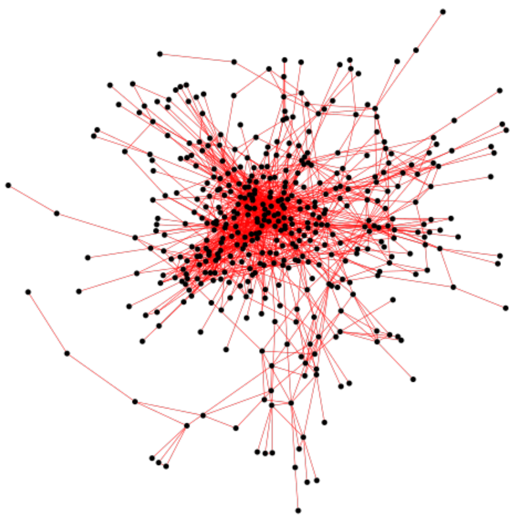
Figure 4.15: Erdos991

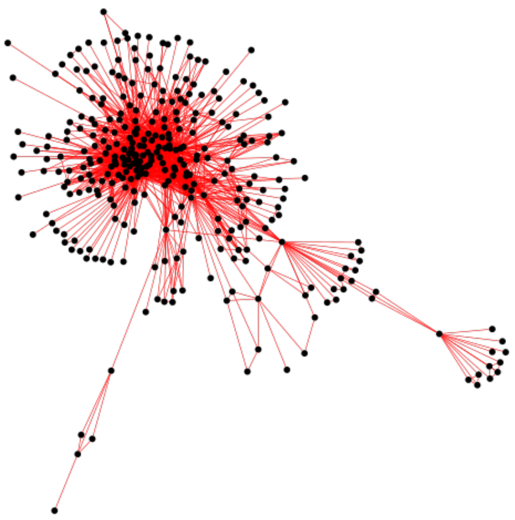
Figure 4.16: USAir97

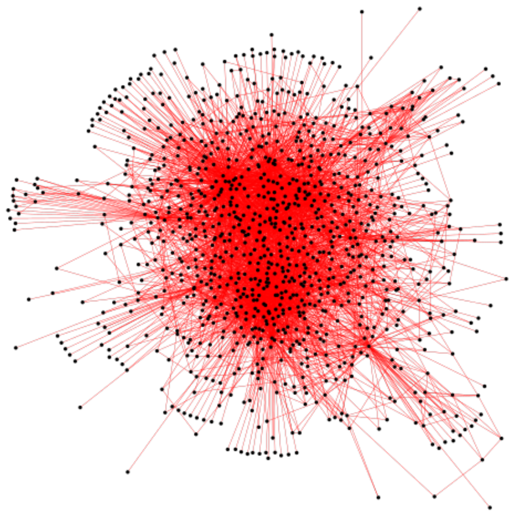
Figure 4.17: SmaGri

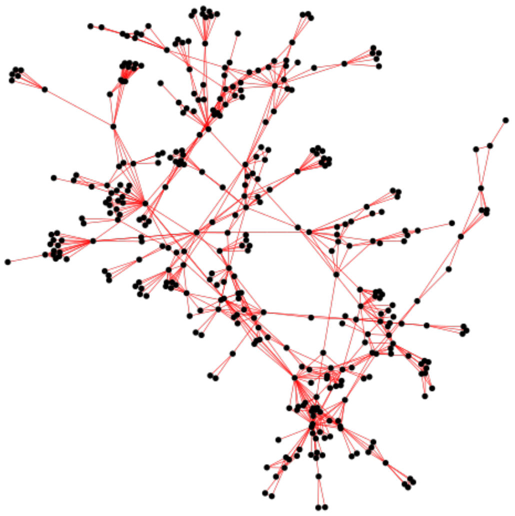
Figure 4.18: Netscience

Graph	**n**	**m**	**Iterations**	**Time(s)**	**Obj function**	**Infeasibility**
Jazz	198	3822978	140762	2250.86	5.954802	1×10^{-8}
SmallW	233	6243934	296544	7946.03	5.246334	1×10^{-8}
Erdos991	446	44061232	1000001	186197.07	11.057955	1.4×10^{-5}
USAir97	332	18132844	599052	45525.61	2.000045	1×10^{-8}
SmaGri	1024	535301120	-	-	-	-
NetScience	379	27005645	996480	111481.47	9.273839	1×10^{-8}

Table 4.25: Experiments using Dykstra applied to real data using ℓ_∞-norm and $\gamma = 2$.

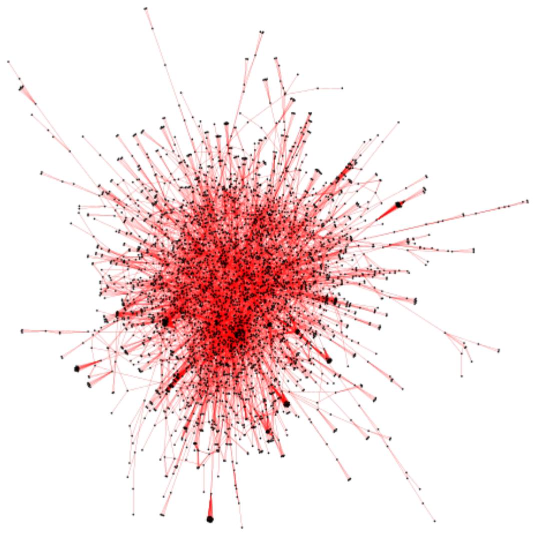

Figure 4.19: CaGrQc

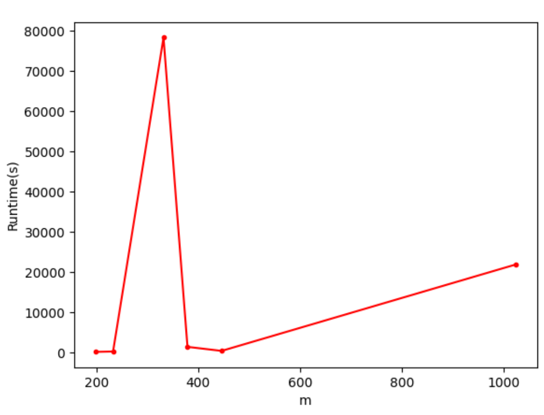

Figure 4.20: Runtime for real data problems using Dykstra to ℓ_1-norm with various $\gamma = 2$.

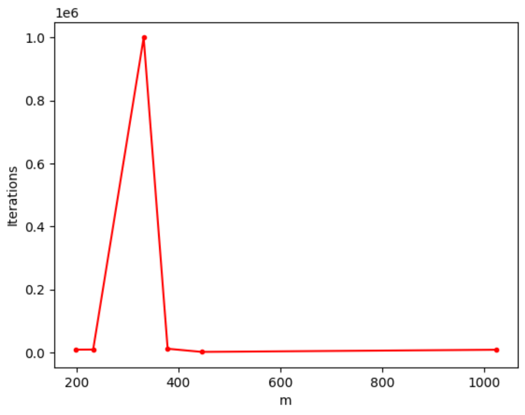

Figure 4.21: Iterations for real data problems using Dykstra to ℓ_1-norm with various $\gamma = 2$.

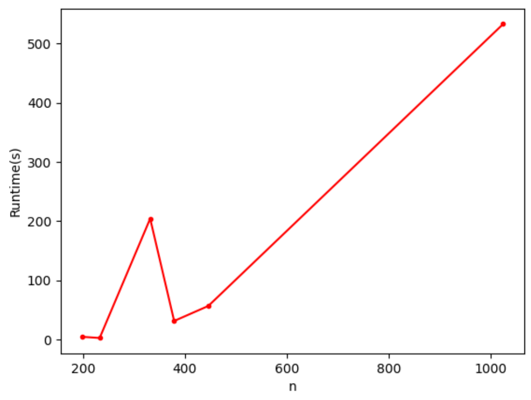

Figure 4.22: Runtime for real data problems using Dykstra to ℓ_2-norm.

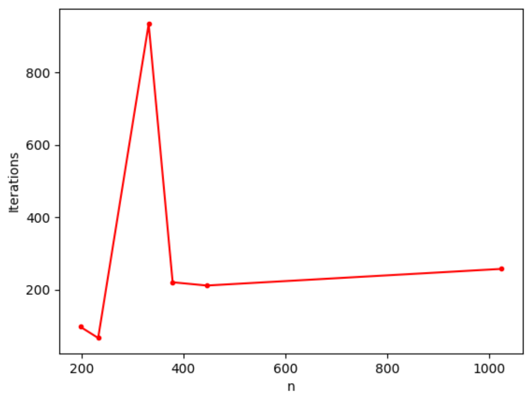

Figure 4.23: Iterations for real data problems using Dykstra to ℓ_2-norm.

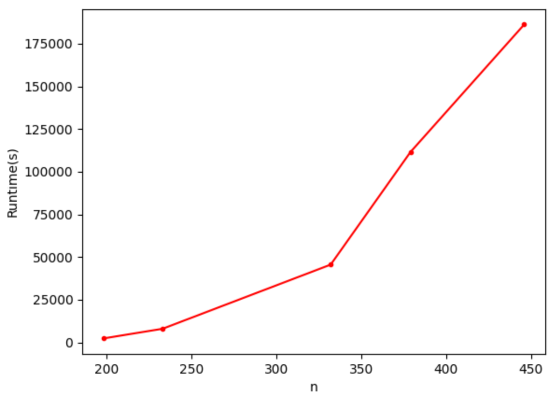

Figure 4.24: Runtime for real data problems using Dykstra to ℓ_∞-norm with various $\gamma = 2$.

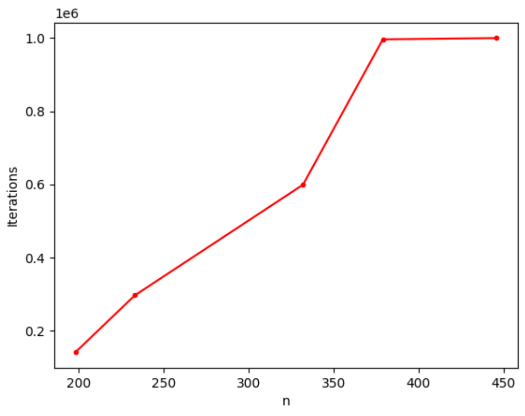

Figure 4.25: Iterations for real data problems using Dykstra to ℓ_∞-norm with various $\gamma = 2$.

Chapter 5

Final considerations

In Chapter 2, a study of the Metric Nearness Problem was developed and it was possible to understand how this problem can be solved using a standard minimization problem considering the three most common norms, ℓ_1, ℓ_2 and ℓ_∞. Then, later in Chapter 3, two methods were studied, Algencan, which can solve linear and non-linear minimization problems, and Dykstra projection method, which finds the projection of a given point in the intersection of closed and convex sets, and for this reason can solve minimization problems where the objective function is a ℓ_2 norm. While in Chapter 4, the focus was on how to use these methods applied to the Metric Nearness Problem and to understand in which conditions Dykstra is applied to the ℓ_1 and ℓ_∞ norms.

The primary objective of the experiments was to explore and analyze the differences between Algencan, Dykstra's algorithm, and Simplex. That is, understand how each method and each norm works, considering important aspects such as computational performance, convergence behavior, and solution accuracy. Algencan showed a great response with the ℓ_2 norm. However, its performance was limited by computational constraints as the number of inequalities increased cubically with the problem size and Algencan uses the Jacobian matrix of the constraint set.

A customized implementation of the Dykstra method was developed specifically for this problem by exploiting the structure and sparsity of the constraints and optimizing memory usage through compact storage of the dual variables and constraint matrix. It was able to handle problems with up to 10^{11} constraints with numerical accuracy for the ℓ_2 norm. This is a result of an implementation adapted to the structure of this problem that allowed it to achieve better results. The solution to apply the ℓ_1 and ℓ_∞ norms was more complicated and needed the use of an unknown value, γ, which introduced a variation in the results depending on the chosen value.

The largest test performed presented a total of 35,917,826,868 constraints, or approximately 10^{11}. It was observed that the average size of the vector y was only $647,522$, a value significantly below the computational limit. This result suggests that, if necessary, even larger problems could be solved. However, the execution time on the available processor begins to pose a problem, since it took $207,865.74$ seconds to solve the problem, which is approximately equivalent to 57 hours.

In [27] and later in [30], the authors propose a Dykstra projection method using parallelism, which allows for the simultaneous execution of multiple projections. With this method, it is possible to solve problems with up to 10^{13} constraints. The main innovation lies in an execution scheme that organizes projections to avoid interference between them, allowing more than one projection to be performed simultaneously. Therefore, the use of parallelism relies primarily on multiple processing cores to execute different parts of the problem, distributing the computational load across multiple processors.

After the initial evaluation, the origin of the problems was expanded and the algorithms developed were applied to known graphs referred to in other works. This also allowed comparing some results with other works. Therefore, it was possible to see that Dykstra's projection method using ℓ_1 and a small value of γ, although efficient in time, showed a lack of accuracy in the objective function results with values that are not optimal.

In conclusion, despite its linear convergence rate, Dykstra constructed for this problem offers an elegant and simple solution. For future work, it would be interesting to work with researchers that apply the method to real problems, like image processing or sensor location, to better understand their requirements in terms of time and accuracy of the results.

Bibliography

[1] R. Andreani, E. G. Birgin, J. M. Martínez, and M. L. Schuverdt. On augmented Lagrangian methods with general lower-level constraints. *SIAM Journal on Optimization*, 18(4):1286–1309, 2008. Cited 2 times on pages 2 and 11.

[2] V. Batagelj. USAir97: US Air Transportation Network. `http://www.cise.ufl.edu/research/sparse/matrices/Pajek/USAir97`, 1997. Undirected weighted graph representing the US air transportation network in 1997. Cited on page 38.

[3] D. Batra, R. Sukthankar, and T. Chen. Semi-Supervised Clustering via Learnt Codeword Distances. In *BMVC*, pages 1–10, 2008. Cited on page 1.

[4] E. G. Birgin. TANGO: A Trustable Algorithm for Nonlinear Optimization. `http://www.ime.usp.br/~egbirgin/tango/`, 2025. Accessed: 2025-07-01. Cited 2 times on pages 14 and 17.

[5] E. G. Birgin and J. M. Martínez. *Practical augmented Lagrangian methods for constrained optimization*. SIAM, 2014. Cited 2 times on pages 11 and 14.

[6] E. G. Birgin and J. M. Martínez. Complexity and performance of an augmented Lagrangian algorithm. *Optimization Methods and Software*, 35(5):885–920, 2020. Cited 2 times on pages 11 and 12.

[7] E. G. Birgin and M. Raydan. Robust stopping criteria for Dykstra's algorithm. *SIAM Journal on Scientific Computing*, 26(4):1405–1414, 2005. Cited on page 16.

[8] J. P. Boyle and R. L. Dykstra. A method for finding projections onto the intersection of convex sets in Hilbert spaces. In *Advances in Order Restricted Statistical Inference: Proceedings of the Symposium on Order Restricted Statistical Inference held in Iowa City, Iowa, September 11–13, 1985*, pages 28–47. Springer, 1986. Cited on page 16.

[9] S. Čapkun, M. Hamdi, and J.-P. Hubaux. GPS-free positioning in mobile ad hoc networks. *Cluster Computing*, 5:157–167, 2002. Cited on page 1.

[10] T. A. Davis and Y. Hu. The University of Florida Sparse Matrix Collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1–25, 2011. Cited on page 38.

[11] M. O. Dayhoff. A model of evolutionary change in proteins. *Atlas of protein sequence and structure*, 5:89–99, 1972. Cited on page 2.

[12] I. S. Dhillon, S. Sra, and J. A. Tropp. The Metric Nearness Problems with applications. *Dept. of Computer Sciences Technical Report TR-03-23, The University of Texas at Austin*, 2003. Cited on page 1.

[13] R. L. Dykstra. An algorithm for restricted least squares regression. *Journal of the American Statistical Association*, 78(384):837–842, 1983. Cited 2 times on pages 2 and 15.

[14] R. Escalante and M. Raydan. *Alternating projection methods.* SIAM, 2011. Cited on page 16.

[15] E. Garfield. SmaGri: Citations to Small and Griffith and Descendants. `https://www.cise.ufl.edu/research/sparse/matrices/Pajek/SmaGri`, 2001. Directed multigraph representing citations to Small and Griffith and their descendants. Cited on page 39.

[16] E. Garfield. SmallW: Small World Citation Network. `https://sparse.tamu.edu/Pajek/SmallW`, 2002. Edited by Vladimir Batagelj. Cited on page 38.

[17] C. Gentile. Distributed sensor location through linear programming with triangle inequality constraints. *IEEE transactions on wireless communications*, 6(7):2572–2581, 2007. Cited 2 times on pages 1 and 7.

[18] P. M. Gleiser and L. Danon. Community structure in jazz. *Advances in complex systems*, 6(04):565–573, 2003. Cited on page 38.

[19] S.-P. Han. A successive projection method. *Mathematical Programming*, 40(1):1–14, 1988. Cited on page 15.

[20] C. Hildreth. A quadratic programming procedure. *Naval research logistics quarterly*, 4(1):79–85, 1957. Cited 2 times on pages 15 and 29.

[21] S. P. Kolodziej, M. Aznaveh, M. Bullock, J. David, T. A. Davis, M. Henderson, Y. Hu, and R. Sandstrom. The suitesparse matrix collection website interface. *Journal of Open Source Software*, 4(35):1244, 2019. Cited on page 38.

[22] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2–es, 2007. Cited on page 39.

[23] J. Leskovec and A. Krevl. Snap datasets: Stanford large network dataset collection. https://snap.stanford.edu/data, 2014. Cited on page 38.

[24] O. L. Mangasarian. Normal solutions of linear programs. In *Mathematical Programming at Oberwolfach II*, pages 206–216. Springer, 2009. Cited on page 32.

[25] M. E. J. Newman. NetScience: Coauthorship Network in Network Science. https://www.cise.ufl.edu/research/sparse/matrices/Newman/netscience, 2006. Undirected weighted graph of coauthorships among scientists working on network theory and experiment. Cited on page 39.

[26] R. Rossi and N. Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015. Cited on page 38.

[27] C. Ruggles, N. Veldt, and D. F. Gleich. A parallel projection method for metric constrained optimization. In *2020 Proceedings of the SIAM Workshop on Combinatorial Scientific Computing*, pages 43–53. SIAM, 2020. Cited 2 times on pages 2 and 46.

[28] V. Singh, L. Mukherjee, P. M. Dinu, J. Xu, and K. R. Hoffmann. Limited view CT reconstruction and segmentation via constrained metric labeling. *Computer Vision and Image Understanding*, 112(1):67–80, 2008. Cited on page 1.

[29] S. Sra, J. Tropp, and I. Dhillon. Triangle fixing algorithms for the metric nearness problem. *Advances in Neural Information Processing Systems*, 17, 2004. Cited 2 times on pages 7 and 17.

[30] P. Tang, B. Jiang, and C. Wang. An efficient algorithm for the l_p norm based metric nearness problem. *Mathematics of Computation*, 2024. Cited 2 times on pages 2 and 46.

[31] N. Veldt, D. Gleich, A. Wirth, and J. Saunderson. A projection method for metric-constrained optimization. *arXiv preprint arXiv:1806.01678*, 2018. Cited 4 times on pages 2, 28, 32, and 34.

[32] S. N. Vitaladevuni and R. Basri. Co-clustering of image segments using convex optimization applied to EM neuronal reconstruction. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2203–2210. IEEE, 2010. Cited on page 2.