Estado de la publicación: No informado por el autor que envía

Aspectos Computacionales del Problema de Fermat-Weber y sus Algoritmos de Solución

Pablo Soto-Quiros

https://doi.org/10.1590/SciELOPreprints.4669

Envíado en: 2022-08-26
Postado en: 2022-09-05 (versión 2)
(AAAA-MM-DD)

Reporte Técnico (Versión: Agosto 2022)
Proyecto Gen-FW (#1440042)

Aspectos Computacionales del Problema de Fermat-Weber y sus Algoritmos de Solución

Computational Aspects of the Fermat-Weber Problem and its Solution Algorithms

Pablo Soto-Quiros[1]*

Resumen
Este artículo científico resume los aspectos computacionales más relevantes de la literatura sobre el problema de Fermat-Weber (FW). El problema FW tiene como objetivo calcular un punto que minimice la suma de las distancias euclidianas ponderadas a p puntos fijos en el plano. Presentamos los pseudocódigos, análisis de convergencia, implementación computacional y simulaciones numéricas de todos los métodos explicados en este artículo científico. Al final, damos un ejemplo de procesamiento de imágenes para determinar la imagen representativa de un conjunto de imágenes. Todas las implementaciones computacionales se desarrollan en MATLAB.

Abstract:
This scientific article summarizes the most relevant computational aspects of the literature on the Fermat-Weber (FW) problem. The FW problem aims to compute a point that minimizes the sum of weighted Euclidean distances to p fixed points in the plane. We present the pseudocodes, convergence analysis, computational implementation, and numerical simulations of all the methods explained in this scientific article. In the end, we give an example of image processing to determine the representative image of a set of images. All computational implementations are developed in MATLAB.

Palabras Claves
Problema de Fermat-Weber – Aspectos Computacionales – MATLAB

Keywords:
Fermat-Weber Problem – Computational Aspects – MATLAB

[1] *Escuela de Matemática, Instituto Tecnológico de Costa Rica, Cartago, Costa Rica*
***Email**: jusoto@tec.ac.cr – **ORCID**: https://orcid.org/0000-0003-2903-3116

Tabla de Contenidos

1. Introducción

El problema de Fermat-Weber (FW) es un problema de optimimización que tiene la finalidad de encontrar un punto (vector) que minimiza la suma de distancias Euclidianas ponderadas a p puntos (vectores) fijos del plano. La formulación matemática de dicho problema es la siguiente: dado los vectores $b_i \in \mathbb{R}^m$ y las constantes $\omega_i > 0$, para $i = 1, ..., p$, el problema FW consiste en encontrar el vector $\widehat{x} \in \mathbb{R}^m$ tal que

$$\widehat{x} = \arg\min_{x \in \mathbb{R}^m} \sum_{i=1}^{p} \omega_i \|x - b_i\|_2, \tag{1}$$

donde $\|\cdot\|_2$ es la norma Euclideana. En la Figura 1 se muestra un ejemplo gráfico para el caso particular de $\omega_j = 1$, $b_j \in \mathbb{R}^2$, para todo $j = 1, ..., p$ con $p = 9$. En este ejemplo, se calcula el vector $\widehat{x} \in \mathbb{R}^2$ que minimice la suma de todas las distancias ponderadas a los vectores $b_1, ..., b_9$.
El problema FW ha sido utilizado en diversas aplicaciones en la vida real. Por ejemplo, en economía [1] en el problema

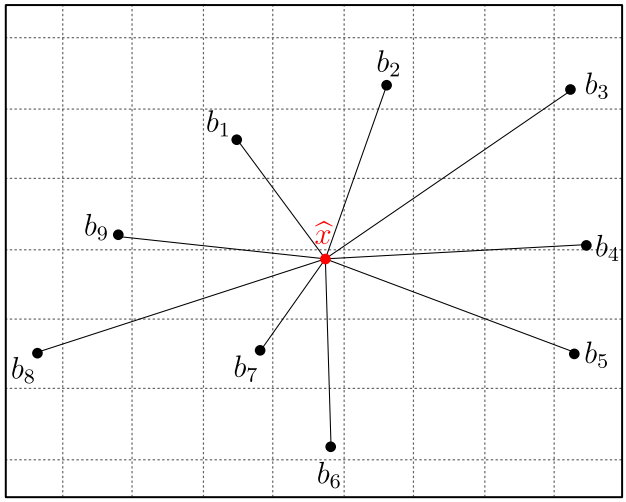


Figura 1. Ejemplo gráfico del problema FW en $\mathbb{R}^2$, cuando $\omega_1 = \omega_2 = ... = \omega_9 = 1$.

de instalar una nueva fábrica de manera que se minimicen los costos de transporte de la materia prima hacia la fábrica y del producto terminado hacia un punto de venta. Otras aplicaciones son en la solución numérica del problema de integración a gran escala con aplicación en la minimización de la longitud lineal del cable para la colocación global [2], el mapeo de superresolución basado en la correlación espacial-espectral para superar la influencia de las condiciones de imagen lineales y no lineales [3], entre otros.

El fin de este artículo científico es realizar un estudio de los métodos computacionales que dan solución al problema FW y sus respectivas variaciones. Nosotros presentamos un resumen completo de los aspectos computacionales más relevantes asociados al problema FW, además de presentar un pseudocódigo de dicha implementación. Adicionalmente, explicaremos la convergencia de cada uno de estos métodos. Por otra parte, en este documento presentaremos otros problemas de optimización que se derivan o generalizan el problema FW, con la finalidad de extender la formulación presentada en (1).

Adicionalmente, mostraremos un conjunto de simulaciones numéricos que ejemplifican la eficiencia y aplicabilidad de los algoritmos implementados. En este documento también se encuentra disponible una explicación de cada una de las funciones implementadas en MATLAB y su respectiva sintáxis. Las implementaciones en MATLAB de los algoritmos y los ejemplos numéricos explicados en este documento están disponbiles para los lectores en GitHub (`https://github.com/jusotoTEC/fermat_weber`).

En la literatura consultada, existen otros artículos científicos que realizan un compendio sobre la teoría y algoritmos relacionados al problema FW, como por ejemplo las investigaciones realizadas en [4, 5, 6, 7, 8, 9]. Sin embargo, en este artículo científico, nosotros nos enfocamos principalmente a la implementación computacional de los algoritmos que dan solución al problema FW y a sus respectivas variantes. Adicionalmente, nosotros facilitamos el código realizado en MATLAB, además de presentar una breve explicación de como usar las funciones implementadas. Lo anterior tiene la finalidad de que otros investigadores puedan utilizar el código implementado en MATLAB, además de modificarlo para utilizarlo en las aplicaciones que consideren oportunas.

2. Aspectos Teóricos del Problema FW

A través de este artículo, utilizaremos la siguiente notación: Denotaremos la función $f : \mathbb{R}^m \to \mathbb{R}^+ \cup \{0\}$, tal que

$$f(x) = \sum_{i=1}^{p} \omega_i \|x - b_i\|_2. \tag{2}$$

El conjunto de vectores $\mathscr{B} = \{b_1, ..., b_p\}$ se conoce como conjunto de nodos y el conjunto de constantes $\mathscr{W} = \{\omega_1, ..., \omega_p\}$ se conoce como conjunto de pesos. Diremos que los elementos de $\mathscr{B}$ son colineales si existen dos vectores $a, v \in \mathbb{R}^m$, con $\|v\|_2 = 1$, y constantes $\beta_i \in \mathbb{R}$ tal que $b_i = \beta_i v + a$, para todo $i = 1, ..., p$.

Basado en las referencias [4, 5, 10], presentamos las propiedades teóricas más relevantes de la función f:

- f es una función continua y convexa.
- f tiene, al menos, un mínimo global.
- El gradiente de f es el operador $\nabla f : \mathbb{R}^m \to \mathbb{R}^m$ tal que

$$\nabla f(x) = \sum_{i=1}^{p} \frac{\omega_i}{\|x - b_i\|_2}(x - b_i), \tag{3}$$

  para todo $x \notin \mathscr{B}$.

- El Hessiano de f es el operador $\nabla f : \mathbb{R}^m \to \mathbb{R}^{m \times m}$ tal que

$$\nabla^2 f(x) = \sum_{i=1}^{p} \frac{\omega_i}{\|x - b_i\|_2^3}(\|x - b_i\|_2^2 I_m - (x - b_i)(x - b_i)^T), \tag{4}$$

  donde $I_m \in \mathbb{R}^{m \times m}$ es la matriz identidad..

- Si los elementos de $\mathscr{B}$ no son colineales, entonces:
  - la función f es estrictamente convexa.
  - f tiene un único minimizador global.
  - Si $\widehat{x}$ es la única solución del problema (1) y $\widehat{x} \notin \mathscr{B}$, entonces $\nabla f(\widehat{x}) = \mathbf{0}_m$, donde $\mathbf{0}_m \in \mathbb{R}^m$ es el vector nulo.

3. Algoritmos de Solución del Problema FW

En esta sección, explicaremos la implementación computacional de tres de los algoritmos iterativos más populares para aproximar una solución del problema FW. Estos algoritmos son el algoritmo de Weiszfeld [11], el algoritmo de Weiszfeld modificado [5] y el algoritmo de Newton-Görner-Kanzow [10]. Para cada uno de estos algoritmos, utilizaremos el vector

inicial $x^{(0)}$ presentado en [10]. Este vector inicial se construye utilizando los siguiente valores:

$$\begin{cases} j = \arg\min_{i=1,\ldots,p} f(b_i), \quad R_j = \sum_{\substack{i=1 \\ i\neq j}}^{p} \frac{\omega_i}{\|b_j - b_i\|_2}(b_j - b_i) \\ d_j = \frac{-R_j}{\|R_j\|_2}, \quad L_j = \sum_{\substack{i=1 \\ i\neq j}}^{p} \frac{\omega_i}{\|b_j - b_i\|_2} \\ \alpha_j = \frac{\|R_j\|_2 - \omega_j}{L_j} \end{cases} \tag{5}$$

Finalmente, el valor inicial se obtiene a través de la fórmula

$$x^{(0)} = b_j + \alpha_j d_j. \tag{6}$$

Antes de explicar los métodos iterativos que dan solución al problema FW, presentamos el siguiente teorema donde se explica un caso partícular donde alguno de los nodos en $\mathscr{B}$ es una solución del problema (5). Este resultado se encuentra en la Proposición 2.2 en [10].

Teorema 1. *Sea $j \in \{1,2,...,p\}$ tal que*

$$f(b_j) = \min_{i=1,\ldots,p} f(b_i).$$

Entonces, b_j es solución del problema FW si y solo si $\|R_j\|_2 \leq \omega_j$.

Este resultado explicado en el Teorema 1 se considerará al inicio de todos los algoritmos presentados en esta sección.

3.1 Método de Weiszfeld

El método de Weiszfeld fue desarrollado por Endreu Vaszonyi Weiszfeld en [11]. Este método genera una sucesión de vectores $\{x^{(k)}\}_{k=0}^{\infty}$ de modo que cada nuevo vector $x^{(k)}$ está más próximo a verificar la identidad $\nabla f(x) = 0_m$. Para esto, se despeja x de dicha identidad, es decir,

$$\begin{aligned} \nabla f(x) = 0_m \quad &\Leftrightarrow \quad \sum_{i=1}^{p} \frac{\omega_i}{\|x - b_i\|_2}(x - b_i) = 0_m \\ &\Leftrightarrow \quad x = \frac{\sum_{i=1}^{p} \frac{\omega_i}{\|x-b_i\|_2} b_i}{\sum_{i=1}^{p} \frac{\omega_i}{\|x-b_i\|_2}}. \end{aligned}$$

Basado en el método iterativo del punto fijo, se define el método iterativo de Weiszfeld de la siguiente manera $x^{(k+1)} = T(x^{(k)})$, para todo $k = 1,2,...$, donde $T : \mathbb{R}^m \to \mathbb{R}^m$ es la función de Weiszfeld definida como

$$T(x) = \frac{\sum_{i=1}^{p} \frac{\omega_i}{\|x-b_i\|_2} b_i}{\sum_{i=1}^{p} \frac{\omega_i}{\|x-b_i\|_2}}.$$

Del Corolario 5.1 en [5], se obtiene que la sucesión $\{x^{(k)}\}_{k=0}^{\infty}$ es acotada. El pseudocódigo del método de Weiszfeld se presenta en el Algoritmo 1.
Por otra parte, note que si $x^{(k)} \in \mathscr{B}$, para algún valor de k, entonces el método de Weiszfeld se indefine. Kuhn presentó en [12] condiciones suficientes para la convergencia del método de Weiszfeld. Dicho resultado se presenta a continuación.

Algoritmo 1: Método de Weiszfeld

Entrada : $b_i \in \mathbb{R}^m$, $\omega_i > 0$, para $i = 1,...,p$, $x^{(0)} \in \mathbb{R}^m$, y $tol > 0$
Salida : $x^{(k)} \in \mathbb{R}^m$, $error \geq 0$

$j = \arg\min_{i=1,\ldots,p} f(b_i)$
$R_j = \sum_{\substack{i=1 \\ i\neq j}}^{p} \frac{\omega_i}{\|b_j - b_i\|_2}(b_j - b_i)$
Si $\|R_j\|_2 \leq \omega_j$ **entonces**
    $x^{(k+1)} = b_j$
    $error = 0$
En otro caso
    $error = tol + 1$
    $k = 0$
    Mientras $error > tol$
        $\alpha_{(k)} = \sum_{i=1}^{p} \frac{\omega_i}{\|x^{(k)} - b_i\|_2}$
        $y^{(k)} = \sum_{i=1}^{p} \frac{\omega_i}{\|x^{(k)} - b_i\|_2} b_i$
        $x^{(k+1)} = \frac{1}{\alpha_{(k)}} y^{(k)}$
        $error = \|x^{(k+1)} - x^{(k)}\|_2$
        $k = k + 1$

Teorema 2 (Convergencia del Método de Weiszfeld). *Sea $\{x^{(k)}\}_{k=0}^{\infty}$ la sucesión generada por el Algoritmo 1. Si $x^{(k)} \notin \mathscr{B}$, para todo $k = 0,1,...$, entonces la sucesión $\{x^{(k)}\}_{k=0}^{\infty}$ converge a la solución del problema (1).*

Del teorema anterior se deduce que el método de Weiszfeld converge solo si $x^{(k)} \notin \mathscr{B}$, para todo $k = 0,1,....$ Además, como el método de Weiszfeld se deduce a partir del gradiente de la función f, entonces otro criterio de parada que se puede considerar para este método es a partir de la desigualdad $\|\nabla f(x^{(k)})\|_2 \leq tol$, donde $tol > 0$ es la tolerancia.
Otro criterio de parada alternativo para el método de Weiszfeld fue desarrollado por Love y Yeong en [13], donde se calcula el vector $x^{(k)}$ que cumple el criterio del error relativo. Para esto, define el siguiente operador:

$$D(x^{(k)}) = f(x^{(k)}) + [\nabla f(x^{(k)})]^T x^{(k)} + \alpha_{min},$$

donde

$$\alpha_{min} = \min_{i=1,\ldots,p} [\nabla f(x^{(k)})]^T b_i.$$

Dicho resultado se presenta en el siguiente teorema.

Teorema 3. *Sea $tol > 0$ y $\{x^{(k)}\}_{k=0}^{\infty}$ la sucesión generada por el método de Weiszfeld. Si se cumple que*

$$\left| \frac{f(x^{(k)}) - D(x^{(k)})}{D(x^{(k)})} \right| \leq tol,$$

Algoritmo 2: Método de Weiszfeld Modificado

Entrada : $b_i \in \mathbb{R}^m$, $\omega_i > 0$, para $i = 1, ..., p$, $x^{(0)} \in \mathbb{R}^m$, y $tol > 0$
Salida : $x^{(k)} \in \mathbb{R}^m$, $error \geq 0$

$j = \arg\min_{i=1,...,p} f(b_i)$
$R_j = \sum_{\substack{i=1 \\ i\neq j}}^{p} \frac{\omega_i}{\|b_j - b_i\|_2}(b_j - b_i)$
si $\|R_j\|_2 \leq \omega_p$ **entonces**
  $x^{(k+1)} = b_j$
  $error = 0$
en otro caso
  $error = tol + 1$
  $k = 0$
  Para $j = 1 : p$
    $R_j = \sum_{\substack{i=1 \\ i\neq j}}^{p} \frac{\omega_i}{\|b_j - b_i\|_2}(b_j - b_i)$
  mientras $error > tol$
    si $x^{(k)} \notin \mathscr{B}$ **entonces**
      $\alpha = \sum_{i=1}^{p} \frac{\omega_i}{\|x^{(k)} - b_i\|_2}$
      $y^{(k)} = \sum_{i=1}^{p} \frac{\omega_i}{\|x^{(k)} - b_i\|_2} b_i$
      $x^{(k+1)} = \frac{1}{\alpha} y^{(k)}$
    si $x^{(k)} = b_j$ y $\|R_j\|_2 > \omega_j$ **entonces**
      $d_j = -R_j / \|R_j\|_2$
      $\alpha_j = (\|R_j\|_2 - \omega_j)/L_j$
      $x^{(k+1)} = b_j + \alpha_j d_j$
    $error = \|x^{(k+1)} - x^{(k)}\|_2$
    $k = k + 1$

entonces $x^{(k)}$ cumple el criterio del error relativo con cota superior tol, es decir,

$$\left| \frac{f(x^{(k)}) - f(\widehat{x})}{f(\widehat{x})} \right| \leq tol,$$

donde $\widehat{x}$ es la solución del problema (1).

Observación 1. Beck y Sabach demostraron en el Corolario 7.1 en [5] que si $x^{(0)} = b_j + \alpha_j d_j$, donde α_j y d_j son definidos en (5), entonces $x^{(k)} \notin \mathscr{B}$, para todo $k \geq 0$. Por lo tanto, la sucesión generada por el método de Weiszfeld converge a la solución del problema (1).

3.2 Método de Weiszfeld Modificado

El método de Weiszfeld tiene el problema que si en la k-ésima interación se obtiene $x^{(k)} = b_i$, para algún $i = 1, ..., p$, y b_i no es solución del problema FW, entonces el método se indefine. Por esta razón se indica en el Teorema 2 que $x^{(k)} \notin \mathscr{B}$. Para resolver este problema, Beck y Sabach proponen en [5] una modificación del método de Weiszfeld. Específicamente, el método considera el vector R_j definido en (5) y analiza tres casos:

- Si $x^{(k)} \neq b_j$, para todo $j = 1, ..., p$, entonces se utiliza la iteración de Weiszfeld para calcular $x^{(k+1)}$.
- Si $x^{(k)} = b_j$ y $\|R_j\|_2 \leq \omega_j$, para algún $j = 1, ..., p$, entonces $x^{(k+1)} = b_j$, ya que por el Teorema 1, se concluye que b_j es una solución del problema FW. Basado en el Teorema 1, esta condición se evalua al inicio del algoritmo para obtener dicha solución con anterioridad, sin tener que realizar el proceso iterativo.
- Si $x^{(k)} = b_j$ y $\|R_j\|_2 > \omega_j$, para algún $j = 1, ..., p$, entonces $x^{(k+1)} = b_j + \alpha_j d_j$, donde α_j y d_j son definidos en (5).

En la práctica, si todas las iteraciones del método de Weiszfeld cumplen que $x^{(k)} \neq b_j$, entonces el método de Weiszfeld modificado es simplemente el método de Weiszfeld explicado en la sección anterior. El pseudocódigo del método de Weiszfeld modificado se presenta en el Algoritmo 2.
El siguiente teorema, desarrollado por Beck y Sabach en [5], presentan la convergencia del método de Weiszfeld modificado.

Teorema 4. *Sea $\{x^{(k)}\}_{k=0}^{\infty}$ la sucesión generada por el Algoritmo 2. Si se cumple que*

$$f(b_j + \alpha_j d_j) < f(b_j),$$

cuando $x^{(k)} = b_j$ y $\|R_j\|_2 > \omega_j$, para todo $j = 1, ..., p$, entonces la sucesión $\{x^{(k)}\}_{k=0}^{\infty}$ converge a una solución óptima del problema (1).

3.3 Método de Newton-Görner-Kanzow

En el 2016, los investigadores Simone Görner y Christian Kanzow proponen en [10] un nuevo método para poder estimar la solución del problema FW. Este método se basa en el clasico método de Newton para resolver problemas de optimización. Además, el método propuesto por Görner y Kanzow considera como valor inicial $x^{(0)} = b_j + \alpha_j d_j$, donde α_j y d_j son definidos en (5). Este método converge cuadráticamente. En este documento, llamaremos a este nuevo método como el método de Newton-Görner-Kanzow. El pseudocódigo del método de Newton-Görner-Kanzow se presenta en el Algoritmo 3.

Observación 2. En principio, el tiempo de ejecución del método de Newton-Görner-Kanzow es mayor que el de los métodos de Weiszfeld y Weiszfeld modificado, ya que los métodos del tipo Newton resuelven un sistema de ecuaciones lineal en cada iteración. Sin embargo, las aplicaciones típicas del problema FW son en $\mathbb{R}^2$ y $\mathbb{R}^3$. Por lo tanto, como se muestra en [10], las soluciones de dichos sistemas de ecuaciones no son muy costosas computacionalmente, y además,

Algoritmo 3: Método de Newton-Görner-Kanzow

Entrada : $b_i \in \mathbb{R}^m$, $\omega_i > 0$, para $i = 1, ..., p$, $x^{(0)} \in \mathbb{R}^m$, y $tol > 0$
Salida : $x^{(k)} \in \mathbb{R}^m$, $error \geq 0$

$error = tol + 1$
$k = 0$
$j = \arg\min_{i=1,...,p} f(b_i)$
$R_j = \sum_{\substack{i=1 \\ i \neq j}}^{p} \frac{\omega_i}{\|b_p - b_i\|_2}(b_p - b_i)$
si $\|R_j\|_2 \leq \omega_p$ **entonces**
  $x^{(k+1)} = b_j$
  $error = 0$
en otro caso
  Escoger $\rho \in]0,1[$
  Escoger $\sigma \in]0, 0.5[$
  mientras $error > tol$
    Calcular $d^{(k)}$ tal que

$$\nabla^2 f(x^{(k)}) d^{(k)} = -\nabla f(x^{(k)})$$

    $bool =$ **verdadero**
    $s = 0$
    mientras $bool$
      $\alpha_k = \rho^s$
      $\mu_1 = f(x^{(k)} + \alpha_k d^{(k)})$
      $\mu_2 = f(x^{(k)}) + \sigma \alpha_k (\nabla f(x^{(k)}))^T d^{(k)}$
      si $\mu_1 \leq \mu_2$ **entonces**
        $bool =$ **falso**
      $s = s + 1$
    $x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$
    $error = \|x^{(k+1)} - x^{(k)}\|_2$
    $k = k + 1$

los resultados numéricos obtenidos en [10] muestran que el método Newton-Görner-Kanzow es significativamente más rápido en términos de tiempos de ejecución que los métodos de Weiszfeld y Weiszfeld modificado. En la Sección (5) realizaremos la prueba computacional de esta afirmación.

El siguiente teorema, desarrollado por Görner y Kanzow en el Teorema 3.1 en [10], presenta la convergencia del método de Newton-Görner-Kanzow.

Teorema 5. *Sea $\{x^{(k)}\}_{k=0}^{\infty}$ la sucesión generada por el Algoritmo 3, donde la condición del Paso 5 nunca se cumple. Entonces $\{x^{(k)}\}_{k=0}^{\infty}$ converge a la única solución óptima del problema* (1).

4. Variaciones del Problema FW

En esta sección presentaremos diversas variaciones del problema FW original que han sido desarrollados en la literatura consultada. Además, explicaremos un método de solución para resolver estos problemas de optimización.

4.1 Problema FW con Norma *s*

El problema FW clásico utiliza la norma 2 de vectores como función de medida. Brimberg y Love propusieron en [14] una extensión del problema FW utilizando la norma *s*. Este problema, llamado el problema FW con norma *s*, se expresa de la siguiente manera: dado los vectores $b_i \in \mathbb{R}^m$ y las constantes $\omega_i > 0$, para $i = 1, ..., p$, el problema FW con norma *s* consiste en encontrar el vector $\widehat{x} \in \mathbb{R}^m$ tal que

$$\widehat{x} = \arg\min_{x \in \mathbb{R}^m} \sum_{i=1}^{p} \omega_i \|x - b_i\|_s, \tag{7}$$

donde

$$\|z\|_s = \left(\sum_{i=1}^{m} |z(i)|^s \right)^{1/s},$$

para $s > 0$ y $z \in \mathbb{R}^m$. Sea $x \in \mathbb{R}^m$ y considere el operador $T_j : \mathbb{R}^n \to \mathbb{R}$ definido por

$$T_j(x) = \begin{cases} \frac{\sum_{i=1}^{p} U_{ij}(x) b_i(j)}{\sum_{i=1}^{p} U_{ij}(x)}, & \text{si } x(j) \neq b_i(j),\ \forall i \in \{1, ..., p\} \\ b_i(j), & \text{si } \exists i \in \{1, ..., p\} : x(j) = b_i(j) \end{cases},$$

para $k = 0, 1, 2, ...$, donde

$$U_{ij}(x) = \frac{\omega_i |x(j) - b_i(j)|^{s-2}}{\|x - b_i\|_s^{s-1}}.$$

Brimberg y Love propusieron en [14] el siguiente método iterativo aproximar la solución del problema (7):

$$x^{(k+1)}(j) = T_j(x^{(k)})$$

para todo $j = 1, 2, ..., m$, es decir,

$$x^{(k+1)} = \begin{pmatrix} T_1(x^{(k)}) \\ T_2(x^{(k)}) \\ \vdots \\ T_m(x^{(k)}) \end{pmatrix}.$$

Este método iterativo solo se define para $s \in [1, 2]$. El pseudocódigo del método de este método itertaivo se presenta en el Algoritmo 4.
El siguiente teorema, desarrollado por Brimberg y Love en el Lemma 1 y Teorema 1 en [14], presenta la convergencia del Algoritmo 4.

Teorema 6. *Sea $\{x^{(k)}\}_{k=0}^{\infty}$ una sucesión generada por el Algoritmo 4 y sea $s \in [1, 2]$. Entonces, $\{x^{(k)}\}_{k=0}^{\infty}$ converge a una solución óptima del problema* (7)*, para cualquier valor inicial $x^{(0)} \in \mathbb{R}^m$.*

Algoritmo 4: Método de Solución del Problem FW con Norma s

Entrada : $b_i \in \mathbb{R}^m$, $\omega_i > 0$, para $i = 1,...,p$, $x^{(0)} \in \mathbb{R}^m$, $s \in [1,2]$ y $tol > 0$
Salida : $x^{(k)} \in \mathbb{R}^m$, $error \geq 0$

$error = tol + 1$
$k = 0$
mientras $error > tol$
  Para $j = 1 : m$
    si $x^{(k)}(j) \neq b_i(j)$, $\forall i \in \{1,...,p\}$ **entonces**
      $num = 0$
      $den = 0$
      Para $i = 1 : p$
        $U_{ij} = \omega_i |x^{(k)}(j) - b_i(j)|^{s-2} / \|x^{(k)} - b_i\|_s^{s-1}$
        $num = num + U_{ij} b_i(j)$
        $den = den + U_{ij}$
      $x^{(k+1)}(j) = num/dem$
    en otro caso
      $x^{(k+1)}(j) = b_i(j)$
    $error = \|x^{(k+1)} - x^{(k)}\|_s$
    $k = k + 1$

4.2 Problema FW con Matrices de Peso

El problema (1) puede re-escribirse de otra forma si consideramos la siguiente igualdad:

$$\omega_i \|x - b_i\|_2 = \|\omega_i^2 I_m x - \omega_i^2 b_i\|_2 = \|\widetilde{A}_i x - \widetilde{b}_i\|_2,$$

donde $\widetilde{A}_i = \omega_i^2 I_m$ y $\widetilde{b}_i = \omega_i^2 b_i$, para todo $i = 1,...,p$. Por lo tanto, el problema FW puede representarse como

$$\widehat{x} = \arg\min_{x \in \mathbb{R}^m} \sum_{i=1}^{p} \|\widetilde{A}_i x - \widetilde{b}_i\|_2,$$

Basado en lo anterior, Overton [15] considera una generalización del problema (1) llamado el problema FW con matrices de peso. Este problema considera lo siguiente: dado las matrices $A_i \in \mathbb{R}^{m \times n}$ y los vectores $b_i \in \mathbb{R}^m$, para $i = 1,...,p$, el problema FW con matrices de peso calcula el vector $\widehat{x} \in \mathbb{R}^n$ tal que

$$\widehat{x} = \arg\min_{x \in \mathbb{R}^n} \sum_{i=1}^{p} \|A_i x - b_i\|_2. \tag{8}$$

Varios métodos se han desarrollado para aproximar la solución del problema (8). Por ejemplo, un algoritmo predictor-corrector desarrollado por Andersen *et al.* en [16] se deriva de un algoritmo de punto interior *primal-dual* aplicando el método de Newton directamente a un sistema de ecuaciones no lineales que representan la viabilidad primaria-dual y una condición complementaria perturbada. Por otro lado, el algoritmo *primal-dual* desarrollado por Qi *et al.* en [17] presenta un método de suavizado aumentado para resolver una ecuación no diferenciable. Al transformar el problema (8) en un problema clásico de programación convexa en forma cónica, Xue y Ye muestran en [18] que una solución óptima del problema (8) se puede estimar de manera eficiente utilizando algoritmos de punto interior. Otros métodos adicionales fueron desarrollados en [15, 19, 20, 21, 22]

Por su fácil implementación y eficiencia computacional, en este documento consideramos el método desarrollado por Andersen *et al.* en [16] para aproximar la solución del problema (8). El método propuesto en [16] se conoce como método predictor-corrector. Para explicar los pasos del método predictor-corrector, consideraremos la siguiente notación: Definamos $A = [A_1^T\ A_2^T\ ...\ A_p^T] \in \mathbb{R}^{n \times pm}$ y $b = [b_1^T\ b_2^T\ ...\ b_p^T]^T \in \mathbb{R}^{mp}$. Sea $y = [y_1^T\ y_2^T\ ...\ y_p^T]^T \in \mathbb{R}^{mp}$ un vector generado aleatoriamente tal que $y_j \in \mathbb{R}^m$, para todo $j = 1,...,p$. Los pasos del método predictor-corrector son los siguientes:

- **Paso 1:** Escoger aleatorimante un número positivo μ cercano a cero y una tolerancia de parada $tol > 0$.
- **Paso 2:** Definir $z_j = A_j x - b_j$ y $\omega_j^\mu = \sqrt{\|z_j\|_2^2 + \mu^2}$, para $j = 1,...,p$. Además, definir la matriz diagonal por bloques $E_\mu, F_\mu \in \mathbb{R}^{mp \times mp}$ tal que

$$E_\mu = \text{blkdiag}(\omega_1^\mu I_m,\ ...\ ,\ \omega_p^\mu I_m) \tag{9}$$

y

$$F_\mu = I_{mp} - \text{blkdiag}\left(\frac{1}{\omega_1^\mu} y_1 z_1^T\ ,...\ ,\frac{1}{\omega_p^\mu} y_p z_p^T\right). \tag{10}$$

- **Paso 3:** Definir $H_\mu = \frac{1}{2} E_\mu^{-1}(F_\mu + F_\mu^T) \in \mathbb{R}^{mp \times mp}$. Si $AH_\mu A^T$ es una matriz singular, entonces redefinir H_μ por una matriz *cerca* positiva definida, es decir,

$$H_\mu = \frac{1}{2}(B + R) + \varepsilon I_{mp}, \tag{11}$$

donde ε es una constante positiva cercana a cero y $B = UR$ es la descomposición polar de $B = \frac{1}{2}(H_\mu + H_\mu^T)$ (ver [23] para más detalles de la descomposición polar d euna matriz).

- **Paso 4:** Determinar los pasos predictores Δx, Δz y Δy tal que

$$AH_\mu A^T \Delta x = AE_\mu^{-1} z, \tag{12}$$
$$\Delta z = -A^T \Delta x \tag{13}$$
$$\Delta y = E_\mu^{-1}(F_\mu \Delta z + r_c) \tag{14}$$

donde $r_c = z - E_\mu y$.

- **Paso 5:** Definir los vectores $\widetilde{x}$, $\widetilde{y}$, $\widetilde{z}$ tal que

$$\widetilde{x} = x + \beta \Delta x \tag{15}$$
$$\widetilde{y} = \gamma(y + \Delta y) \tag{16}$$
$$\widetilde{z} = z + \beta \Delta z \tag{17}$$

donde

$$\beta = \arg\min_{\widetilde{\beta}\in[0,1]} \sum_{j=1}^{p} \sqrt{\|z_j + \widetilde{\beta}\Delta z_j\|_2^2 + \mu^2}, \tag{18}$$

$$\gamma = \max\{\widetilde{\gamma} : \widetilde{\gamma}\|y_j + \Delta y_j\|_2 \leq 1, \forall j = 1, ..., p\}. \tag{19}$$

- **Paso 6:** Definir $\widetilde{\mu}$ y $\omega_j^{\widetilde{\mu}}$ tal que

$$\widetilde{\mu} = \frac{[\text{gap}(y+\Delta y, z+\Delta z)]^3}{p[\text{gap}(y,z)]^2}, \tag{20}$$

$$\omega_j^{\widetilde{\mu}} = \sqrt{\|z_j\|^2 + \widetilde{\mu}^2},$$

donde

$$\text{gap}(y,z) = \sum_{j=1}^{p} (\|z_j\|_2 - y_j^T z_j). \tag{21}$$

- **Paso 7:** Determinar los pasos correctores Δx, Δz y Δy tal que

$$AH_\mu A^T \Delta x = A(E_\mu^{-1} r_c^c + y) \tag{22}$$

$$\Delta z = -A^T \Delta x \tag{23}$$

$$\Delta y = E_\mu^{-1}(F_\mu \Delta z + r_c^c) \tag{24}$$

donde

$$r_c^c = [(r_c^c)_1 \ (r_c^c)_2 \ ... \ (r_c^c)_p]^T \tag{25}$$

tal que $(r_c^c)_j = p_j + q_j + t_j$,

$$p_j = z_j - \omega_j^{\widetilde{\mu}} y_j - \frac{z_j^T \Delta z_j}{\omega_j^{\widetilde{\mu}}} \Delta y_j - \frac{\|\Delta z_j\|_2^2}{2\omega_j^{\widetilde{\mu}}} y_j + \frac{(z_j^T \Delta z_j)^2}{2(\omega_j^{\widetilde{\mu}})^3} y_j$$

$$q_j = (\omega_j^{\mu} - \omega_j^{\widetilde{\mu}})\Delta y_j + \left(\frac{1}{\omega_j^{\mu}} - \frac{1}{\omega_j^{\widetilde{\mu}}}\right)(z_j^T \Delta z_j) y_j$$

$$t_j = \frac{1}{2\omega_j^{\mu}}((\Delta z_j^T y_j) z_j - (\Delta z_j^T z_j) y_j).$$

- **Paso 8:** Si $(E_{\widetilde{\mu}}^{-1} z)^T \Delta z < 0$, entonces $\mu = \widetilde{\mu}$ y recalcular $\widetilde{x}$, $\widetilde{y}$ y $\widetilde{z}$ utilizando ecuaciones (15) – (17), respectivamente.

- **Paso 9:** Remplazar x, y, z y μ por $\widetilde{x}$, $\widetilde{y}$, $\widetilde{z}$ y $\widetilde{\mu}$, respectivamente.

- **Paso 10:** Si $\|x^{(k+1)} - x^{(k)}\| < tol$ entonces salir. En caso contrario, ir a Paso 2.

El pseudocódigo del método de predictor-corrector se presenta en el Algoritmo 5.
El Algoritmo 5 tiene propiedades notables, como se indica en [16]. Por ejemplo, este método presenta una convergencia robusta a la solución óptima del problema (8) y una convergencia local rápida, basado en los experimentos numéricos presentados en [16], por lo que se requiere una cantidad pequeña de iteraciones a pesar de la demanda de alta precisión. Sin embargo, no existe un resultado matemático que demuestre la convergencia teórica del Algoritmo 5.

Algoritmo 5: Método Predictor-Corrector

Entrada : $x^{(0)} \in \mathbb{R}^n$, $Y^{(0)} = [y_1^{(0)} \ ... \ y_p^{(0)}] \in \mathbb{R}^{m\times p}$, $B = [b_1 \ ... \ b_p] \in \mathbb{R}^{m\times p}$, $A = [A_1^T \ A_2^T \ ... \ A_p^T] \in \mathbb{R}^{n\times pm}$, $\mu^{(0)} > 0$, $tol > 0$
Salida : $x^{(k)} \in \mathbb{R}^n$, $error \geq 0$

1 $k = 0$
2 $error = tol + 1$
3 **mientras** $error > tol$
4 **Para** $j = 1 : p$
5 $z_j^{(k)} = b_j - A_j x^{(k)}$
6 $\omega_j^{\mu^{(k)}} = \sqrt{\|z_j^{(k)}\|_2^2 + (\mu^{(k)})^2}$
7 Calcular $E_{\mu^{(k)}}$ y $F_{\mu^{(k)}}$ usando (9) and (10)
8 $H_{\mu^{(k)}} = \frac{1}{2} E_{\mu^{(k)}}^{-1}(F_{\mu^{(k)}} + F_{\mu^{(k)}}^T)$
9 $A = [A_1^T \ A_2^T \ ... \ A_p^T]$
10 **si** $AH_{\mu^{(k)}}A^T$ es singular **entonces**
11 Actualizar $H_{\mu^{(k)}}$ usando (11)
12 $y^{(k)} = \begin{pmatrix} y_1^{(k)} \\ \vdots \\ y_p^{(k)} \end{pmatrix}$, $z^{(k)} = \begin{pmatrix} z_1^{(k)} \\ \vdots \\ z_p^{(k)} \end{pmatrix}$, $b^{(k)} = \begin{pmatrix} b_1^{(k)} \\ \vdots \\ b_p^{(k)} \end{pmatrix}$
13 Calcular $\Delta x^{(k)}$, $\Delta z^{(k)}$ y $\Delta y^{(k)}$ usando (12), (13) y (14)
14 Calcular $\widetilde{x}^{(k)}$, $\widetilde{y}^{(k)}$ y $\widetilde{z}^{(k)}$ usando (15), (16) y (17)
15 Calcular $\widetilde{\mu}^{(k)}$ usando (20)
16 **Para** $j = 1 : p$
17 $\omega_j^{\widetilde{\mu}^{(k)}} = \sqrt{\|z_j^{(k)}\|_2^2 + (\widetilde{\mu}^{(k)})^2}$
18 Calcular $\Delta x^{(k)}$, $\Delta z^{(k)}$ y $\Delta y^{(k)}$ usando (22), (23) y (24)
19 Calcular $E_{\widetilde{\mu}^{(k)}}$ usando (9)
20 **si** $(E_{\widetilde{\mu}^{(k)}}^{-1} z^{(k)})^T \Delta z^{(k)} < 0$ **entonces**
21 $\mu^{(k)} = \widetilde{\mu}^{(k)}$
22 Calcular $\widetilde{x}^{(k)}$, $\widetilde{y}^{(k)}$ y $\widetilde{z}^{(k)}$ usando (15), (16) y (17)
23 $x^{(k+1)} = \widetilde{x}^{(k)}$, $y^{(k+1)} = \widetilde{y}^{(k)}$, $z^{(k+1)} = \widetilde{z}^{(k)}$, $\mu^{(k)} = \widetilde{\mu}^{(k)}$
24 $error = \|x^{(k+1)} - x^{(k)}\|_2$
25 $k = k + 1$

Observación 3. En [16], los autores definen otro criterio de parada que involucra la funcion $\text{gap}(y,z)$ presentada en (21). En este artículo, consideramos la fórmula de error absoluto que se encuentra en el Paso 24 del Algoritmo 5, con el fin de utilizar el mismo criterio de parada en todos los algoritmos presentados en este documento.

5. Simulaciones Numéricas

En esta sección presentaremos algunas simulaciones numéricas que mostrarán la eficiencia y precisión de los algoritmos explicados en las secciones anteriores. Estas simulaciones se implementaron en MATLAB R2019a utilizando una

Método	Tamaño (p)	Tiempo ($seg.$)	Iteraciones (k)	Error Absoluto $\|x^{(k+1)} - x^{(k)}\|_2$	Error Gradiente $\|\nabla f(x^{(k+1)})\|_2$
Weiszfeld	10	4.401×10^{-3}	61	1.928×10^{-16}	3.388×10^{-16}
	100	1.964×10^{-3}	35	1.602×10^{-16}	6.466×10^{-15}
	500	2.493×10^{-2}	36	1.640×10^{-16}	1.869×10^{-14}
	750	4.864×10^{-2}	34	1.258×10^{-16}	8.372×10^{-14}
	1000	8.291×10^{-2}	33	9.087×10^{-17}	2.410×10^{-14}
	2000	3.175×10^{-1}	32	8.452×10^{-17}	4.957×10^{-14}
	5000	1.941×10^{0}	31	1.798×10^{-16}	1.531×10^{-13}
	10000	8.079×10^{0}	31	1.671×10^{-16}	2.768×10^{-13}
	15000	1.745×10^{1}	31	7.125×10^{-17}	4.839×10^{-13}
	20000	3.210×10^{1}	30	1.654×10^{-16}	1.161×10^{-12}
	30000	7.073×10^{1}	30	1.804×10^{-16}	4.786×10^{-13}
Weiszfeld Modificado	10	4.481×10^{-3}	61	1.928×10^{-16}	3.388×10^{-16}
	100	2.213×10^{-3}	35	1.602×10^{-16}	6.466×10^{-15}
	500	2.641×10^{-2}	36	1.640×10^{-16}	1.869×10^{-14}
	750	5.033×10^{-2}	34	1.258×10^{-16}	8.372×10^{-14}
	1000	8.383×10^{-2}	33	9.087×10^{-17}	2.410×10^{-14}
	2000	3.268×10^{-1}	32	8.452×10^{-17}	4.957×10^{-14}
	5000	1.988×10^{0}	31	1.798×10^{-16}	1.531×10^{-13}
	10000	8.216×10^{0}	31	1.671×10^{-16}	2.768×10^{-13}
	15000	1.752×10^{1}	31	7.125×10^{-17}	4.839×10^{-13}
	20000	3.162×10^{1}	30	1.654×10^{-16}	1.161×10^{-12}
	30000	7.021×10^{1}	30	1.804×10^{-16}	4.786×10^{-13}
Newton-Görner-Kanzow	10	6.475×10^{-3}	6	1.387×10^{-17}	6.620×10^{-17}
	100	3.516×10^{-3}	9	2.861×10^{-17}	3.797×10^{-13}
	500	3.485×10^{-2}	10	3.198×10^{-17}	4.505×10^{-6}
	750	4.763×10^{-2}	4	7.244×10^{-17}	3.650×10^{-15}
	1000	8.088×10^{-2}	4	6.206×10^{-17}	1.482×10^{-14}
	2000	3.153×10^{-1}	4	5.123×10^{-17}	6.983×10^{-15}
	5000	2.101×10^{0}	11	1.817×10^{-16}	3.514×10^{-8}
	10000	8.068×10^{0}	4	7.949×10^{-18}	2.679×10^{-14}
	15000	1.759×10^{1}	4	4.726×10^{-17}	6.819×10^{-14}
	20000	3.121×10^{1}	5	3.839×10^{-17}	8.686×10^{-9}
	30000	7.209×10^{1}	6	9.455×10^{-17}	8.955×10^{-10}

Tabla 1. Resultados de la Simulación Numérica 1 asociados al tiempo computacional, iteraciones, error absoluto y error del gradiente de los métodos de Weiszfeld, Weiszfeld Modificado y Newton-Görner-Kanzow para aproximar la solución del problema FW cuando $m = 2$.

computadora de escritorio con un procesador de 2.80 GHz (Intel Core i9-10900F) y 32.00 GB de RAM. El código de MATLAB de todos los experimentos numéricos están disponible en GitHub (`https://github.com/jusotoTEC/fermat_weber`)

• Simulación Numérica 1: Algoritmos de Weiszfeld, Weiszfeld Modificado y Newton-Görner-Kanzow

En este ejemplo compararemos los métodos de Weiszfeld, Weiszfeld Modificado y Newton-Görner-Kanzow que se encuentran en los Algoritmos 1, 2 y 3, respectivamente. Estos métodos aproximan la solución al problema FW en (5).
En este caso, consideraremos vectores $b_i \in \mathbb{R}^m$ y pesos $\omega_i > 0$, para $i = 1, ..., p$, donde cada b_i y ω_i son generados de manera aleatoria utilizando una distribución uniforme en $[0, 1]$. Además, se utilizó una tolerancia $tol = 2^{-52}$.

Las Tablas 1 y 2 presentan el tiempo de ejecución, iteraciones y errores asociados con los Algoritmos 1, 2 y 3, donde $p = 10, 100, 500, 750, 1000, 2000, 5000, 10000, 15000, 20000$ y 30000. La Tabla 1 considera el caso $m = 2$ y la Tabla 2 considera el caso $m = 20$.

Los resultados obtenidos en las Tablas 1 y 2 muestran que el Algoritmo de Newton-Görner-Kanzow necesita menos iteraciones para aproximar la solución del problema FW. Sin embargo, el tiempo de ejecución de los tres métodos es similar.

Por otra parte, los resultados obtenidos en el cálculo de los errores absolutos y de gradiente de los tres algoritmos permiten demostrar numéricamente la precisión de dicho algoritmos.

Método	Tamaño (p)	Tiempo ($seg.$)	Iteraciones (k)	Error Absoluto $\|x^{(k+1)}-x^{(k)}\|_2$	Error Gradiente $\|\nabla f(x^{(k+1)})\|_2$
Weiszfeld	10	4.081×10^{-3}	28	2.775×10^{-17}	4.658×10^{-16}
	100	1.385×10^{-3}	16	1.279×10^{-16}	3.282×10^{-15}
	500	2.516×10^{-2}	14	8.577×10^{-17}	8.769×10^{-15}
	750	5.450×10^{-2}	14	4.997×10^{-17}	2.947×10^{-14}
	1000	9.618×10^{-2}	13	1.809×10^{-16}	3.715×10^{-14}
	2000	3.662×10^{-1}	13	2.101×10^{-16}	3.174×10^{-14}
	5000	2.297×10^{0}	13	7.204×10^{-17}	1.082×10^{-13}
	10000	9.308×10^{0}	13	4.376×10^{-17}	2.349×10^{-13}
	15000	2.098×10^{1}	13	3.821×10^{-17}	3.428×10^{-13}
	20000	3.730×10^{1}	13	3.451×10^{-17}	3.733×10^{-13}
	30000	8.444×10^{1}	13	4.045×10^{-17}	9.338×10^{-13}
Weiszfeld Modificado	10	3.714×10^{-3}	28	2.775×10^{-17}	4.658×10^{-16}
	100	1.657×10^{-3}	16	1.279×10^{-16}	3.282×10^{-15}
	500	2.596×10^{-2}	14	8.577×10^{-17}	8.769×10^{-15}
	750	5.628×10^{-2}	14	4.997×10^{-17}	2.947×10^{-14}
	1000	9.931×10^{-2}	13	1.809×10^{-16}	3.715×10^{-14}
	2000	3.745×10^{-1}	13	2.101×10^{-16}	3.174×10^{-14}
	5000	2.349×10^{0}	13	7.204×10^{-17}	1.082×10^{-13}
	10000	9.254×10^{0}	13	4.376×10^{-17}	2.349×10^{-13}
	15000	2.105×10^{1}	13	3.821×10^{-17}	3.428×10^{-13}
	20000	3.745×10^{1}	13	3.451×10^{-17}	3.733×10^{-13}
	30000	8.506×10^{1}	13	4.045×10^{-17}	9.338×10^{-13}
Newton-Görner-Kanzow	10	4.482×10^{-3}	4	1.272×10^{-16}	1.207×10^{-15}
	100	1.867×10^{-3}	4	2.047×10^{-16}	1.726×10^{-15}
	500	2.745×10^{-2}	4	1.386×10^{-16}	4.285×10^{-15}
	750	5.742×10^{-2}	4	3.794×10^{-17}	1.088×10^{-14}
	1000	9.890×10^{-2}	4	2.105×10^{-16}	1.314×10^{-14}
	2000	7.723×10^{-1}	9	1.966×10^{-17}	2.006×10^{-10}
	5000	2.370×10^{0}	4	1.509×10^{-16}	3.355×10^{-14}
	10000	9.519×10^{0}	7	1.725×10^{-16}	3.068×10^{-10}
	15000	2.103×10^{1}	4	1.816×10^{-16}	1.513×10^{-13}
	20000	3.818×10^{1}	4	1.275×10^{-16}	1.583×10^{-13}
	30000	8.399×10^{1}	5	6.580×10^{-17}	7.394×10^{-11}

Tabla 2. Resultados de la Simulación Numérica 1 asociados al tiempo computacional, iteraciones, error absoluto y error del gradiente de los métodos de Weiszfeld, Weiszfeld Modificado y Newton-Görner-Kanzow para aproximar la solución del problema FW cuando $m = 20$.

• Simulación Numérica 2: Algoritmo de Solución del Problema FW con Norma s

En este experimento analizaremos la ejecución y eficiencia del Algoritmo 4 para aproximar la solución del problema FW con norma s, donde $s \in [1,2]$.

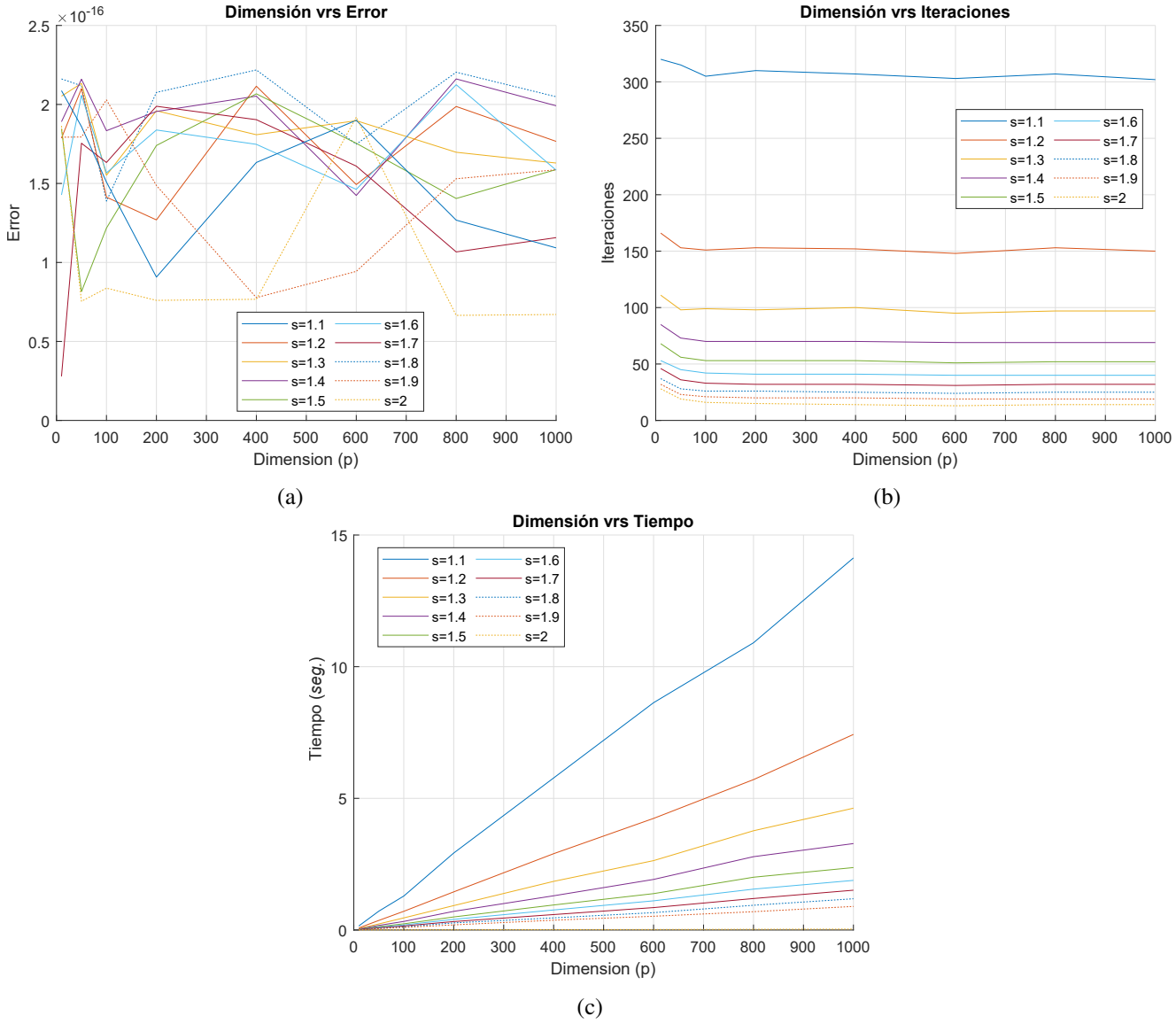
En este caso, consideraremos vectores $b_i \in \mathbb{R}^m$ y pesos $\omega_i > 0$, para $i = 1,...,p$, donde cada b_i y ω_i son generados de manera aleatoria utilizando una distribución uniformeen $[0,1]$. Además, se utilizó una tolerancia $tol = 2^{-52}$. La Figura 2 presenta el error, tiempo de ejecución e iteraciones asociadas con el Algoritmo 4, donde $p = 10,50,100,200,400,600,800$ y 1000 y $s = 1.1,1.2,1.3,1.4,1.5,1.6,1.7,1.8,1.9$ y 2.

La Figura 2(a) muestra que el Algoritmo 4 aproxima con una buena precisión la solución del problema FW con norma s para diferentes valores de s. Por otra parte, podemos observar en la Figura 2(b) que la cantidad de iteraciones necesarias para aproximar la solución del problema (7) aumenta mientras el valor de s, asociado con la norma, aumenta también. Por consiguiente, y como se muestra en la Figura 2(c), el tiempo de ejecución disminuye mientras el valor de s disminuye también.

• Simulación Numérica 3: Algoritmo Predictor-Corrector

En esta simulación, replicaremos el ejemplo numérico propuesto por Overton en [15] para resolver el problema (8) utilizando el Algoritmo 5.

En este ejemplo consideramos $A_j \in \mathbb{R}^{2\times2}$ y $b_j \in \mathbb{R}^2$, para $j = 1,2,3$, tal que $b_1 = (-1,0)^T$, $b_2 = (0,\sigma)^T$, $b_3 = (1,0)^T$, $A_1 = A_3 = I_2$ y $A_2 = \sigma I_2$, donde $\sigma > 0$. La solución exacta de este problema es $x^* = (0,1)^T$.



(a) (b)

(c)

Figura 2. Gráficas de los resultados de la Simulación Numérica 2 asociados al error absoluto, iteraciones y tiempo computacional del Algoritmo 4 para aproximar la solución del problema FW con norma s.

En el Algoritmo 5, utilizaremos una tolerancia $tol = 10^{-12}$ y $\sigma = 2$. Además, los vectores $x^{(0)} \in \mathbb{R}^2$ y $y_j^{(0)} \in \mathbb{R}^2$, para $j = 1,2,3$, son vectores cuyas entradas son todas iguales a 1. Utilizando los datos indicados anteriormente, el Algoritmo 5 genera una aproximación de

$$x^{(k)} = (-2.648849 \times 10^{-12}, 9.999999 \times 10^{-1})^T,$$

con $k = 29$, y un error absoluto de

$$\|x^{(29)} - x^{(28)}\|_2 = 3.358254 \times 10^{-16}.$$

Finalmente, note que

$$\|x^* - x^{(29)}\|_2 = 5.941740 \times 10^{-12},$$

lo cual indica que el Algoritmo 5 realiza una buena estimación de la solución del problema (8).

• Simulación Numérica 4: Problema FW Aplicado al Cálculo de una Imágen Representativa de una Base de Datos

A continuación, presentaremos una aplicación del problema FW aplicado a la obtención de una imagen representativa de un grupo de imágenes que tienen una característica en común. En este caso, nosotros consideraremos una base de datos de rostros frontales a escala de grises. En esta simulación, utilizaremos la base de datos de rostros FEI [24], la cual contiene 400 imágenes de rostros de tamaño 300×250 que están espacialmente normalizadas y cuyas variaciones son minimizadas debido al protocolo de adquisición utilizado, según se indica en [24]. En nuestro primer ejemplo, obtendremos la imagen representativa de 12 imágenes de dicha base de datos. Estas 12 imágenes de rostros frontales se presentan en la Figura 3.

Figura 3. Muestra de 12 imágenes de la base de datos de rostros FEI en [24].

Figura 4. Imagen promedio de las imágenes en $\mathscr{B}$.

Sean $\mathscr{B}_1 = \{I_1, I_2, ..., I_{12}\}$ el conjunto de los 12 rostros que se muestran en la Figura 3, donde $I_j \in \mathbb{R}^{300\times250}$, para todo $j = 1, 2, ..., 12$. Los pasos para poder encontrar un rostro representativo de las imágenes presentadas en la Figura 3, utilizando el problema FW, se explican a continuación:

- **Paso 1**: Calcular la imagen promedio del conjunto $\mathscr{B}$. Esta imagen se calcula con la siguiente fórmula:

$$I_{pr} = \frac{1}{12}\sum_{j=1}^{12} I_j.$$

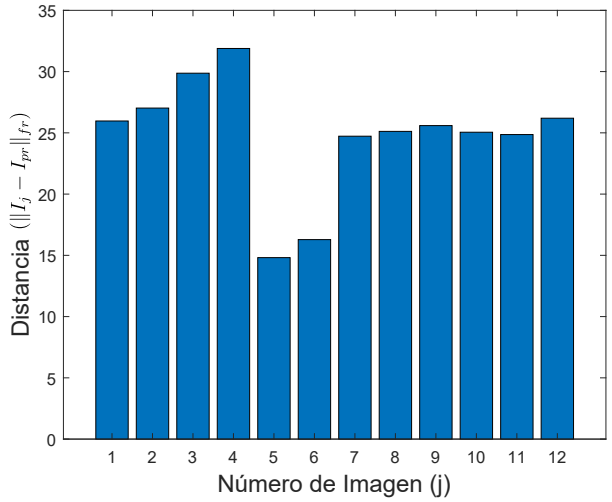
  La imagen promedio I_{pr} se presenta en la Figura 4.

- **Paso 2**: Calcular las distancias que hay entre cada una de las imágenes en $\mathscr{B}_1$ y la imagen promedio I_{pr}. Las distancias se almacenarán en un vector $d \in \mathbb{R}^{12}$, donde

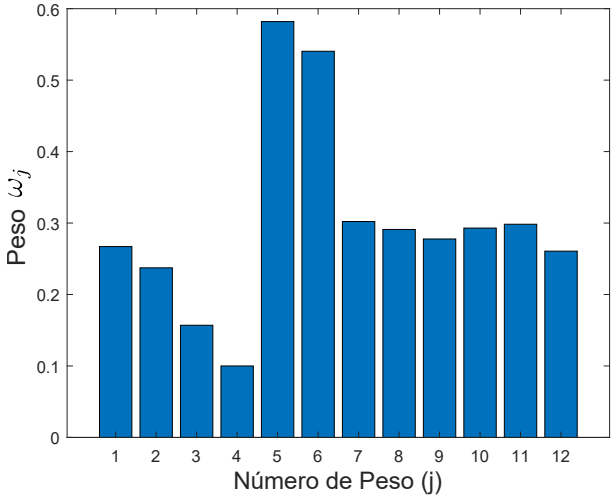
$$d(j) = \|I_j - I_{pr}\|_{fr},$$

  para todo $j = 1, 2, ..., 12$ y $\|\cdot\|_{fr}$ es la norma de Frobenius. Un gráfico de barras de las distancias se encuentra en la Figura 5(a).

- **Paso 3:** Construir los vectores (nodos) $b_1, ..., b_{12}$ y los pesos $\omega_1, ..., \omega_{12}$ para utilizar el problema FW:



(a)



(b)

Figura 5. Gráfico de barras de: (a) distancias entre las imágenes I_j y la imagen promedio I_{pr}, (b) pesos asignados a cada vector b_j

(a) (b)

Figura 6. (a) Imagen representativa del conjunto de imágenes $\mathscr{B}_1$, basado en el problema FW. (b) Imagen obtenida de (a) al aplicar inteligencia artificial generativa.

– Los nodos se construyen realizando una vectorización de cada unas de las imágenes en $\mathscr{B}_1$, es decir, transformando cada imagen en un vector columna apilando verticalmente cada una de sus columnas. Consideremos la imagen $I_j = [I_j^{(1)}\ I_j^{(2)}\ ...I_j^{(250)}]$, donde $I_j^{(i)}$ es la $i-$ésima columna de I_j, para todo $j = 1,2,...,12$, entonces

$$b_j = \text{vec}(I_j) = \begin{pmatrix} I_j^{(1)} \\ \vdots \\ I_j^{(250)} \end{pmatrix} \in \mathbb{R}^{75000},$$

donde $\text{vec}(\cdot)$ es el operador de vectorización.

– Los pesos se construyen a partir de las distancias calculadas en el Paso 2, normalizando dichas distancias con valores en el intervalo $[0,1]$. Para esto, utilizaremos el siguiente criterio: entre más cerca está una imagen I_j del promedio I_{pr}, entonces el peso se va acercando a 1; en caso contrario, entre más lejos esté una imagen I_j del promedio I_{pr}, entonces el peso se va acercando a 0. El cálculo de estos pesos permite darle más importancia a las imágenes que están más cerca de la imagen promedio I_{pr}. En este ejemplo, el valor máximo del vector d se le asigna el peso 0.1. Basado en la explicación anterior, los pesos se calculan con la siguiente fórmula:

$$\omega_i = 1 - 0.9\frac{d(i)}{d_{max}},$$

donde $d_{max} = \max\{d(1),d(2),...,d(12)\}$. Un gráfico de barras de las distancias se encuentra en la Figura 5(b).

- **Paso 4:** Utilizando los nodos y pesos obtenidos en el Paso 3, aplicamos el método de Weiszfeld, presentado en el Algoritmo 1, para encontrar la imagen representativa I_{rep} de la base de datos de rostros $\mathscr{B}_1$. La imagen I_{rep} obtenida del Algoritmo 1 se presenta en la Figura 6(a).

Observación 4. La imagen representativa presentada en la Figura 6(a) es el rostro que mejor representa a la base de datos de rostros en $\mathscr{B}_1$, basado en el problema FW que utiliza los pesos y nodos explicados en el Paso 3. Un diagrama ilustrativo de la distribución de los rostros en $\mathscr{B}_1$ y la imagen representativa I_{rep} se muestra en la Figura 7.

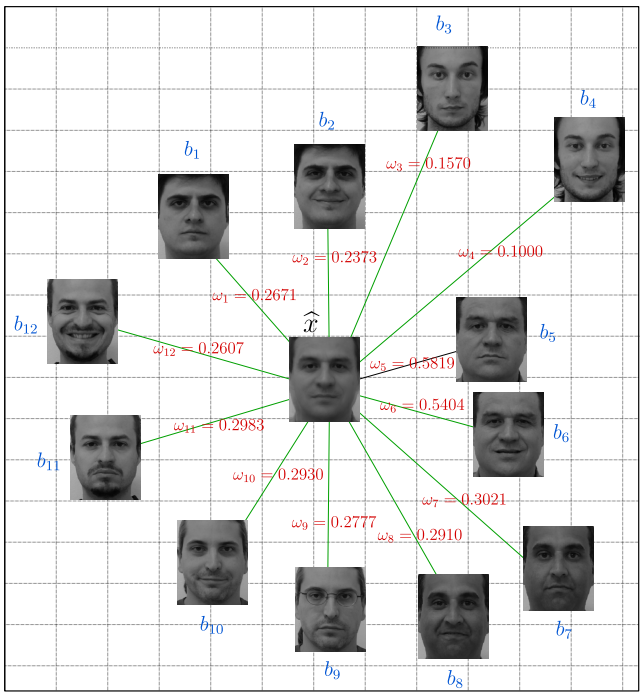


Figura 7. Representación gráfica del problema FW y su solución aplicado en el cálculo de un rostro representativo de un conjunto de imágenes de caras frontales.

Observación 5. Se puede mejorar la calidad del rostro en la Figura 6(a) si utilizamos tecnología de inteligencia artificial generativa para reparar imágenes borrosas y de baja resolución. Por ejemplo, las aplicaciones Remini [25] y Pixelup [26] son aplicaciones basadas en inteligencia artificial que aumentan la resolución de imagen sin perder calidad. Adicionalmente, se puede aumentar la resolución de la imagen utilizando *deep learning* en MATLAB [27] y Python [28]. En la Figura 6(b) se presenta la imagen obtenida usando el Algoritmo 1 después de aplicar inteligencia artificial generativa con la aplicación Remini.

Adicionalmente, presentamos un segundo ejemplo más completo donde repetimos la simulación explicada en los Paso 1 – 4 pero a todas las 400 imágenes de rostros de la base de datos de rostros FEI. Sea $\mathscr{B}_2 = \{I_1, I_2, ..., I_{400}\}$ el conjunto de imágenes de la base de datos de rostros FEI. Repitiendo los Pasos 1 – 4 explicados anteriormente, obtenemos la imagen representativa del conjunto $\mathscr{B}_2$, basado en el problema FW. Esta imagen representativa se muestra en la Figura 8(a). Por otra parte, y siguiendo la misma idea del primer ejemplo, la Figura 8(b) presenta la imagen de la Figura 8(a) después de aplicar inteligencia artificial generativa, utilizando la aplicación Remini.

Método	Algoritmo	Nombre de la Función y Sintáxis
Weiszfeld	Algoritmo 1	`[x,error,k]=metodo_weiszfeld(B,w,x0,tol)`
Weiszfeld Modificado	Algoritmo 2	`[x,error,k]=metodo_weiszfeld_mod(B,w,x0,tol)`
Newton-Görner-Kanzow	Algoritmo 3	`[x,error,k]=metodo_nkg(B,w,x0,tol)`
Problema FW con norma s	Algoritmo 4	`[x,error,k]=metodo_norma_s(B,w,x0,s,tol)`
Predictor-Corrector	Algoritmo 5	`[x,error,k]=metodo_predictor_corrector(A,B,x0,y,tol)`

Tabla 3. Funciones en MATLAB de todos los algoritmos implementados en este documento.

(a) (b)

Figura 8. (a) Imagen representativa del conjunto de imágenes $\mathscr{B}_2$, basado en el problema FW. (b) Imagen obtenida de (a) al aplicar inteligencia artificial generativa.

6. FW Toolbox

Las simulaciones numéricas presentadas en la Sección 5 fueron realizadas en MATLAB. Para cada uno de los algoritmos presentados en este documento, se realizó la implementación computacional como una función. En esta sección explicaremos la sintaxis de cada una de las funciones implementadas en MATLAB, relacionadas al problema FW. Este conjunto de funciones, llamado `FW Toolbox`, proporciona todos los algoritmos necesarios para aproximar la solución del problema FW y sus respectivas variaciones, basado en el trabajo teórico desarrollado en este documento.
En la Tabla 3 se presenta el nombre de la función y sintáxis en MATLAB de cada uno de los métodos y algoritmos estudiados en este documento. A continuación, se explicará el significado de cada uno de los parámetros de estas funciones.

- **Parámetros de entrada**:
  - `B` $= [b_1 \dots b_p]$ es una matriz de tamaño $m \times p$ que contiene los nodos.
  - `w` $= [w_1 \dots w_p]$ es un vector de tamaño p que contiene los pesos.
  - `x0` es el vector inicial de tamaño m. **Nota:** En la función `metodo_predictor_corrector`, `x0` es un vector de tamaño n.
  - `tol` es la tolerancia.
  - `s` es el número en el intervalo $[1,2]$ de la norma a utilizar en la función `metodo_norma_s`.
  - `A` $= [A_1 \dots A_p]$ es una matriz de tamaño $m \times np$. Este parámetro solo se utiliza en la función `metodo_predictor_corrector`.
  - `y` $= [y_1 \dots y_p]$ es una matriz de tamaño $m \times p$. Este parámetro solo se utiliza en la función `metodo_predictor_corrector`.
- **Parámetros de salida**:
  - `xk` es un vector de tamaño m que es la aproximación de la solución del problema asociado al método. En la función `metodo_predictor_corrector`, `x0` es un vector de tamaño n.
  - `error` es el error asociado a cada algoritmo. En este caso, el error se mide con el valor $\|x^{(k+1)} - x^{(k)}\|_2$.
  - `k` es el número de iteraciones realizadas por cada algoritmo.

Adicionalmente, se implementaron otras funciones auxiliares para cada uno de estos métodos:

- `funcion_objetivo`: dado un vector x, esta función calcula la imagen de la función $f(x)$, usando la fórmula presentada en (2).
- `grad`: dado un vector x, esta función calcula el gradiente $\nabla f(x)$, usando la fórmula presentada en (3).
- `hessiano`: dado un vector x, esta función calcula el hessiano $\nabla^2 f(x)$, usando la fórmula presentada en (4).
- `paso`: calcula el valor α_k óptimo del Algoritmo 3 (ver Pasos 14 – 21).
- `R_j`: Calcula el vector R_j, usando la fórmula presentada en (5).
- `L_j`: Calcula la constante L_j, usando la fórmula presentada en (5).
- `isInB`: Determina si un vector x es una columna o no de una matriz B.
- `valor_inicial`: Calcula el vector inicial, usando la fórmula presentada en (6).
- `actualizar`: Calcula el valor $x^{(k+1)}$, usando los Pasos 5 – 14 del Algoritmo 4.

- `entradas_iguales`: Dado un vector x y una matriz B, determina si existe algun i y j, tal que $x(j) = b_i(j)$, donde b_i es la i-ésima columna de B.
- `array2block`: Convierte un arreglo de dimension $m \times n \times p$ en una matriz de tamaño $mp \times n$, concatenando las matrices verticalmente.
- `block2array`: Convierte una matriz de tamaño $mp \times n$ en un arreglo de dimension $m \times n \times p$.
- `near_spd`: Redefine la matriz H_μ usando la fórmula presentada en (11).
- `beta`: Imagen de la función objetivo presentada en (18).
- `gamma_max`: Calcula el valor de λ, usando la fórmula presentada en (19).
- `gap`: Calcule el valor de la fórmula presentada en (21).
- `vect_rcc`: Calcula el vector r_c^c que se presenta en el (25).

Todas las funciones explicadas en esta sección se encuentran disponibles en la carpeta `FW Toolbox` del repositorio de GitHub `https://github.com/jusotoTEC/fermat_weber`.

7. Conclusiones

En este documento hemos representa un resumen relacionado a los aspectos computacionales del problema de Fermat-Weber (FW). En este documento se presentaron los algoritmos relacionados al problema FW, además de su análisis de convergencia, basado en el estudio teórico elaborado en la literatura consultada. Adicionalmente, se presentaron 2 variaciones del Problema FW, que son el problema FW con norma s, donde utilizan la norma $s \in [1,2]$ en lugar de la norma 2, y el problema FW con matrices de pesos. Al final, se realizaron un conjunto de simulaciones numéricas para conocer la eficiecia y precisión de estos métodos. Por último, se elaboró un ejemplo de procesamiento de imágenes para determinar el rostro representativo de un conjunto de imágenes de caras frontales. Adicionalmente, se utilizó una aplicación basada en inteligencia artificial generativa para obtener una imagen de mejor calidad del rostro representativo de una base de imágenes.

Conflicto de Intereses

Los autores declaran no tener ningún conflicto de intereses y asume la plena responsabilidad por el contenido del artículo.

Agradecimientos

Este trabajo fue financiado por la Vicerrectoría de Investigación y Extensión del Instituto Tecnológico de Costa Rica (Proyecto #1440042).

Referencias

[1] Alfred Weber. On the location of industries. *Progress in Human Geography*, 6(1):120–128, 1982.

[2] Charles J Alpert, Tony F Chan, Andrew B Kahng, Igor L Markov, and Pep Mulet. Faster minimization of linear wirelength for global placement. *IEEE transactions on computer-aided design of integrated circuits and systems*, 17(1):3–13, 1998.

[3] Peng Wang, Liguo Wang, Henry Leung, and Gong Zhang. Super-resolution mapping based on spatial–spectral correlation for spectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 59(3):2256–2268, 2020.

[4] Laura Montes. Métodos de optimización para el problema de localización de Fermat-Weber. Tesis para grado de licenciatura en matemática, Universidad Nacional de Córdoba, Argentina, 2018.

[5] Amir Beck and Shoham Sabach. Weiszfeld's method: Old and new results. *Journal of Optimization Theory and Applications*, 164(1):1–40, 2015.

[6] Roberto J Canavate Bernal and Ma Belén Cobacho Tornel. *Una revisión histórica de los métodos clásicos de resolución del problema de Fermat-Weber*. Disponible en `https://www.um.es/asepuma04/comunica/canavate_cobacho.pdf`.

[7] Roberto Javier Cañavate Bernal, María Belén Cobacho Tornel, José Miguel Rodríguez Gómez, et al. *El algoritmo de Weiszfeld para la resolución del problema económico de Weber*. Disponible en `https://repositorio.upct.es/xmlui/handle/10317/1507`.

[8] Gustavo Bravo Viadero. Problemas de localización. Trabajo fin de grado en matemática, Universidad de Cantabria, España, 2019.

[9] Alejandro Benítez Llambay, Pablo Benítez Llambay, and Elvio A Pilotta. On characterizing the solution for the Fermat–Weber location problem. *Universidad Nacional de Córdoba. Argentina. Facultad de Matemática, Astronomía y Física. Serie A. Trabajos de Matemática*, 2009.

[10] Simone Görner and Christian Kanzow. On Newton's method for the Fermat-Weber location problem. *Journal of Optimization Theory and Applications*, 170(1):107–118, 2016.

[11] Endre Weiszfeld. Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Mathematical Journal, First Series*, 43:355–386, 1937.

[12] Harold W Kuhn. A note on Fermat's problem. *Mathematical programming*, 4(1):98–107, 1973.

[13] Robert F Love and Wee y Yeong. A stopping rule for facilities location algorithms. *Aiie Transactions*, 13(4):357–362, 1981.

[14] Jack Brimberg and Robert F Love. Global convergence of a generalized iterative procedure for the minisum location problem with l_p distances. *Operations Research*, 41(6):1153–1163, 1993.

[15] Michael L Overton. A quadratically convergent method for minimizing a sum of Euclidean norms. *Mathematical Programming*, 27(1):34–63, 1983.

[16] Knud D Andersen, Edmund Christiansen, Andrew R Conn, and Michael L Overton. An efficient primal-dual interior-point method for minimizing a sum of Euclidean norms. *SIAM Journal on Scientific Computing*, 22(1):243–262, 2000.

[17] Liqun Qi, Defeng Sun, and Guanglu Zhou. A primal–dual algorithm for minimizing a sum of Euclidean norms. *Journal of Computational and Applied Mathematics*, 138(1):127–150, 2002.

[18] Guoliang Xue and Yinyu Ye. An efficient algorithm for minimizing a sum of Euclidean norms with applications. *SIAM Journal on Optimization*, 7(4):1017–1036, 1997.

[19] Knud D Andersen. An efficient Newton barrier method for minimizing a sum of Euclidean norms. *SIAM Journal on Optimization*, 6(1):74–95, 1996.

[20] Liqun Qi and Guanglu Zhou. A smoothing Newton method for minimizing a sum of Euclidean norms. *SIAM Journal on Optimization*, 11(2):389–410, 2000.

[21] Robert J Plemmons and Stephen J Wright. An efficient parallel scheme for minimizing a sum of Euclidean norms. *Linear Algebra and its Applications*, 121:71–85, 1989.

[22] Guanglu Zhou, KC Toh, and Defeng Sun. Globally and quadratically convergent algorithm for minimizing the sum of Euclidean norms. *Journal of optimization theory and applications*, 119(2):357–377, 2003.

[23] Nicholas J Higham. Computing a nearest symmetric positive semidefinite matrix. *Linear algebra and its applications*, 103:103–118, 1988.

[24] Image Processing Laboratory in Centro Universitario da FEI. *FEI Face database*. Disponible en `https://fei.edu.br/~cet/facedatabase.html`.

[25] Splice Video Editor S.r.l. *Remini*. Disponible en `https://remini.ai/`.

[26] Codeway Dijital. *Pixelup*. Disponible en `https://play.google.com/store/apps/details?id=com.codeway.pixelup`.

[27] MathWorks. *Increase Image Resolution Using Deep Learning*. Disponible en `https://la.mathworks.com/help/images/single-image-super-resolution-using-deep-learning.html`.

[28] Xavier Weber. *Deep Learning based Super Resolution with OpenCV*. Disponible en `https://towardsdatascience.com/deep-learning-based-super-resolution-with-opencv-4fd736678066`.

Este preprint fue presentado bajo las siguientes condiciones:

- Los autores declaran que son conscientes de que son los únicos responsables del contenido del preprint y que el depósito en SciELO Preprints no significa ningún compromiso por parte de SciELO, excepto su preservación y difusión.
- Los autores declaran que se obtuvieron los términos necesarios del consentimiento libre e informado de los participantes o pacientes en la investigación y se describen en el manuscrito, cuando corresponde.
- Los autores declaran que la preparación del manuscrito siguió las normas éticas de comunicación científica.
- Los autores declaran que los datos, las aplicaciones y otros contenidos subyacentes al manuscrito están referenciados.
- El manuscrito depositado está en formato PDF.
- Los autores declaran que la investigación que dio origen al manuscrito siguió buenas prácticas éticas y que las aprobaciones necesarias de los comités de ética de investigación, cuando corresponda, se describen en el manuscrito.
- Los autores declaran que una vez que un manuscrito es postado en el servidor SciELO Preprints, sólo puede ser retirado mediante solicitud a la Secretaría Editorial deSciELO Preprints, que publicará un aviso de retracción en su lugar.
- Los autores aceptan que el manuscrito aprobado esté disponible bajo licencia Creative Commons CC-BY.
- El autor que presenta el manuscrito declara que las contribuciones de todos los autores y la declaración de conflicto de intereses se incluyen explícitamente y en secciones específicas del manuscrito.
- Los autores declaran que el manuscrito no fue depositado y/o previamente puesto a disposición en otro servidor de preprints o publicado en una revista.
- Si el manuscrito está siendo evaluado o siendo preparando para su publicación pero aún no ha sido publicado por una revista, los autores declaran que han recibido autorización de la revista para hacer este depósito.
- El autor que envía el manuscrito declara que todos los autores del mismo están de acuerdo con el envío a SciELO Preprints.