Universidad de Sevilla

Facultad de Matemáticas

Trabajo Fin de Estudios

Grado en Matemáticas

Programación Lineal Cónica

Ramón Pascual Peña

Tutor

Antonio Beato Moreno

Dpto. de Estadística e Investigación Operativa

Abstract

The present work is focused on the analysis of conic linear programming problems. For this purpose, the theoretical treatment of these problems, as well as their different types, their characteristics and the algorithms applied to solve them will be addressed. Regarding the latter, the automated resolution of such problems by means of software with the use of the ROI package of R will be dealt with. We will show its characteristics of use, as well as its application to different types of specific problems. This allows us to illustrate how the algorithms are used in practice by means of the operation of different solvers that must be chosen in an appropriate way according to the type of problem tackled.

Resumen

El presente trabajo de fin de grado se centra en el análisis de los problemas de programación lineal cónica. Para ello se aborda el tratamiento teórico de dichos problemas, así como sus distintos tipos, sus características y los algoritmos aplicados a su resolución. En cuanto a esto último se trata la resolución automatizada de dichos problemas mediante software con el uso del paquete ROI de R, del que se muestran sus características de uso, así como su aplicación a distintos tipos de problemas concretos que permiten ilustrar cómo se emplean en la práctica los algoritmos mediante el funcionamiento de distintos solvers que se deben elegir de manera adecuada en función del tipo de problema que se aborde.

Índice

1. Introducción a los problemas de programación matemática **3**
1.1. Problemas de Programación Matemática 3
1.2. Conos convexos 5
1.3. El problema de programación lineal cónica 7

2. Propiedades de los problemas lineales cónicos **12**
2.1. Teorema de separación en Programación Lineal Cónica 12
2.2. Dualidad en programación lineal cónica.
Ejemplos 14
2.3. Complementariedad 23

3. Algoritmos de resolución de problemas en programación lineal cónica **24**
3.1. Complementariedad en programación semidefinida 24
3.2. Métodos de punto interior 29
3.3. Otros métodos 37

4. Resolución de problemas lineales cónicos mediante el entorno de software R. **40**
4.1. Paquetes específicos 41
4.2. Construcción y resolución de problemas 42
4.2.1. Problema Lineal 43
4.2.2. Problema Lineal con Solución Múltiple 48
4.2.3. Problema SDP 50
4.2.4. Problema Cono de Segundo Orden 51
4.3. Resolución de un problema mediante diferentes métodos 55
4.3.1. Resolución como Problema de Programación Lineal 55
4.3.2. Resolución como Problema SDP 57
4.3.3. Resolución como Problema de Cono de Segundo Orden 59

Conclusiones **61**

Bibliografía **62**

Índice de figuras

1.1. Vistas de la región factible del cono de matrices semidefinidas positivas . . 6
1.2. Vista del cono de segundo orden 6

3.1. Problema de localización de redes de sensores en cadena 29
3.2. Función de penalización cuadrática 32
3.3. Función de penalización de valores absolutos 32
3.4. Conjuntos robustos y no robustos 33

Capítulo 1

Introducción a los problemas de programación matemática

1.1. Problemas de Programación Matemática

La programación matemática es una parte de la matemática que se dedica a estudiar cómo describir la situación que se plantea en un problema en el que intervienen determinadas magnitudes que interactúan entre ellas para alcanzar el mejor valor posible que, cuando se dispone de una medida cuantitativa, es lo que se denomina valor óptimo. La relaciones entre las magnitudes están limitadas por las características del problema concreto que se aborda, restringiendo los posibles valores que pueden ser válidos en dicho problema.

La programación matemática se desarrolla a partir de la Investigación Operativa (IO), disciplina científca que estudia la toma de mejores decisiones mediante la aplicación de métodos analíticos avanzados, cuyos orígenes son muy antiguos debido a la necesidad de encontrar solución al problema de optimizar los recursos cuando estos son limitados.

Tanto el origen de lo que se conoce como investigación operativa como su denominación como tal de manera formal se encuentran en el ámbito militar, en las proximidades del comienzo de la Segunda Guerra Mundial, cuando se crea en la estación de radar de Suffolk un grupo de investigación militar con el objetivo de mejorar el sistema de detección de aviones enemigos y cuyos primeros avances dieron lugar a su aprovechamiento por parte de la *Royal Air Force*. Fue Albert P. Row, militar de la citada estación quien, en 1938, introdujo el término *Operational Research* y el físico Patrick M.S. Blackett de la RAF quien dirigió un grupo que aplicó con eficacia estos métodos para la colocación estratégica

de cargas de profundidad en la lucha contra los submarinos enemigos.

En la misma época se abordan desde el ámbito civil investigaciones que contribuyen notablemente al desarrollo de la programación matemática, como son el tratamiento de sistemas de desigualdades lineales, la búsqueda de una solución al que se conoce como el "problema del viajero", el inicio de la teoría de grafos, ...

Merece la pena resaltar el abordaje mediante la programación lineal de la solución del conocido como *problema del transporte* para enviar, al mínimo coste, productos desde sus lugares de producción o almacenaje a los mercados. Aunque este problema tiene su origen en una versión anterior conocida como *el problema del transporte de masa* ya planteada por el matemático francés Gaspard Monge a finales del siglo XVIII, fue Frank L. Hitchcock quien en 1941 formalizó la versión actual de dicho problema y propuso un esbozo del algoritmo para su solución. No obstante, fue George B. Dantzig quien realizó una precisa formulación del problema, así como el desarrollo teórico y su resolución posterior.

En los años 50, las aportaciones de Dantzig y otros como Koopmans, Kantorovich o Kuhn contribuyeron notablemente al desarrollo de la programación lineal consolidándola como disciplina diferenciada de la Investigación Operativa. Entre estas aportaciones cabe destacar el desarrollo del método del simplex, que es considerado uno de los diez algoritmos más importantes del siglo pasado.

No es hasta 1963 con la publicación del libro *Linear programming and extesions* por parte de Dantzig, cuando se concibe la programación matemática como la conocemos en la actualidad.

Actualmente, gracias al uso de las nuevas tecnologías, podemos resolver problemas con una cantidad inmensa de datos, lo que ha permitido un auge de la programación matemática, que a su vez se plantea como uno de los pilares fundamentales en el desarrollo de la inteligencia artificial.

El objetivo de la Programación Matemática es el estudio teórico de los problemas de optimización, así como de la puesta en marcha de los algoritmos de resolución cuando los objetivos y el conjunto factible pueden formalizarse mediante funciones reales de variable real. La formulación estandar de un problema de este tipo es:

$$\begin{aligned} \text{minimizar} \quad & f(x) \\ \text{sujeto a:} \quad & g(x) \leq 0 \\ & h(x) = 0 \\ & x \in X. \end{aligned}$$

donde $f : \mathbb{R}^n \longrightarrow \mathbb{R}^p$, $\quad q : \mathbb{R}^n \longrightarrow \mathbb{R}^m$, $\quad h : \mathbb{R}^n \longrightarrow \mathbb{R}^l$ y $X \in \mathbb{R}^n$.

1.2. Conos convexos

En esta sección procederemos a tratar el concepto de cono convexo y de determinadas propiedades de este.

El concepto de convexidad se aplica a aquellos conjuntos en los que se verifica la propiedad de que el segmento que une cualesquiera dos puntos del conjunto está completamente incluido en dicho conjunto. De manera más formal, se dice que un conjunto $K \subset \mathbb{E}^n$ es convexo si $\forall x, y \in K$ y $\forall \lambda \in \mathbb{R}, 0 \leq \lambda \leq 1$ se verifica $\lambda x + (1-\lambda)y \in K$.

La convexidad de conjuntos se conserva con respecto a determinadas operaciones como se refleja en las siguientes propiedades.

Sean $C, D \subset \mathbb{E}^n$ conjuntos convexos, $\alpha \in \mathbb{R}$ entonces:

1. $\alpha C = \{x \, : \, x = \alpha c, c \in C\}$ es convexo.
2. $C + D = \{x \, : \, x = c + d, c \in C, d \in D\}$ es convexo.
3. $C \cap D$ es convexo.

En cuanto a la figura de cono, podemos definirla de la siguiente manera:
Consideramos que un conjunto $C \subset \mathbb{E}^n$ es un cono si $\forall c \in C$ y $\forall \alpha \in \mathbb{R}, \alpha > 0$ se verifica que $\alpha c \in C$.

Esto significa que en un cono, si tomamos un punto cualquiera de la semirrecta que une el vértice del cono y dicho punto, dicha semirrecta está íntegramente contenida, a excepción del vértice, en el mismo.

Con respecto al concepto de cono convexo, se trata de aplicar el concepto de conjunto convexo al caso del cono, es decir se entiende por cono convexo aquel conjunto que es cono y conjunto convexo a la vez.

Por tanto, de manera formal consideramos que cono convexo es un subconjunto $C \subset \mathbb{E}^n$ que es convexo, no vacío y tal que si $c \in C$, entonces $\alpha c \in C$, $\forall \alpha > 0$.

Una vez definido el concepto de cono convexo, veamos varios ejemplos de los conos más utilizados en la programación lineal cónica:

1. El ortante n-dimensional no negativo, $E^n_+ = \{\mathbf{x} \in E^n \, : \, \mathbf{x} \geq \mathbf{0}\}$.
2. *El cono de matrices semidefinidas positivas* S^n_+, formado por el conjunto de todas las matrices n-dimensionales, simétricas y semidefinidas positivas.

   En el caso de R^3, la región factible está formada por el interior y la frontera de la figura que se representa con distintas vistas a continuación:

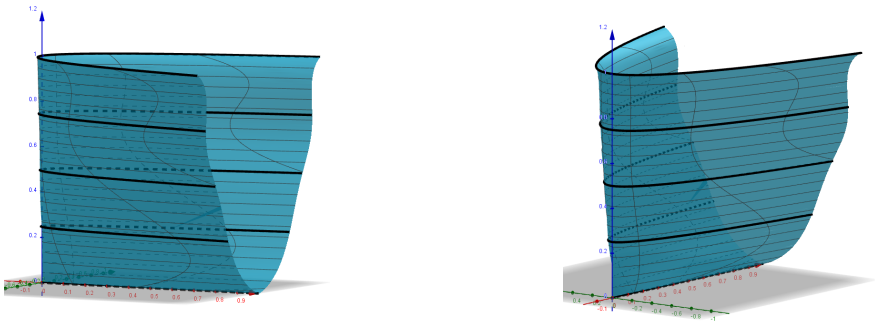

Figura 1.1: Vistas de la región factible del cono de matrices semidefinidas positivas

3. *El cono de orden-p* formado por el conjunto $C = \{(u; \mathbf{x}) \in E^{n+1} : u \geq |\mathbf{x}|_p\}$ es un cono convexo en E^{n+1}, con $1 \leq p < \infty$, donde $|\mathbf{x}|_p$ es la norma de orden p del vector $\mathbf{x}$, es decir, si $\mathbf{x} \in E^n$ entonces $|\mathbf{x}|_p = \sqrt[p]{\sum_{i=1}^{n} |x_i|^p}$

   Si $p = 2$ tenemos un cono de segundo orden, obtenido de la siguiente manera: $C = \{(z; (x, y)) \in \mathbb{R}^3 : z \geq ||(x, y)||_2\} = \{(z; (x, y)) \in \mathbb{R}^3 : z \geq \sqrt{x^2 + y^2}\}$ que se corresponde con el interior de la figura que se representa a continuación:

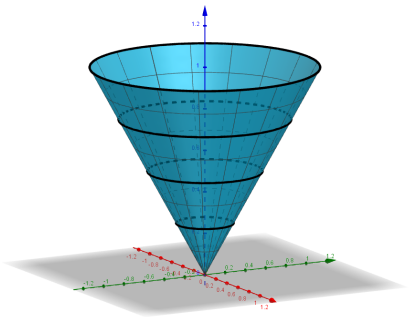

Figura 1.2: Vista del cono de segundo orden

En cuanto al concepto del dual de un cono, resulta necesario, antes de pasar a definirlo, abordar las siguientes definiciones:

Dadas las matrices $A, B \in E^{(n+1)\times(n+1)}$, definimos el producto interior como:

$$\mathbf{A} \bullet \mathbf{B} = traza(\mathbf{A}^t\mathbf{B}) = \sum_{i,j} a_{ij}b_{ij}$$

La norma de Frobenius es la norma matricial asociada al producto interior definida por:

$$|\mathbf{X}|_f = \sqrt{\mathbf{X} \bullet \mathbf{X}}$$

Por consiguiente de manera formal tenemos que el dual de un cono K es:

$$K^* = \{\mathbf{Y} : \mathbf{X} \bullet \mathbf{Y} \geq 0 \text{ para todo } \mathbf{X} \in K\}$$

Podemos ver los casos de cono dual para los ejemplos vistos anteriormente:

- Si se trata del cono $K = E^n_+$, el ortante n-dimensional no negativo, el cono dual se reduce a sí mismo.
- Si K es el cono de matrices semidefinidas positivas S^n_+, al igual que en el caso anterior el cono dual se reduce a él mismo.
- Si K es el cono de orden p, su dual sería el cono de orden q tal que:

$$\frac{1}{p} + \frac{1}{q} = 1$$

1.3. El problema de programación lineal cónica

La programación lineal cónica es una extensión de la programación lineal cuyos componentes son puntos de un cono convexo, cuya definición se ha visto en el apartado anterior.

A pesar de que se conoce que los PLC son problemas de optimización convexa, no han sido de vital importancia hasta principios de este siglo, cuando se desarrolló el método de punto interior, el cual desarrollaremos en la sección 3.2.

En la programación lineal cónica la formulación estándar de un problema (PLC) se construye de la siguiente manera:

Sean las matrices $\mathbf{C}, \mathbf{A}_i \in E^{k\times n}$, $i = 1, 2, \dots m$, $\quad b \in E^m$, $K \in E^{k\times n}$ un cono convexo cerrado. Sea $\mathbf{X} \in E^{k\times n}$ una matriz desconocida. La formulación del problema es:

$$\begin{aligned} \text{(PLC)} \quad & \text{minimizar} \quad \mathbf{C} \bullet \mathbf{X} \\ & \text{sujeto a:} \quad \mathbf{A}_i \bullet \mathbf{X} = b_i,\ i = 1, 2, \dots, m,\ \mathbf{X} \in K. \end{aligned} \tag{1.1}$$

Existen otras formulaciones más compactas del problema PLC que pueden obtenerse definiendo un operador que transforma una matriz simétrica en un vector:

$$\mathcal{A}\mathbf{X} = \begin{pmatrix} A_1 \bullet \mathbf{X} \\ A_2 \bullet \mathbf{X} \\ \dots \\ A_m \bullet \mathbf{X} \end{pmatrix}$$

De esta manera, el problema PLC se escribe de forma compacta como:

$$\begin{aligned} \text{(PLC)} \quad & \text{minimizar} \quad \mathbf{C} \bullet \mathbf{X} \\ & \text{sujeto a:} \quad \mathcal{A}\mathbf{X} = \mathbf{b},\ \mathbf{X} \in K \end{aligned} \tag{1.2}$$

En los ejemplos de conos citados anteriormente podemos ver cómo realizar una formulación más compacta del problema:

- Si se trata del cono $k = E_+^n$, el ortante n-dimensional no negativo, PLC se reduce a un problema de programación lineal estándar, donde $\mathcal{A}$ se convierte en la matriz de restricciones $\mathbf{A}$.
- Si K es el cono de matrices semidefinidas positivas S_+^n, PLC se convierte en un caso de programación semidefinida (SDP).
- Si K es el cono de orden p, PLC se trata de un problema de programación cónica de orden p.

A continuación vamos a ver diferentes ejemplos de problemas de programación lineal cónica:

Ejemplo 1 Este ejemplo sirve para ilustrar las diferentes formas de enunciar un problema de programación lineal.

Supongamos un problema de programación lineal cuya formulación estándar sea:

$$\begin{aligned} \text{minimizar} \quad & 2x_1 + x_2 + x_3 \\ \text{sujeto a:} \quad & x_1 + x_2 + x_3 = 1 \\ & (x_1, x_2, x_3) \geq \mathbf{0}. \end{aligned}$$

La formulación de este mismo problema como un problema de programación semidefinida requiere que la dimensión de la matriz $\mathbf{X}$ sea dos:

$$\begin{aligned} \text{minimizar} \quad & 2x_1 + x_2 + x_3 \\ \text{sujeto a:} \quad & x_1 + x_2 + x_3 = 1 \\ & \begin{bmatrix} x_1 & x_2 \\ x_2 & x_3 \end{bmatrix} \succeq \mathbf{0}. \end{aligned}$$

La formulación del problema como SPD sería:

$$\begin{aligned} \text{minimizar} \quad & \mathbf{C} \bullet \mathbf{X} \\ \text{sujeto a:} \quad & \mathbf{A}_1 \bullet \mathbf{X} = 1, \, \mathbf{X} \in S_+^2, \end{aligned}$$

donde:

$$\mathbf{C} = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 1 \end{bmatrix} \qquad \mathbf{A}_1 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}.$$

Por último como problema de cono de segundo orden se tendría:

$$\begin{aligned} \text{minimizar} \quad & 2x_1 + x_2 + x_3 \\ \text{sujeto a:} \quad & x_1 + x_2 + x_3 = 1 \\ & \sqrt{x_2^2 + x_3^2} \leq x_1. \end{aligned}$$

Ejemplo 2 En este ejemplo vamos a ver un caso de optimización cuadrática binaria

La formulación inicial sería:

$$\begin{aligned} &\text{maximizar} && \mathbf{x}^t\mathbf{Q}\mathbf{x} + 2\mathbf{c}^t\mathbf{x} \\ &\text{sujeto a:} && x_j = \{1, -1\},\ j = 1, \dots, n. \end{aligned}$$

Dado que este es un problema complicado de resolver lo podemos reescribir como:

$$\begin{aligned} z^* \equiv \quad &\text{maximizar} && \begin{bmatrix} \mathbf{x} \\ \mathbf{1} \end{bmatrix}^t \begin{bmatrix} \mathbf{Q} & \mathbf{c} \\ \mathbf{c}^t & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{1} \end{bmatrix} \\ &\text{sujeto a:} && (x_j)^2 = 1, \quad j = 1, \dots, n. \end{aligned}$$

Una vez escrito de la manera anterior puede transformarse mediante su formulación como un problema cuadrático binario homogéneo:

$$\begin{aligned} z^* \equiv \quad &\text{maximizar} && \begin{bmatrix} \mathbf{Q} & \mathbf{c} \\ \mathbf{c}^t & \mathbf{0} \end{bmatrix} \bullet \begin{bmatrix} \mathbf{x} \\ x_{n+1} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ x_{n+1} \end{bmatrix}^t \\ &\text{sujeto a:} && \mathbf{I}_j \bullet \begin{bmatrix} \mathbf{x} \\ x_{n+1} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ x_{n+1} \end{bmatrix}^t = 1, \quad j = 1, \dots, n+1, \end{aligned}$$

donde $\mathbf{I}_j$ es la matriz $(n+1) \times (n+1)$ de componentes nulas exceptuando la posición j-ésima de la diagonal principal donde tenemos un 1.

Sea $(\mathbf{x}^*; x^*_{n+1})$ una solución del problema homogéneo, entonces se tendría que $\mathbf{x}^*/x^*_{n+1}$ sería solución del problema original.

Ejemplo 3 Localización de sensores en red

En este problema se abordará cómo determinar la posición de los sensores de una red cuando se conoce cuál es la separación que hay entre algunos de ellos a través de la medida de la distancia euclídea, pero se desconoce su ubicación.

Supongamos que los dispositivos se encuentran ubicados de forma que n puntos distintos representan las posiciones de dichos dispositivos. Denotemos estos puntos mediante $\mathbf{x}_j \in E^d$, $j = 1, \dots, n$. Consideremos N_e como el subconjunto de aristas que unen cada punto de los citados $\mathbf{x}_i$ con otro $\mathbf{x}_j$, representadas mediante los pares (i, j) y cuya distancia entre ellos d_{ij} es conocida.

El problema de la localización consiste en encontrar las posiciones $\mathbf{x}_j$, $j = 1, \dots, n$, tales que:

$$|\mathbf{x}_i - \mathbf{x}_j|^2 = (d_{ij})^2,\ \forall (i, j) \in N_e.$$

Sea $\mathbf{X} = [\mathbf{x}_1\, \mathbf{x}_2\, \dots\, \mathbf{x}_n]$ la matriz $d \times n$ a determinar. Entonces:

$$|\mathbf{x}_i - \mathbf{x}_j|^2 = (\mathbf{e}_i - \mathbf{e}_j)^t \mathbf{X}^t \mathbf{X} (\mathbf{e}_i - \mathbf{e}_j),$$

con $\mathbf{e}_i \in E^n$ el vector formado por un 1 en la i-ésima posición y ceros en el resto.

Usamos la matriz $\mathbf{Y} = \mathbf{X}^t\mathbf{X}$ de orden n como un cambio de variable, lo que nos permite replantear el problema de localización de forma que este consiste en encontrar $\mathbf{Y}$ tal que:

$$(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^t \bullet \mathbf{Y} = (d_{ij})^2, \forall (i,j) \in N_e,$$

$$\mathbf{Y} \succeq \mathbf{0}.$$

lo que ha supuesto una relajación del problema inicial mediante el empleo de una matriz semidefinida positiva.

En este problema la función objetivo es nula, por tanto se reduce a encontrar una solución factible que cumpla las restricciones. Ahora bien, si las distancias empleadas corresponden a mediciones no exactas, se puede emplear una función objetivo que minimice el error total y también añadir variables de holgura adicionales. La formulación anterior del problema contemplando estos cambios se transforma en:

$$\begin{array}{rl} \text{minimizar} & \sum_{(i,j)\in N_e} |z_{ij}| \\ \text{sujeto a:} & (\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^t \bullet \mathbf{Y} + z_{ij} = (d_{ij})^2, \quad \forall (i,j) \in N_e \\ & \mathbf{Y} \succeq \mathbf{0}. \end{array}$$

El problema se puede convertir en un problema de programación lineal cónica mediante la utilización de conos semidefinidos y el ortante no negativo.

Ejemplo 4 Localización de sensores en red con anclas

En las redes de sensores, aquellos cuya posición es conocida se denominan anclas. Usando la notación del ejemplo anterior sobre localización de sensores en red, denotamos a las anclas por $(\mathbf{a}_1, \dots \mathbf{a}_{d+1})$. Así pues, este problema consiste en determinar los vectores de posición $\mathbf{x}_j$, $j = d+2, d+3, \dots, n$, tales que:

$$|\mathbf{x}_i - \mathbf{x}_j|^2 = (d_{ij})^2, \forall (i,j) \in N_e$$
$$|\mathbf{a}_a - \mathbf{x}_j|^2 = (d_{aj})^2, \forall (a,j) \in N_a,$$

si representamos con N_a el subconjunto de aristas entre anclas y sensores (a,j) $(a = 1, \dots, d+1)$, cuya distancia d_{aj} es conocida.

En este caso el planteamiento se transforma en un problema de programación semidefinida de la siguiente manera:

$$(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^t \bullet \mathbf{Y} = (d_{ij})^2,\ \forall (i,j) \in N_e$$

$$\begin{pmatrix} \mathbf{a}_a \\ -\mathbf{e}_j \end{pmatrix} \begin{pmatrix} \mathbf{a}_a \\ -\mathbf{e}_j \end{pmatrix}^t \bullet \begin{pmatrix} I_d & \mathbf{X} \\ \mathbf{X}^t & \mathbf{Y} \end{pmatrix} = (d_{aj})^2,\ \forall (a,j) \in N_a$$

$$\mathbf{Z} = \begin{pmatrix} I_d & \mathbf{X} \\ \mathbf{X}^t & \mathbf{Y} \end{pmatrix} \succeq \mathbf{0}.$$

Con I_d la matriz identidad de dimensión d y $\mathbf{Z}$ la matriz simétrica de dimensión $d+n$. Si el rango de la matriz $\mathbf{Z}$ es d, tendremos las localizaciones exactas.

Capítulo 2

Propiedades de los problemas lineales cónicos

2.1. Teorema de separación en Programación Lineal Cónica

Resulta conveniente abordar de manera previa el concepto de interior de un cono. Para ello veamos qué se considera punto interior de un cono.

Definición 2.1 Se dice que $\mathbf{X}$ es un punto interior de un cono K, cuando y solo cuando se verifique que para cualquier $\mathbf{Y}$ perteneciente al cono dual K^* si $\mathbf{Y} \bullet \mathbf{X} = 0 \Rightarrow \mathbf{Y} = \mathbf{0}$.

Se denota por $\overset{\circ}{K}$ al conjunto de puntos interiores de K.

El interior de los conos que se citaron a modo de ejemplo en la sección anterior se muestra a continuación:

- Para el cono E^n_+, el ortante n-dimensional no negativo, su interior es el conjunto de vectores cuyas entradas son todas positivas.
- En el caso del cono de matrices semidefinidas positivas S^n_+, su interior es el conjunto de todas las matrices definidas positivas.
- Por último, para el cono de orden p, su interior es el conjunto $\{(u; \mathbf{x}) \in E^{n+1} : u > |\mathbf{x}|_p\}$

Para los puntos interiores de un cono resulta necesario citar la siguiente proposición que se da para ellos:

Proposición 2.2 Sea K un cono, entonces si $\mathbf{X} \in \mathring{K}$ e $\mathbf{Y} \in K^*$, para toda constante $c \geq 0$, si $\mathbf{Y} \bullet \mathbf{X} \leq c$, entonces $\mathbf{Y}$ está acotado.

Resulta conveniente realizar un análisis de la región factible para los casos de problemas de PLC:

En estos casos la región factible será:

$$\mathcal{F} = \{\mathbf{X} : \mathcal{A}\mathbf{X} = \mathbf{b},\ \mathbf{X} \in K\},$$

cuyo interior es:

$$\mathring{\mathcal{F}} = \{\mathbf{X} : \mathcal{A}\mathbf{X} = \mathbf{b},\ \mathbf{X} \in \mathring{K}\}.$$

En el caso de que $\mathcal{F} = \emptyset$ siendo el cono $K = E^n_+$, aplicando el lema de Farkas para programación lineal, siempre existirá un vector $\mathbf{y} \in E^m$ con $\mathbf{y}^t\mathbf{A} \leq \mathbf{0}$ e $\mathbf{y}^t\mathbf{b} > 0$, lo que imposibilita que el sistema $\{\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ tenga solución.

Para el caso de programación lineal cónica, analicemos si estas relaciones alternativas se verifican en caso de que K se trate de un cono cerrado convexo. Definimos un operador que transforma un vector en una matriz:

$$\mathbf{y}^t\mathcal{A} = \sum_{i=1}^{m} \mathbf{A}_i y_i.$$

Por tanto, para cualquier matriz $\mathbf{X} \in E^{k\text{x}n}$, se cumple la propiedad asociativa:

$$\mathbf{y}^t\mathcal{A} \bullet \mathbf{X} = \mathbf{y}^t(\mathcal{A}\mathbf{X}).$$

Y aplicando la trasposición de matrices $(\mathbf{y}^t\mathcal{A})^t = \mathcal{A}\mathbf{y}^t$.

Las preguntas que habría que formularse para este caso serían:

- Cuando $\mathcal{F} = \emptyset$, ¿existe un vector $\mathbf{y} \in E^m$ tal que $-\mathbf{y}^t\mathcal{A} \in K^*$ e $\mathbf{y}^t\mathbf{b} > 0$?
- Cuando el conjunto $\{\mathbf{y} : \mathbf{C}^t - \mathbf{y}^t\mathcal{A} \in K\} = \emptyset$, ¿existe una matriz $\mathbf{X} \in K^*$ tal que $\mathcal{A}\mathbf{X} = 0$ y $\mathbf{C} \bullet \mathbf{X} < 0$?

Para la segunda pregunta, si se trata del cono $K = E^n_+$, la respuesta es afirmativa, sin embargo en ambas cuestiones, cuando se trata de un cono cualquiera, la respuesta es que no necesariamente se da la existencia de lo que cada cuestión propone.

Sin embargo, si se le imponen condiciones adicionales a $\mathcal{A}$ la respuesta sería afirmativa como puede verse en el Lema de Farkas para PLC.

Teorema 2.3 *Lema de Farkas para PLC*

- *Sea el conjunto*

$$\mathcal{F}_p = \{\mathbf{X} : \mathcal{A}\mathbf{X} = \mathbf{b}, \mathbf{X} \in K\}.$$

Supongamos que existe un vector $\mathring{\mathbf{y}}$ *tal que* $-\mathring{\mathbf{y}}^t\mathcal{A} \in \mathring{K}^*$. *Entonces:*

1. *El conjunto* $C = \{\mathcal{A}\mathbf{X} \in E^m : \mathbf{X} \in K\}$ *es convexo cerrado.*
2. $\mathcal{F}_p$ *tiene una solución factible si y solo si el conjunto* $\{\mathbf{y} : -\mathbf{y}^t\mathcal{A} \in K^*, \mathbf{y}^t\mathbf{b} > 0\}$ *no posee soluciones factibles.*

- *Sea el conjunto*

$$\mathcal{F}_d = \{\mathbf{y} : \mathbf{C}^t - \mathbf{y}^t\mathcal{A} \in K\}.$$

Supongamos que existe un vector $\mathring{\mathbf{X}} \in \mathring{K}^*$ *tal que* $\mathcal{A}\mathring{\mathbf{X}} = \mathbf{0}$. *Entonces:*

1. *El conjunto* $C = \{\mathbf{S} + \mathbf{y}^t\mathcal{A} : \mathbf{S} \in K\}$ *es convexo cerrado.*
2. $\mathcal{F}_d$ *tiene solución factible si y solo si, el conjunto* $\{\mathbf{X} : \mathcal{A}\mathbf{X} = \mathbf{0}, \mathbf{X} \in K^*, \mathbf{C} \bullet \mathbf{X} < 0\}$ *no posee soluciones factibles.*

La prueba de este teorema puede hallarse en Luenberger, David G; Ye, Yinyu, y Cham, Springer; 2021, [1].

2.2. Dualidad en programación lineal cónica. Ejemplos

Como la programación lineal cónica es una extensión de la programación lineal clásica, existiría de manera natural un problema dual al problema primal constituyendo este problema dual también un problema de programación lineal cónica. De hecho, en programación lineal cónica, la relación entre el problema dual y el primal se da del mismo modo y en mayor medida, que la existente entre los problemas primal y dual en el caso de la programación lineal clásica.

El problema dual del problema primal en PLC viene dado por:

$$\begin{aligned} \text{(DLC)} \quad & \text{maximizar} \quad \mathbf{y}^t\mathbf{b} \\ & \text{sujeto a:} \quad \textstyle\sum_i^m yi\mathbf{A}_i + \mathbf{S} = \mathbf{C}^t,\ \mathbf{S} \in K^*. \end{aligned} \tag{2.1}$$

Que escrito de forma simplificada:

$$\begin{aligned} \text{(DLC)} \quad & \text{maximizar} \quad \mathbf{y}^t\mathbf{b} \\ & \text{sujeto a:} \quad \mathbf{y}^t\mathcal{A} + \mathbf{S} = \mathbf{C}^t,\ \mathbf{S} \in K^*. \end{aligned} \tag{2.2}$$

con **S** una matriz que representa las variables de holgura, por lo que el problema se puede expresar, sin usar estas variables, como:

$$\begin{array}{lll} \text{(DLC)} & \text{maximizar} & \mathbf{y}^t\mathbf{b} \\ & \text{sujeto a:} & \sum_i^m yi\mathbf{A}_i \preccurlyeq_{K^*} \mathbf{C}^t. \end{array} \tag{2.3}$$

Tal y como ocurre en la programación lineal clásica, en la programación lineal cónica el dual del problema dual (PDLC) es el problema primal (PPLC).

Podemos ilustrar mediante ejemplos, los casos de distintos problemas duales en programación lineal cónica (PDLC) para ejemplos vistos anteriormente.

Ejemplo 1 En este ejemplo se ilustran los problemas duales del ejemplo 1 anterior. [1]

Se tratarán tres casos distintos, aunque en todos ellos y es un escalar.

a. El dual para un problema de programación lineal con formulación estandar es:

$$\begin{array}{rl} \text{maximizar} & y \\ \text{sujeto a:} & y(1,1,1) + (s_1, s_2, s_3) = (2,1,1), \\ & \mathbf{s} = (s_1, s_2, s_3) \in K^* = E^3_+. \end{array}$$

b. El dual del problema de programación semidefinida es:

$$\begin{array}{rl} \text{maximizar} & y \\ \text{sujeto a:} & y\mathbf{A}_1 + \mathbf{S} = \mathbf{C}, \\ & \mathbf{S} \in K^* = S^2_+, \end{array}$$

donde:

$$\mathbf{C} = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 1 \end{bmatrix} \quad \text{y} \quad \mathbf{A}_1 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}.$$

c. Por último el dual del problema del cono de segundo orden es:

$$\begin{array}{rl} \text{maximizar} & y \\ \text{sujeto a:} & y(1,1,1) + (s_1, s_2, s_3) = (2,1,1), \\ & \sqrt{s_2^2 + s_3^2} \leq s_1, \text{ o bien } \mathbf{s} = (s_1, s_2, s_3) \text{ pertenece a un cono de} \\ & \qquad \text{segundo orden.} \end{array}$$

[1] Véase la página 8.

Ejemplo 2 El dual del problema de maximización cuadrática binaria.

Si se considera la relajación del problema de maximización cuadrática binaria correspondiente al ejemplo citado, 2, obtendremos su problema dual que es: [2]

$$\begin{array}{rl} \text{minimizar} & \sum_{j=1}^{n+1} y_j \\ \text{sujeto a:} & \sum_{j=1}^{n+1} y_j \mathbf{I}_j - \mathbf{S} = \begin{bmatrix} \mathbf{Q} & \mathbf{c} \\ \mathbf{c}^t & 0 \end{bmatrix}, \mathbf{S} \succeq 0. \end{array}$$

Ejemplo 3 El problema dual para el problema de localización de sensores en red.

Si se considera la relajación de la programación semidefinida para el caso del problema de localización de sensores en red, 3, se obtiene como problema dual: [3]

$$\begin{array}{rl} \text{maximizar} & \sum_{(i,j)\in N_e} y_{ij} \\ \text{sujeto a:} & \sum_{(i,j)\in N_e} y_{ij}(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^t + \mathbf{S} = \mathbf{0},\ \mathbf{S} \succeq 0. \end{array}$$

donde y_{ij} representa una tensión o una fuerza en la arista (i,j). Por supuesto $y_{ij} = 0$ $\forall (i,j) \in N_e$ constituye una solución factible para el dual. En estos casos, la maximización del dual resulta de gran ayuda para alcanzar el objetivo fundamental de encontrar fuerzas internas no triviales en diseño de estructuras y redes.

Ejemplo 4 Localización euclídea de instalaciones.

Este es un ejemplo que, como muchos otros problemas de optimización, se puede plantear directamente en su forma dual.

Este problema consiste en determinar la ubicación de instalaciones que surten a n clientes situados en el espacio euclídeo, cuya respectivas ubicaciones se conocen y se denotan por $a_j \in E^d$, $j = 1, \ldots, n$. La ubicación de la instalación debería minimizar la suma de las distancias euclídeas desde dicha instalación a cada uno de los clientes.

Si se representa por $\mathbf{f} \in E^d$ al vector que proporciona la localización de las instalaciones, el problema es:

$$\text{minimizar} \sum_{j=1}^{n} |\mathbf{f} - \mathbf{a}_j|.$$

Si bien, se puede reformular de la siguiente manera:

$$\begin{array}{rll} \text{minimizar} & \sum_{j=1}^{n} \delta_j & \\ \text{sujeto a:} & \mathbf{s}_j + \mathbf{f} = \mathbf{a}_j, & \forall j = 1, \ldots, n, \\ & |\mathbf{s}_j| \leq \delta_j, & \forall j = 1, \ldots, n. \end{array}$$

[2] Véase la página 9.
[3] Véase la página 9.

Este es un problema de formulación cónica en su forma dual (DLC). Se puede ver más claramente si usamos un caso particular en el que $d = 2$ y $n = 3$ y, si llamamos:

$$\mathbf{A}^t = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix} \in E^{9\,\mathrm{x}\,5}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \in E^5, \quad \mathbf{c} = \begin{bmatrix} 0 \\ \mathbf{a}_1 \\ 0 \\ \mathbf{a}_2 \\ 0 \\ \mathbf{a}_3 \end{bmatrix} \in E^9,$$

y el vector de variables es:

$$\mathbf{y} = [\delta_1;\ \delta_2;\ \delta_3;\ \mathbf{f}] \in E^5.$$

Entonces el problema de la localización de la instalación se plantea como:

$$\begin{aligned} \text{minimizar} \quad & \mathbf{y}^t\mathbf{b} \\ \text{sujeto a:} \quad & \mathbf{y}^t\mathbf{A} + \mathbf{s}^t = \mathbf{c}^t, \quad \mathbf{s} \in K; \end{aligned}$$

donde K es el producto de tres conos de segundo orden, cada uno de ellos de dimensión 3. De manera más precisa, se puede decir que los primeros tres elementos de $\mathbf{s} \in E^9$ se encuentran en un cono de segundo orden de dimensión tres, y lo mismo ocurre con los segundos tres elementos de $\mathbf{s}$ y con los terceros tres elementos.

Para referirnos a lo anteriormente expuesto se va a denotar el producto de tres conos (que podrían ser mixtos), K_1, K_2 y K_3, por $K_1 \oplus K_2 \oplus K_3$, de manera que cuando $\mathbf{X} \in K_1 \oplus K_2 \oplus K_3$ significa que $\mathbf{X}$ se divide en tres componentes tales que:

$$\mathbf{X} = (\mathbf{X}_1;\ \mathbf{X}_2;\ \mathbf{X}_3), \text{ donde } \mathbf{X}_1 \in K_1,\ \mathbf{X}_2 \in K_2, \text{ y } \mathbf{X}_3 \in K_3.$$

El dual del problema de localización de las instalaciones en DLC se formularía de la siguiente manera:

$$\begin{aligned} \text{minimizar} \quad & \mathbf{c}^t\mathbf{x} \\ \text{sujeto a:} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \in K^*; \end{aligned}$$

donde

$$K^* = (K_1 \oplus K_2 \oplus K_3)^* = K_1^* \oplus K_2^* \oplus K_3^*,$$

es decir, en este caso los tres primeros elementos de $\mathbf{x} \in E^9$ se encuentran en un cono de segundo orden y dimensión tres y lo mismo ocurre con los segundos y terceros tres elementos de $\mathbf{x}$.

Este problema dual se puede simplificar si se consideran las restricciones de igualdad. En este caso, el planteamiento del problema sería:

$$\begin{array}{rll} \text{maximizar} & \sum_{j=1}^{3} \mathbf{a}_j^t \mathbf{x}_j & \\ \text{sujeto a:} & \sum_{j=1}^{3} \mathbf{x}_j & = \mathbf{0} \in E^2, \\ & |\mathbf{x}_j| & \leq 1, \, \forall j = 1, 2, 3. \end{array}$$

Ejemplo 5 Restricciones cuadráticas.

Mediante el empleo del concepto de *Complementos de Schur* se pueden transformar restricciones cuadráticas en su forma lineal semidefinida.

El complemento de Schur se define de la siguiente manera:

Sea $\mathbf{A}$ una matriz (simétrica) semidefinida positiva de dimensión m, $\mathbf{C}$ una matriz simétrica de dimensión n y $\mathbf{B}$ una matriz de dimensión $m \times n$. Se llama complemento de Schur de $\mathbf{A}$ en la matriz $\mathbf{Z}$ a:

$$\mathbf{S} = \mathbf{C} - \mathbf{B}^t \mathbf{A}^{-1} \mathbf{B}$$

donde:

$$\mathbf{Z} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^t & \mathbf{C} \end{bmatrix}.$$

Se verifica que $\mathbf{Z}$ es semidefinida positiva cuando y solo cuando $\mathbf{S}$ es semidefinida positiva.

En este ejemplo se va a considerar una restricción cuadrática genérica de la forma:

$$\mathbf{y}^t \mathbf{B}^t \mathbf{B} \mathbf{y} - \mathbf{c}^t \mathbf{y} - d \leq 0. \tag{2.4}$$

Que se puede reformular de manera equivalente mediante:

$$\begin{bmatrix} \mathbf{I} & \mathbf{B}\mathbf{y} \\ \mathbf{y}^t \mathbf{B}^t & \mathbf{c}^t \mathbf{y} + d \end{bmatrix} \succeq \mathbf{0} \tag{2.5}$$

El complento de Schur de la matriz $\begin{bmatrix} \mathbf{I} & \mathbf{B}\mathbf{y} \\ \mathbf{y}^t \mathbf{B}^t & \mathbf{c}^t \mathbf{y} + d \end{bmatrix}$ respecto a $\mathbf{I}$ es $-(\mathbf{y}^t \mathbf{B}^t \mathbf{B} \mathbf{y} - \mathbf{c}^t \mathbf{y} - d)$, donde $\mathbf{y}$ no figura de manera cuadrática, sino de manera afín. De hecho la expresión 2.5 se puede reformular como:

$$\mathbf{P}(\mathbf{y}) = \mathbf{P}_0 + y_1 \mathbf{P}_1 + y_2 \mathbf{P}_2 + \cdots + y_n \mathbf{P}_n \succeq \mathbf{0}, \tag{2.6}$$

siendo

$$\mathbf{P}_0 = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & d \end{bmatrix}, \quad \mathbf{P}_i = \begin{bmatrix} \mathbf{0} & \mathbf{b}_i \\ \mathbf{b}_i^t & c_i \end{bmatrix}, \quad \text{para } i = 1, 2, \ldots, n$$

donde $\mathbf{b}_i$ es la i-ésima columna de $\mathbf{B}$ y c_i es la i-ésima componente de $\mathbf{c}$.

En la forma dual de los problemas de programación semidefinida, la restricción aparece como figura en la expresión 2.6.

Existe una manera más eficiente de formular la desigualdad 2.4 mediante el uso de programación semidefinida y conos de segundo orden. Esta formulación reduce la dimensión del cono semifinido. Para ello se introducen primero la variable de holgura $\mathbf{s}$ y s_0 mediante restricciones lineales:

$$\mathbf{By} - \mathbf{s} = \mathbf{0}.$$

En cuyo caso se tiene $|\mathbf{s}| \leq s_0$ o para el caso del cono de segundo orden $(s_0; \mathbf{s})$ y además:

$$\begin{bmatrix} 1 & s_0 \\ s_0 & \mathbf{c}^t\mathbf{y} + d \end{bmatrix} \succeq \mathbf{0}.$$

De nuevo, la restricción matricial aparece en la forma dual de un cono semidefinido pero con dimensión fija 2.

La relación entre los valores óptimos de las programaciones dual y primal se manifiesta en la dualidad. La forma débil de esta relación se describe en el siguiente lema:

Lema 2.4 *Dualidad débil en PLC*

Sea $\mathbf{X}$ *factible para (PPLC) y* $(\mathbf{y}, \mathbf{S})$ *factible para (PDLC), entonces:*

$$\mathbf{C} \bullet \mathbf{X} \geq \mathbf{y}^t\mathbf{b}.$$

La dualidad fuerte en (PLC), como en otros casos de dualidad es débil a menos que se verifiquen otras condiciones. Por ejemplo, la brecha de dualidad, que es la diferencia entre las soluciones primal y dual, puede no ser cero, como puede verse en el siguiente ejemplo:

Ejemplo 6 En este ejemplo de programación semidefinida aparece una brecha de dualidad:

$$\mathbf{C} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{A}_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{A}_2 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

y

$$\mathbf{b} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}.$$

El valor objetivo primal mínimo es 0, y se obtiene cuando:

$$\mathbf{X} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

y el valor objetivo dual máximo es -2 obtenido cuando $\mathbf{y} = [0, -1]$. Por tanto la brecha de dualidad es 2.

Sin embargo, cuando se dan ciertas condiciones, no aparecería esta brecha de dualidad, es decir, sería nula. Una de las condiciones está relacionada con el hecho de si existe una solución factible interior a la región factible primal $\mathcal{F}_p$ o en la región factible dual.

Hay que tener en cuenta que se considera que $\mathcal{F}_p$ tiene una solución factible interior si y solo si $\mathring{\mathcal{F}}_p = \{\mathbf{X} : \mathcal{A}\mathbf{X} = \mathbf{b}, \mathbf{X} \in \mathring{K}\} \neq \emptyset$ y $\mathcal{F}_d$ tiene solución factible interior si y solo si: $\mathring{\mathcal{F}}_d = \{(\mathbf{y}, \mathbf{S}) : \mathbf{y}^t\mathcal{A} + \mathbf{S} = \mathbf{C}, \mathbf{S} \in \mathring{K}^*\} \neq \emptyset$.

Resulta conveniente citar ahora una versión del teorema de dualidad fuerte.

Lema 2.5 *Dualidad fuerte en PLC*

I. a) *Dado* $(PPLC)$ *no factible y* $(PDLC)$ *factible y con interior, entonces* $(PDLC)$ *es ilimitado.*

   b) *Dado* $(PDLC)$ *no factible y* $(PPLC)$ *factible y con interior, entonces* $(PPLC)$ *es ilimitado.*

II. *Sean* (PLC) *y* (DLC) *factibles, y uno de ellos con interior, entonces no existe brecha de dualidad entre ellos.*

III. *Sean* (PLC) *y* (DLC) *factibles y con interior, entonces ambos tienen soluciones óptimas sin brecha de dualidad entre ellos.*

La mayoría de ejemplos en los que el teorema de dualidad fuerte falla son de poca importancia y una pequeña perturbación provocaría el fallo.

Por tanto, se va a considerar de aquí en adelante, salvo indicación contraria, que tanto el $(PPLC)$ como el $(PDLC)$ tienen interior cuando son factibles, y por tanto un par de soluciones óptimas para cualquier problema primal y dual deben satisfacer las condiciones de optimalidad siguientes:

$$\begin{aligned} \mathbf{C} \bullet \mathbf{X} - \mathbf{y}^t\mathbf{b} &= 0 \\ \mathcal{A}\mathbf{X} &= \mathbf{b} \\ \mathbf{y}^t\mathcal{A} + \mathbf{S} &= \mathbf{C}^t \\ \mathbf{X} \in K, \quad \mathbf{S} &\in K^*, \end{aligned}$$

o

$$\begin{aligned} \mathbf{X} \bullet \mathbf{S} &= 0 \\ \mathcal{A}\mathbf{X} &= \mathbf{b} \\ \mathbf{y}^t\mathcal{A} + \mathbf{S} &= \mathbf{C}^t \\ \mathbf{X} \in K, \quad \mathbf{S} &\in K^*. \end{aligned}$$

Ejemplo 7 Diseño Robusto de Carteras.

El modelo de Markowitz de diseño de carteras es

$$\begin{aligned} \text{minimizar} \quad & \mathbf{x}^t\mathbf{\Sigma}\mathbf{x} \\ \text{sujeto a:} \quad & \mathbf{1}^t\mathbf{x} = 1,\ \mathbf{\Pi}^t\mathbf{x} \geq \pi, \end{aligned}$$

donde $\mathbf{\Sigma}$ es la matriz de varianzas y covarianzas, el vector $\mathbf{\Pi}$ es la tasa de rentabilidad esperada de un conjunto de inversiones y π es la tasa de rentabilidad deseada de la cartera. Este problema se puede reescribir de manera equivalente como un problema cónico mixto:

$$\begin{aligned} \text{minimizar} \quad & \mathbf{\Sigma} \bullet \mathbf{X} \\ \text{sujeto a:} \quad & \mathbf{1}^t\mathbf{x} = 1,\ \mathbf{\Pi}^t\mathbf{x} \geq \pi \\ & \mathbf{X} - \mathbf{x}\mathbf{x}^t \succeq \mathbf{0}. \end{aligned}$$

Supongamos ahora que $\mathbf{\Sigma}$ está incompleta y/o con valores inciertos, y se expresa por:

$$\mathbf{\Sigma}_0 + \sum_{i=1}^{m} y_i\mathbf{\Sigma}_i (\succeq \mathbf{0}),$$

para variables y_i. Entonces se quiere resolver el modelo robusto:

$$\begin{aligned} \text{minimizar} \quad & \left\{ \begin{array}{rl} \text{maximizar}_{\mathbf{y}} & (\mathbf{\Sigma}_0 + \sum_{i=1}^{m} y_i\mathbf{\Sigma}_i) \bullet \mathbf{X} \\ \text{sujeto a} & \mathbf{\Sigma}_0 + \sum_{i=1}^{m} y_i\mathbf{\Sigma}_i \succeq \mathbf{0} \end{array} \right\} \\ \text{sujeto a:} \quad & \mathbf{1}^t\mathbf{x} = 1,\ \mathbf{\Pi}^t\mathbf{x} \geq \pi \\ & \mathbf{X} - \mathbf{x}\mathbf{x}^t \succeq \mathbf{0}. \end{aligned}$$

Se trata de un problema SDP. Suponiendo que se verifica la dualidad fuerte se recurre al planteamiento mediante su dual:

$$\begin{aligned} \text{minimizar} \quad & \left\{ \begin{array}{rll} \text{minimizar}_{\mathbf{Y}} & \mathbf{\Sigma}_0 \bullet (\mathbf{Y} + \mathbf{X}) & \\ \text{sujeto a} & \mathbf{\Sigma}_i \bullet (\mathbf{Y} + \mathbf{X}) = 0, & \forall i = 1, \ldots, m, \\ & \mathbf{Y} \succeq \mathbf{0} & \end{array} \right\} \\ \text{sujeto a:} \quad & \mathbf{1}^t\mathbf{x} = 1,\ \mathbf{\Pi}^t\mathbf{x} \geq \pi \\ & \mathbf{X} - \mathbf{x}\mathbf{x}^t \succeq \mathbf{0}. \end{aligned}$$

Si agrupamos conjuntamente ambos problemas en uno solo se puede formular:

$$\begin{aligned} \text{minimizar} \quad & \mathbf{\Sigma}_0 \bullet (\mathbf{Y} + \mathbf{X}) \\ \text{sujeto a} \quad & \mathbf{1}^t\mathbf{x} = 1, \quad \mathbf{\Pi}^t\mathbf{x} \geq \pi \\ & \mathbf{\Sigma}_i \bullet (\mathbf{Y} + \mathbf{X}) = 0, \quad \forall i = 1, \ldots, m, \\ & \mathbf{Y} \succeq \mathbf{0}, \quad \mathbf{X} - \mathbf{x}\mathbf{x}^t \succeq \mathbf{0}. \end{aligned}$$

En la siguiente tabla pueden verse las reglas que se derivan de la defición original y la equivalencia entre la formulación de problemas en la programación lineal cónica.

Matriz o vector de coeficientes del objetivo Lado derecho $\mathcal{A}$	Lado derecho Matriz o vector de coeficientes del objetivo $\mathcal{A}^t$
Maximización	Minimización
$\mathbf{x}_j \in K$	El bloque j ésimo de las variables de holgura $\in K^*$
$\mathbf{x}_j$ libre	El bloque j ésimo de las variables de holgura $= \mathbf{0}$
El bloque i ésimo de las variables de holgura $\in K$	$\mathbf{y}_i \in K^*$
El bloque i ésimo de las variables de holgura $= \mathbf{0}$	$\mathbf{y}_i$ libre

$\mathbf{x}_j$ es el bloque j ésimo de variables
$\mathbf{y}_i$ es el vector dual asociado con el bloque i ésimo de restricciones.

Tabla 2.1: Reglas de equivalencia entre elementos de problemas primal y dual en programación lineal cónica.

En la siguiente tabla podemos ver como pueden relacionarse los elementos que aparecen en los problemas primal y dual para, formulado uno de los dos problemas, poder formular el otro:

Elemento del problema	Problema Primal (o Dual)	Problema Dual (o Primal)	
Función objetivo	Maximizar	Minimizar	
Restricciones	$\leq$	$\geq$	Variables
	$\geq$	$\leq$	
	$=$	Irrestringido	
Variables	$\geq$	$\geq$	Restricciones
	$\leq$	$\leq$	
	Irrestringido	$=$	

Tabla 2.2: Reglas de equivalencia entre elementos de problemas primal y dual en programación lineal.

2.3. Complementariedad

La complementariedad en programación lineal es una propiedad que se define en el espacio cuando se cumple que:

Si $\mathbf{x} \geq \mathbf{0}$ y $\mathbf{s} \geq \mathbf{0}$, cuando $0 = \mathbf{x} \bullet \mathbf{s} = \mathbf{x}^t\mathbf{s} = \sum_{j=1}^{n} x_j s_j \Longrightarrow x_j s_j = 0$, $\forall j = 1, \ldots, n$.

Por tanto, además de ser factibles y óptimas los pares de soluciones en programación lineal satisfacen la propiedad de complementariedad anteriormente descrita.

Capítulo 3

Algoritmos de resolución de problemas en programación lineal cónica

3.1. Complementariedad en programación semidefinida

Una vez citada la propiedad de complementariedad en el apartado 2.3 anterior, nos disponemos a ver cómo actúa la complementariedad en la programación semidefinida.

Consideremos el cono semidefinido S_+^n. Puesto que $\mathbf{X} \succeq \mathbf{0}$ y $\mathbf{S} \succeq \mathbf{0}$, $0 = \mathbf{X} \bullet \mathbf{S}$ implican que $\mathbf{XS} = \mathbf{0}$, es decir, la matriz producto $\mathbf{XS}$ es nula, refiriendonos al producto ordinario de matrices y, por tanto, cada columna (o fila) de $\mathbf{X}$ es ortogonal a cada columna (o fila) de $\mathbf{S}$.

Propiedades sobre el rango en soluciones de problemas SDP

Veamos determinadas proposiciones referidas al rango en soluciones de problemas SDP.

Proposición 3.1 Sea $\mathbf{X}^*$ e $(\mathbf{y}^*, \mathbf{S}^*)$ cualquier par solución óptima del problema SDP con una brecha de dualidad nula. Por tanto la complementariedad de $\mathbf{X}^*$ y $\mathbf{S}^*$ implica

$$\text{rango}(\mathbf{X}^*) + \text{rango}(\mathbf{S}^*) \leq n.$$

Más aún, son equivalentes las condiciones siguientes:

a. Existe un dual óptimo $\mathbf{S}^*$ tal que $\text{rango}(\mathbf{S}^*) \geq d$,

b. El rango de cualquier problema primal óptimo $\mathbf{X}^*$ está acotado superiormente por $n - d$, donde $0 \leq d \leq n$ es entero.

En algunos problemas SDP, podría interesar encontrar una solución óptima de rango mínimo, mientras que en el algoritmo de punto interior para SDP que veremos en la sección 3.2, genera una solución cuyo rango es máximo para el primal y el dual respectivamente. Por tanto, resulta necesario para conseguir este objetivo un método de reducción de rango.

En el caso de la programación lineal en su forma estándar, se conoce que si existe una solución óptima factible, existe también una solución óptima básica $\mathbf{x}^*$ con a lo sumo m entradas positivas. Se estudiará a continuación si esta propiedad se da también para un problema de programación semidefinida.

Para el caso de la programación semidefinida, existe una propiedad similar:

Proposición 3.2 Si existe una solución óptima del problema SDP, entonces existe una solución óptima del problema SDP cuyo rango r satisface $\frac{r(r+1)}{2} \leq m$.

Métodos de reducción de rango

Método de reducción del espacio vacío

Sea $\mathbf{X}^*$ una solución óptima para el problema SDP con rango r. Si $r(r+1)/2 > m$, se factoriza $\mathbf{X}^*$ ortonormalmente de la forma:

$$\mathbf{X}^* = (\mathbf{V}^*)^t\mathbf{V}^*, \quad \mathbf{V}^* \in E^{r\times n}.$$

Se considera ahora el problema SPD relacionado:

$$\begin{aligned} \text{minimizar} \quad & \mathbf{V}^*\mathbf{C}(\mathbf{V}^*)^t \bullet \mathbf{U} \\ \text{sujeto a:} \quad & \mathbf{V}^*\mathbf{A}_i(\mathbf{V}^*)^t \bullet \mathbf{U} = b_i,\ i = 1,\dots,m \\ & \mathbf{U} \in S_+^r. \end{aligned} \tag{3.1}$$

Para cualquier solución factible de 3.1 se puede construir una solución factible para el problema SDP original usando:

$$\mathbf{X}(\mathbf{U}) = (\mathbf{V}^*)^t\mathbf{U}\mathbf{V}^* \quad \text{y} \quad \mathbf{C} \bullet \mathbf{X}(\mathbf{U}) = \mathbf{V}^*\mathbf{C}(\mathbf{V}^*)^t \bullet \mathbf{U}.$$

Por tanto, el valor mínimo de 3.1 es también z^*, y en particular $\mathbf{U} = \mathbf{I}$ es un minimizador de la función objetivo de la expresión 3.1, ya que:

$$\mathbf{V}^*\mathbf{C}(\mathbf{V}^*)^t \bullet \mathbf{I} = \mathbf{C} \bullet (\mathbf{V}^*)^t\mathbf{V}^* = \mathbf{C} \bullet \mathbf{X}^* = z^*.$$

Además, se puede ver que cualquier solución factible $\mathbf{U}$ de 3.1 es su propio minimizador, por tanto $\mathbf{X(U)}$ es un minimizador del problema SDP original.

Considérese el sistema de equaciones lineales homogéneas:

$$\mathbf{V}^*\mathbf{A}_i(\mathbf{V}^*)^t \bullet \mathbf{W} = 0,\ i = 1, \ldots, m.$$

donde $\mathbf{W} \in S^r$, una matriz de orden r no necesariamente definida positiva. Este sistema tiene m ecuaciones y $r(r+1)/2$ variables reales. Así pues, siempre que $r(r+1)/2 > m$, se podrá encontrar una matriz simétrica $\mathbf{W} \neq \mathbf{0}$ para satisfacer todas las m ecuaciones. Sin perdida de generalidad, sea $\mathbf{W}$ indefinida o semidefinida negativa (en este último caso, se toma $-\mathbf{W}$ en vez de $\mathbf{W}$), es decir, $\mathbf{W}$ tiene al menos un autovalor negativo. Entonces, considérese:

$$\mathbf{U}(\alpha) = \mathbf{I} + \alpha\mathbf{W}.$$

Tomando un α^* suficientemente grande de forma que $\mathbf{U}(\alpha^*) \succeq \mathbf{0}$ y que tenga al menos un autovalor nulo, es decir, que rango$(\mathbf{U}(\alpha^*)) < r$, se tiene que:

$$\mathbf{V}^*\mathbf{A}_i(\mathbf{V}^*)^t \bullet \mathbf{U}(\alpha^*) = \mathbf{V}^*\mathbf{A}_i(\mathbf{V}^*)^t \bullet (\mathbf{I} + \alpha^*\mathbf{W}) = \mathbf{V}^*\mathbf{A}_i(\mathbf{V}^*)^t \bullet \mathbf{I} = b_i,\ i = 1, \ldots, m.$$

Es decir que $\mathbf{U}(\alpha^*)$ es factible y óptimo para el problema 3.1. Entonces $\mathbf{X}(\mathbf{U}(\alpha^*))$ es un nuevo minimizador para el problema SPD original, y su rango es estrictamente menor que r. Este proceso puede repetirse hasta que el sistema de ecuaciones lineales homogéneas tenga únicamente como solución la nula, que se da necesariamente cuando $r(r+1)/2 \leq m$. Este procedimiento determinísta de reducción de rango de la solución se llama método de reducción del espacio vacío.

Se puede ver una aplicación de la Proposición 3.2 considerando una minimización genereal cuadrática restringida a la esfera.

$$\begin{aligned} z^* \equiv \quad & \text{minimizar} \quad \mathbf{x}^t\mathbf{Q}\mathbf{x} + 2\mathbf{c}^t\mathbf{x} \\ & \text{sujeto a:} \quad |\mathbf{x}|^2 = 1,\ \mathbf{x} \in E^n. \end{aligned}$$

Este problema tiene una relajación SDP:

$$\begin{aligned} z^{SDP} \equiv \quad & \text{maximizar} \quad \begin{bmatrix} \mathbf{Q} & \mathbf{c} \\ \mathbf{c}^t & 0 \end{bmatrix} \bullet \mathbf{Y} \\ & \text{sujeto a:} \quad \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0}^t & 0 \end{bmatrix} \bullet \mathbf{Y} = 1, \\ & \qquad\qquad \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0}^t & 1 \end{bmatrix} \bullet \mathbf{Y} = 1, \\ & \qquad\qquad \mathbf{Y} \in S_+^{n+1}. \end{aligned}$$

Nótese que tanto la relajación como su dual tienen interior de tal manera que se verifica el teorema de dualidad fuerte, y debe tener una solución óptima en el problema SDP de

rango 1 dado que $m = 2$. Pero una solución óptima de SDP de rango 1 debe ser óptima para el problema de minimización genereal cuadrática restringida a la esfera. Por tanto, se tiene que $z^* = z^{SDP}$.

Método de reducción de rango de la proyeción Gaussiana

Existe también un procedimiento aleatorio para conseguir una solución del problema SDP aproximada con el menor rango d que se quiera. De nuevo, $\mathbf{X}^*$ es una solución óptima del problema SDP con rango $r > d$. Se factoriza $\mathbf{X}^*$ como:

$$\mathbf{X}^* = (\mathbf{V}^*)^t\mathbf{V}^*, \quad \mathbf{V}^* \in E^{r\times n}.$$

A continuación, se generan las variables aleatorias Gaussianas independientes e idénticamente distribuidas(iid) ξ_i^j con media 0 y varianza $1/d$, $i = 1,\dots,r$; $j = 1,\dots,d$. Se forman los vectores aleatorios $\boldsymbol{\xi}^j = (\xi_1^j;\dots;\xi_r^j)$, $j = 1,\dots,d$. Finalmente obtenemos

$$\hat{\mathbf{X}} = (\mathbf{V}^*)^t \left[\sum_{j=1}^{d} \boldsymbol{\xi}^j(\boldsymbol{\xi}^j)^t\right] \mathbf{V}^*.$$

Fijemonos que el rango de $\hat{\mathbf{X}}$ es d y que

$$\mathrm{E}(\hat{\mathbf{X}}) = (\mathbf{V}^*)^t\mathrm{E}\left[\sum_{j=1}^{d} \boldsymbol{\xi}^j(\boldsymbol{\xi}^j)^t\right] \mathbf{V}^* = (\mathbf{V}^*)^t\mathbf{I}\mathbf{V}^* = \mathbf{X}^*.$$

Podría verse además que $\hat{\mathbf{X}}$ sería en muchos casos una buena aproximación de rango d a la solución SDP.

Método de reducción de rango mediante variable aleatoria binaria

Se quiere producir un vector $\mathbf{x}$ cuyas entradas sean 1 o -1. Un procedimiento para ello es el siguiente:

Sea $\mathbf{X}^*$ cualquir solución óptima del problema SDP, que factorizamos de la siguiente manera:

$$\mathbf{X}^* = (\mathbf{V}^*)^t\mathbf{V}^*, \quad \mathbf{V}^* \in E^{n\times n}.$$

Se genera entonces un vector $\boldsymbol{\xi}$, aleatorio de dimensión n, en el que sus componentes son variables aleatorias gaussianas independientes e idénticamente distribuidas(iid) con media 0 y varianza 1. Así, tenemos que:

$$\hat{\mathbf{x}} = \mathrm{signo}((\mathbf{V}^*)^t\boldsymbol{\xi})$$

donde

$$\text{signo}(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ -1 & \text{en caso contrario} \end{cases}$$

Fue probado por Sheppard, 1900 [2]

$$\mathrm{E}[\hat{x}_i \hat{x}_j] = \frac{2}{\pi} \operatorname{arc\,sen}(\mathbf{X}^*_{ij}),\ i, j = 1, 2, \ldots, n.$$

Se puede comprobar que $\hat{\mathbf{x}}$ es una buena aproximación del problema cuadrático binario original. Considérese el problema homogéneo de maximización cuadrático binario:

$$\begin{aligned} z^* = \quad & \text{maximizar} \quad \mathbf{x}^t \mathbf{Q} \mathbf{x} \\ & \text{sujeto a:} \quad x_j = \{1, -1\},\ \forall j = 1, \ldots, n, \end{aligned}$$

donde se asume que $\mathbf{Q}$ es semidefinida positiva. Por tanto, la relajación del problema SDP sería:

$$\begin{aligned} z^{SDP} = \quad & \text{maximizar} \quad \mathbf{Q} \bullet \mathbf{X} \\ & \text{sujeto a:} \quad \mathbf{I}_j \bullet \mathbf{X} = 1,\ \forall j = 1, \ldots, n \\ & \qquad\qquad\ \ \mathbf{X} \in S^n_+, \end{aligned}$$

y sea $\mathbf{X}^*$ una solución óptima cualquiera, de la cual se genera un vector binario $\hat{\mathbf{x}}$ aleatorio. A continuación evaluando el valor objetivo esperado:

$$\mathrm{E}(\hat{\mathbf{x}}^t \mathbf{Q} \hat{\mathbf{x}}) = \mathrm{E}(\mathbf{Q} \bullet \hat{\mathbf{x}} \hat{\mathbf{x}}^t) = \mathbf{Q} \bullet \mathbf{E}(\hat{\mathbf{x}} \hat{\mathbf{x}}^t) = \mathbf{Q} \bullet \frac{2}{\pi} \operatorname{arc\,sen}[\mathbf{X}^*] = \frac{2}{\pi}(\mathbf{Q} \bullet \operatorname{arc\,sen}[\mathbf{X}^*]),$$

donde $\operatorname{arc\,sen}[\mathbf{X}^*] \in S^n$, cuya componente (i, j) es $\operatorname{arc\,sen}(\mathbf{X}^*_{ij})$. Además se puede comprobar que:

$$\operatorname{arc\,sen}[\mathbf{X}^*] - \mathbf{X}^* \succeq \mathbf{0},$$

por tanto, como $\mathbf{Q} \succeq \mathbf{0}$, se tiene:

$$\mathbf{Q} \bullet \operatorname{arc\,sen}[\mathbf{X}^*] \geq \mathbf{Q} \bullet \mathbf{X}^* = z^{SDP} \geq z^*,$$

es decir, el valor objetivo esperado de $\hat{\mathbf{x}}$ no es inferior al producto del valor máximo del problema cuadrático binario por $\frac{2}{\pi}$.

Este método de reducción de rango se puede extender al caso de optimización cuadrática con restricciones acotadas tales como $x_j^2 \leq 1$.

Método de reducción de rango mediante guía de objetivo

A menudo, puede ayudar a encontrar una solución SDP de bajo rango añadiendo ciertos términos en la función objetivo. Considérese el problema de localización de redes de sensores en cadena representado en la siguiente figura:

Figura 3.1: Problema de localización de redes de sensores en cadena

donde las aristas $(0,1),(1,2),(2,3)$ tienen longitud 1 para cuatro puntos denotados por $\{0,1,2,3\}$. Fijando la posición $\mathbf{x}_0 = \{0\}$ como origen, queremos formular un problema de localización de redes de sensores para encontrar las restantes tres posiciones resolviendo:

$$\begin{aligned} |\mathbf{x}_1|^2 &= 1 \\ |\mathbf{x}_2 - \mathbf{x}_1|^2 &= 1 \\ |\mathbf{x}_3 - \mathbf{x}_2|^2 &= 1. \end{aligned}$$

Si aplicamos simplemente la relajación SDP para resolver este problema de factibilidad, la solución SDP debería tener rango mayor que 1.

Sin embargo, si añadimos una función objetivo a las condiciones anteriores se da lugar a un problema que se enuncia como:

$$\begin{aligned} \text{maximizar} \quad & |\mathbf{x}_3|^2 \\ \text{sujeto a:} \quad & |\mathbf{x}_1|^2 = 1 \\ & |\mathbf{x}_2 - \mathbf{x}_1|^2 = 1 \\ & |\mathbf{x}_3 - \mathbf{x}_2|^2 = 1, \end{aligned}$$

y aplicando la relajación SDP para resolver esta nueva formulación la solución SDP tendría rango 1.

Esto quiere decir que seríamos capaces de encontrar las posiciones a lo largo de una simple línea en dimensión 1. El objetivo añadido crearía fuerzas de tensión no nulas en las aristas que deben considerarse para equilibrar sus fuerzas en cada punto de unión. Se usa el término integridad de tensión para el objetivo. En este caso las fuerzas de tensión en las aristas son las variables duales de la relajación SDP.

3.2. Métodos de punto interior

Como la (PLC) no deja de ser un problema de minimización convexa, se pueden emplear muchos algoritmos para la resolución de estos problemas. Sin embargo destacamos el algoritmo de punto interior.

Para desarrollar un algoritmo de punto interior eficiente, se debe disponer de una función barrera adecuada. Existe una teoría general para la selección de funciones barreras en la programación lineal cónica dependiendo del cono convexo de que se trate.

Consideramos el siguiente problema a resolver:

$$\begin{aligned} \text{minimizar} \quad & f(\mathbf{x}) \\ \text{sujeto a:} \quad & \mathbf{x} \in \mathbf{\Omega} \end{aligned} \tag{3.2}$$

donde $f \in E^n$ es una función continua y $\mathbf{\Omega} \in E^n$ el conjunto de restricciones.

Los métodos de punto interior consisten en transformar la función objetivo complementándola con una función de penalización o una función barrera. Veamos cada uno de estos métodos:

Método de penalización

La idea de este método es transformar el problema 3.2 en uno sin restricciones de la forma:

$$\text{minimizar} \quad q(c, \mathbf{x}) = f(\mathbf{x}) + cP(\mathbf{x}) \tag{3.3}$$

donde c es una constante positiva y $P \in E^n$, conocida como función de penalización, es una función que satisface:

I. P es continua.

II. $P(\mathbf{x}) \geq 0$, $\forall \mathbf{x} \in E^n$.

III. $P(\mathbf{x}) = 0 \iff \mathbf{x} \in \mathbf{\Omega}$.

Supongamos que $\mathbf{\Omega}$ se define mediante una serie de restricciones de igualdad y desigualdad:

$$\mathbf{\Omega} = \{\mathbf{x} : h_i(\mathbf{x}) = 0,\ i = 1, 2, \ldots, m, \quad g_j(\mathbf{x}) \geq 0,\ j = 1, 2, \ldots, p\}.$$

La siguiente tabla recoge algunos ejemplos y características de funciones de penalización usadas:

Tipo de función de penalización	Función de penalización	Características
Penalización cuadrática (agrupada)	$P(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{m}(h_i(\mathbf{x}))^2 + \frac{1}{2}\sum_{j=1}^{p}(\bar{g_j}(\mathbf{x}))^2$ siendo $\bar{g_j}(\mathbf{x}) = \text{mín}[0, g_j(\mathbf{x})]$, $j = 1, 2, \ldots, p$.	▪ En la región factible $P(\mathbf{x}) = 0$. ▪ Cuando no se cumplen las restricciones, es decir, cuando g_j son negativos es cuando P toma valores no nulos. ▪ Se necesita que c tienda a infinito para encontrar una solución factible. ▪ El crecimiento de c provoca que se dificulte la convergencia para el problema sin restricciones.
Penalización de valor absoluto (agrupada)	$P(\mathbf{x}) = \sum_{i=1}^{m}\lvert h_i(\mathbf{x})\rvert + \sum_{j=1}^{p}(-\bar{g_j}(\mathbf{x}))$ siendo $\bar{g_j}(\mathbf{x}) = \text{mín}[0, g_j(\mathbf{x})]$, $j = 1, 2, \ldots, p$.	▪ c necesita crecer hasta infinito para encontrar la solución factible.
Penalización Lagrangiana (Penalización precisa)	$P(\mathbf{x}) = \sum_{i=1}^{m} w_i^h h_i(\mathbf{x}) + \sum_{j=1}^{p} w_j^g g_j(\mathbf{x})$ considerando los valores w_i^h y w_j^g como pesos de penalización que son: ▪ w_i^h positivos o negativos ▪ $w_j^g \leq 0$ cuando $g_j(\mathbf{x}) \leq 0$ ▪ $w_j^g = 0$ cuando $g_j(\mathbf{x}) > 0$	▪ Los pesos de penalización w_i^h y w_j^g son los negativos del multiplicador de Lagrange o también la solución óptima del dual del problema con restricciones original.

Tabla 3.1: Diferentes tipos de funciones de penalización y sus características

La siguiente gráfica muestra el caso de la función de penalización cuadrática $cP(\mathbf{x})$ para distintos valores de c.

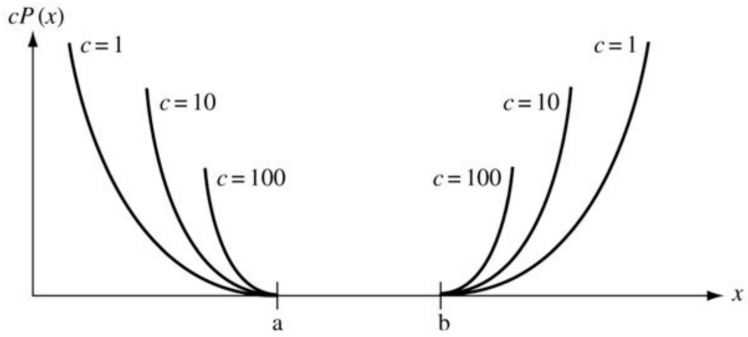

Figura 3.2: Función de penalización cuadrática

Se trata del caso de penalización cuadrática en dimensión 1 con $g_1(x) = b - x$, $g_2(x) = x - a$. En el caso de valores absolutos se tiene:

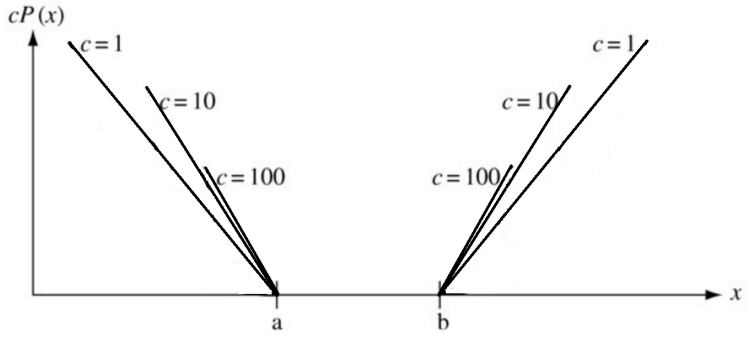

Figura 3.3: Función de penalización de valores absolutos

Para valores grandes de c el punto que minimiza el problema 3.3 se encuentra en la región en la que P es pequeño. Por tanto, al ir aumentando el valor de c es de esperar que los puntos de la solución correspondiente vayan aproximándose a la región factible $\mathbf{\Omega}$ y dado que está cerca, minimizará f.

Por tanto, dado que $c \longrightarrow \infty$ los puntos solución del problema de penalización convergerán a una solución del problema con restricciones.

Descripción de los distintos procesos para el método barrera:

1. Método de la M-grande:

   a) Fijar un valor para $c = M$, con M grande y positivo (M-grande).

   b) Resolver el problema con penalización sin restricciones para ese valor de c.

2. Mediante una sucesión de valores para c:

   a) Se toma una sucesión de valores $\{c_k\}_{k=1,2,\ldots}$, de forma que los $c_k \longrightarrow \infty$, $c_k \geq 0$ y $c_{k+1} > c_k$.

   b) Resolver el problema con penalización sin restricciones para cada valor de $c = c_k,\ k = 1, 2, \ldots$ obteniendo su punto solución $\mathbf{x}_k$.

   c) Cualquier punto límite de la sucesión $\{\mathbf{x}_k\}$ de puntos solución citados en el apartado anterior, es una solución del problema original con restricciones 3.2.

Métodos barrera

Los métodos barrera se aplican al problema en el que la región factible $\mathbf{\Omega}$ es un conjuno robusto, es decir, que tiene un interior no vacío que es arbitrariamente cerrado a cualquier punto de $\mathbf{\Omega}$ (esto último quiere decir, de manera intuitiva, que es posible acceder a cualquier punto de la frontera aproximándose a él desde el interior). En la figura siguiente se puede apreciar cuándo un conjunto es robusto y cuándo no:

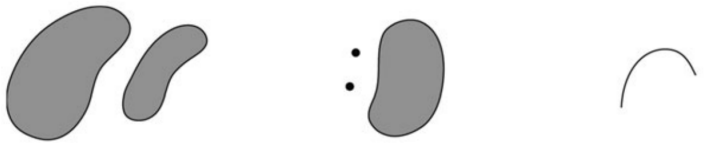

Figura 3.4: Conjuntos robustos y no robustos

Los conjuntos robustos aparecen de la mano con las restricciones de desigualdad, donde:

$$\mathbf{\Omega} = \{\mathbf{x} : g_j(\mathbf{x}) \geq 0,\ j = 1, 2, \ldots, p\}.$$

Los métodos barrera funcionan mediante el establecimiento de una barrera en la frontera de la región factible que evite en el procedimiento de búsqueda de la solución abandonar dicha región.

Una función barrera es una función B definida en el interior de $\mathbf{\Omega}$ que satisface:

I. B es continua.

II. $B(\mathbf{x})$ está acotada inferiormente.

III. $B(\mathbf{x}) \longrightarrow \infty$ conforme $\mathbf{x}$ se aproxima a la frontera de $\mathbf{\Omega}$.

Usamos la función barrera para considerar un problema sin restricciones derivado del problema original 3.2:

$$\begin{aligned} \text{minimizar} \quad & r(c, \mathbf{x}) = f(\mathbf{x}) + \tfrac{1}{c} B(\mathbf{x}) \\ \text{sujeto a:} \quad & \mathbf{x} \in \text{interior de } \boldsymbol{\Omega}, \end{aligned} \tag{3.4}$$

donde c es una constante positiva. Usualmente se utiliza el valor $\mu = \frac{1}{c}$ como parámetro, por lo que el problema se suele formular usando este parámetro:

$$\begin{aligned} \text{minimizar} \quad & f(\mathbf{x}) + \mu B(\mathbf{x}) \\ \text{sujeto a:} \quad & \mathbf{x} \in \text{interior de } \boldsymbol{\Omega} \end{aligned} \tag{3.5}$$

Cuando se formula el problema usando la constante c se toman valores grandes para ella. En cambio si se formula el problema usando μ este parámetro tomará valores pequeños.

De cualquier manera, en ambos casos, se trata de un problema con restricciones, lo que lo convierte en un problema más complicado de resolver que el original. Sin embargo, tiene la ventaja de que puede resolverse usando una técnica de búsqueda sin restricciones.

Para encontrar la solución se comienza en un punto interior inicial y desde ese punto se van encontrando los siguientes aplicando de manera iterada métodos de descenso disponibles para problemas sin restricciones.

Dado que el valor de la función objetivo se aproxima a infinito cerca de la frontera de $\boldsymbol{\Omega}$, la técnica de búsqueda permanecerá dentro del interior de $\boldsymbol{\Omega}$, no siendo necesario tener en cuenta de manera explícita las restricciones. Esto hace que aunque los problemas con función barrera formulados con c (problema 3.4) o con μ (problema 3.5) tengan formalmente restricciones, computacionalmente se hace innecesario comprobar que se verifican al aplicar el método, por lo que pueden tratarse como problemas sin restricciones.

La siguiente tabla recoge algunos ejemplos y características de funciones barrera:

Tipo de función barrera	Función barrera	Características
Función barrera recíproca	$B(\mathbf{x}) = \sum_{j=1}^{p} \frac{1}{g_j(\mathbf{x})}$	
Función barrera logarítmica negativa	$B(\mathbf{x}) = -\sum_{j=1}^{p} \log[g_j(\mathbf{x})]$	▪ Es la función barrera usada de manera generalizada en los métodos de punto interior. ▪ Está acotada inferiormente cuando $\mathbf{\Omega}$ está acotada.

Tabla 3.2: Diferentes tipos de funciones de penalización y sus características

Descripción de los métodos

1. Método de la M-grande:
   a) Fijar un valor para $c = M$ (o $\mu = \frac{1}{M}$), con M grande y positivo (M-grande).
   b) Resolver el problema con penalización sin restricciones para ese valor de c (o μ).
2. Mediante una sucesión de valores para c:
   a) Se toma una sucesión de valores $\{c_k\}_{k=1,2,\ldots}$, de forma que los $c_k \longrightarrow \infty$, $c_k \geq 0$ y $c_{k+1} > c_k$.
   b) Resolver el problema con penalización sin restricciones para cada valor de $c = c_k$, $k = 1, 2, \ldots$ obteniendo su punto solución $\mathbf{x}_k$
   c) Cualquier punto límite de la sucesión $\{\mathbf{x}_k\}$ de puntos solución generados por el método barrera, es una solución del problema original con restricciones 3.2

Veamos algunos casos de funciones barrera adecuadas a distintos tipos de conos convexos que se han citado anteriormente:

- El ortante no negativo de dimensión n, E_+^n:

$$B(\mathbf{x}) = -\sum_{j=1}^{n} \log(x_j).$$

- El cono semidefinido de dimensión n, S_+^n:

$$B(\mathbf{X}) = -\log(\det \mathbf{X}).$$

- El cono de segundo orden de dimensión $(n+1)$ $\{(u;\mathbf{x}) : u \geq |\mathbf{x}|\}$:

$$B(u;\mathbf{x}) = -\log(u^2 - |\mathbf{x}|^2).$$

A continuación nos ocuparemos de resolver problemas de (SDP). Al igual que en la programación lineal, consideramos los problemas (SDP) con la función barrera añadida en la función objetivo. En el caso primal se tiene:

$$\begin{array}{lll} (SDPB) & \text{minimizar} & \mathbf{C} \bullet \mathbf{X} - \mu \log\det(\mathbf{X}) \\ & \text{sujeto a:} & \mathbf{X} \in \mathring{\mathcal{F}}_p. \end{array}$$

En el caso dual:

$$\begin{array}{lll} (SDDB) & \text{maximizar} & \mathbf{y}^t\mathbf{b} + \mu \log\det(\mathbf{S}) \\ & \text{sujeto a:} & (\mathbf{y},\mathbf{S}) \in \mathring{\mathcal{F}}_d, \end{array}$$

donde $\mu > 0$ es el peso de la barrera. Para un valor de μ dado, los minimizadores de (SDPB) y (SDDB), denotados por $(\mathbf{X}(\mu), \mathbf{y}(\mu), \mathbf{S}(\mu))$, satisfacen las siguientes condiciones:

$$\begin{aligned} \mathbf{XS} &= \mu\mathbf{I} \\ \mathcal{A}\mathbf{X} &= \mathbf{b} \\ \mathcal{A}^t\mathbf{y} + \mathbf{S} &= \mathbf{C} \\ \mathbf{X} \succ \mathbf{0}, \quad & \mathbf{S} \succ \mathbf{0}. \end{aligned}$$

Dado que se verifica que

$$\mu = \frac{\text{traza}(\mathbf{XS})}{n} = \frac{\mathbf{X} \bullet \mathbf{S}}{n} = \frac{\mathbf{C} \bullet \mathbf{X} - \mathbf{y}^t\mathbf{b}}{n},$$

entonces μ es el promedio de la complementariedad o el promedio de la brecha de dualidad. Los minimizadores $(\mathbf{X}(\mu), \mathbf{y}(\mu), \mathbf{S}(\mu))$, forman el camino central de SDP para $\mu \in (0, \infty)$. Se sabe que cuando $\mu \longrightarrow 0$, $(\mathbf{X}(\mu), \mathbf{y}(\mu), \mathbf{S}(\mu))$ tiende a una solución óptima con rango maximal.

Podemos extender el uso de las funciones barrera de la programación lineal a la semidefinida positiva:

$$\psi_{n+\rho}(\mathbf{X}, \mathbf{S}) = (n+\rho)\log(\mathbf{X} \bullet \mathbf{S}) - \log(\det(\mathbf{X}) \cdot \det(\mathbf{S}))$$

donde $\rho \geq 0$.

Es conveniente señalar que si $\mathbf{X}$ y $\mathbf{S}$ son matrices diagonales, las definiciones anteriores se reducen al caso de la programación lineal.

Una vez que se tiene un punto interior factible $(\mathbf{X}, \mathbf{y}, \mathbf{S})$, podemos generar una nueva iteración $(\mathbf{X}^+, \mathbf{y}^+, \mathbf{S}^+)$ resolviendo para $(\mathbf{D_X}, \mathbf{d_y}, \mathbf{D_s})$ del sistema primal-dual de ecuaciones lineales

$$\begin{aligned} \mathbf{D}^{-1}\mathbf{D_X}\mathbf{D}^{-1} + \mathbf{D_S} &= \tfrac{n}{n+\rho}\mu\mathbf{X}^{-1} - \mathbf{S}, \\ \mathbf{A}_i \bullet \mathbf{D_X} &= 0, \forall i, \\ \textstyle\sum_i^m (\mathbf{d_y})_i \mathbf{A}_i + \mathbf{D_s} &= \mathbf{0}, \end{aligned} \tag{3.6}$$

donde $\mathbf{D}$ es la matriz:

$$\mathbf{D} = \mathbf{X}^{\frac{1}{2}}(\mathbf{X}^{\frac{1}{2}}\mathbf{S}\mathbf{X}^{\frac{1}{2}})^{-\frac{1}{2}}\mathbf{X}^{\frac{1}{2}}$$

y $\mu = \mathbf{X} \bullet \mathbf{S}/n$.

Entonces se asigna $\mathbf{X}^+ = \mathbf{X} + \alpha\mathbf{D_X}$, $\mathbf{y}^+ = \mathbf{y} + \alpha\mathbf{d_y}$, y $\mathbf{S}^+ = \mathbf{S} + \alpha\mathbf{D_S}$ con $\alpha > 0$. Más aún, se puede probar que:

$$\psi_{n+\rho}(\mathbf{X}^+, \mathbf{S}^+) - \psi_{n+\rho}(\mathbf{X}, \mathbf{S}) \leq -\delta$$

siendo $\delta > 0.2$, para un determinado valor concreto de $\alpha = \bar{\alpha}$,

En resumen el algoritmo se describe con los siguientes pasos:

1. Dado $(\mathbf{X}^0, \mathbf{y}^0, \mathbf{S}^0) \in \mathring{\mathcal{F}}$. Fijamos $\rho \geq \sqrt{n}$ y $k = 0$.
2. Fijemos $(\mathbf{X}, \mathbf{S}) = (\mathbf{X}^k, \mathbf{S}^k)$ y calculamos $(\mathbf{D_X}, \mathbf{d_y}, \mathbf{D_s})$ citados anteriormente en 3.6.
3. Sea $\mathbf{X}^{k+1} = \mathbf{X}^k + \bar{\alpha}\mathbf{D_X}$, $\mathbf{y}^{k+1} = \mathbf{y}^k + \bar{\alpha}\mathbf{d_y}$, $\mathbf{S}^{k+1} = \mathbf{S}^k + \bar{\alpha}\mathbf{D_S}$, donde:

$$\bar{\alpha} = \arg\min_{\alpha \geq 0} \psi_{n+\rho}(\mathbf{X}^k + \alpha\mathbf{D_X}, \mathbf{S}^k + \alpha\mathbf{D_S}).$$

4. Pasamos a la iteración siguiente, con el valor $k+1$. Si $\frac{\mathbf{X}^k \bullet \mathbf{S}^k}{\mathbf{X}^0 \bullet \mathbf{S}^0} \leq \epsilon$, se dan por finalizadas las iteraciones. En caso contrario se sigue iterando desde el paso 2.

3.3. Otros métodos

Algoritmo Auto Dual Homogéneo (HSD)

Considérese el problema de minimización:

$$\begin{aligned} (HSDCLP) \quad \text{minimizar} \quad & (n+1)\theta \\ \text{sujeto a} \quad & \mathcal{A}\mathbf{X} - \mathbf{b}\tau + \bar{\mathbf{b}}\theta = \mathbf{0}, \\ & -\mathcal{A}^t\mathbf{y} + \mathbf{C}\tau - \bar{\mathbf{C}}\theta = \mathbf{S} \in K^*, \\ & \mathbf{b}^t\mathbf{y} - \mathbf{C} \bullet \mathbf{X} + \bar{z}\theta = k \geq 0, \\ & -\bar{\mathbf{b}}^t\mathbf{y} + \bar{\mathbf{C}} \bullet \mathbf{X} - \bar{z}\tau = -(n+1), \\ & \mathbf{y} \text{ libre}, \mathbf{X} \in K, \tau \geq 0, \quad \theta \text{ libre} \end{aligned}$$

donde

$$\bar{\mathbf{b}} = \mathbf{b} - \mathcal{A}\mathbf{X}^0, \quad \bar{\mathbf{C}} = \mathbf{C} - \mathbf{S}^0, \quad \bar{z} = \mathbf{C} \bullet \mathbf{X}^0 + 1$$

Aquí $\mathbf{X}^0$ y $\mathbf{S}^0$ son cualquier pareja de puntos interiores en el interior de K y K^*, tal que forman puntos del camino central con $\mu = 1$. $\mathbf{X}^0$ y $\mathbf{S}^0$ no tienen por qué satisfacer otras restricciones de igualdad, por lo que se pueden reconocer facilmente. Por ejemplo:

- Para el caso del cono ortante no negativo de dimensión n, E_+^n se tiene $\mathbf{x}^0 = \mathbf{s}^0 = \mathbf{1}$.
- Para el cono semidefinido de dimensión n, S_+^n, se tiene $\mathbf{X}^0 = \mathbf{S}^0 = \mathbf{I}$.
- Para el cono de orden p, $\{(u; \mathbf{x}) : u \geq |\mathbf{x}|\}$ se tiene $\mathbf{x}^0 = \mathbf{s}^0 = (1; \mathbf{0})$.

Sea $\mathcal{F}$ el conjunto de todos los puntos factibles $(\mathbf{y}, \mathbf{X} \in K, \tau \geq 0, \theta, \mathbf{S} \in K^*, \kappa \geq 0)$. Por tanto sea $\mathring{\mathcal{F}}$ el conjunto de puntos interiores factibles $(\mathbf{y}, \mathbf{X} \in \mathring{K}, \tau \geq 0, \theta, \mathbf{S} \in \mathring{K}^*$, $\kappa \geq 0)$.

Teorema 3.3 *Considérese la optimización cónica* $(HSDCLP)$

I. $(HSDCLP)$ *es autodual, es decir, su dual tiene forma idéntica a* $(HSDCLP)$

II. $(HSDCLP)$ *tiene una solución óptima y su conjunto de soluciones óptimas está acotado.*

III. $(HSDCLP)$ *tiene un punto interior factible*

$$\mathbf{y} = \mathbf{0}, \quad \mathbf{X} = \mathbf{X}^0, \quad \tau = 1, \quad \theta = 1, \quad , \mathbf{S} = \mathbf{S}^0, \quad \kappa = 1.$$

IV. *Para cualquier punto factible* $(\mathbf{y}, \mathbf{X}, \tau, \theta, \mathbf{S}, \kappa) \in \mathcal{F}$

$$\mathbf{S}^0 \bullet \mathbf{X} + \mathbf{X}^0 \bullet \mathbf{S} + \tau + \kappa - (n+1)\theta = (n+1),$$

y

$$\mathbf{X} \bullet \mathbf{S} + \tau\kappa = (n+1)\theta.$$

V. *El valor de la función objetivo de* $(HSDCLP)$ *es cero, es decir, cualquier solución óptima de* $(HSDCLP)$ *cumple:*

$$\mathbf{X}^* \bullet \mathbf{S}^* + \tau^*\kappa^* = (n+1)\theta^* = 0.$$

Ahora se puede aplicar el algoritmo de punto interior comenzando desde un punto interior inicial disponible en la solución factible para resolver el $(HSDCLP)$.

La relación entre una solución óptima de $(HSDCLP)$ y las soluciones óptimas del problema (PLC) original y su dual (DLC) vienen en el siguiente teorema:

Teorema 3.4 *Sea* $(\mathbf{y}^*, \mathbf{X}^*, \tau^*, \theta^* = 0, \mathbf{S}^*, \kappa^*)$ *una solución óptima de* $(HSDCLP)$ *de rango máximo.*

I. *(PLC) y (DLC) tienen solución óptima si y solo si* $\tau^* > 0$. *En este caso* $\mathbf{X}^*/\tau^*$ *es solución óptima para (PLC) y* $(\mathbf{y}^*/\tau^*, \mathbf{S}^*/\tau^*)$ *es solución óptima de (DLC).*

II. *(PLC) o (DLC) tienen un certificado de inviabilidad si y solo si* $\kappa^* > 0$. *En este caso, tanto* $\mathbf{X}^*/\kappa^*$ *como* $\mathbf{S}^*/\kappa^*$ *son utilizados para probar la inviabilidad.*

III. *En cualquier otro caso,* $\tau^* = \kappa^* = 0$.

Capítulo 4

Resolución de problemas lineales cónicos mediante el entorno de software R.

R es un entorno de programación enfocado al análisis estadístico, pero que tiene muchas otras aplicaciones dada su versalitilidad para la creación de distintos paquetes con los que se extienden sus posibilidades de aplicación en muchos otros aspectos. Dispone también de la posibilidad de creación y tratamiento de gráficos de gran calidad.

Con respecto a su aplicación en programación matemática, tanto lineal como no lineal, se usa el paquete "ROI"(R Optimization Infrastructure), mediante el cuál y con el uso de diferentes "solvers" (código implementado con el enfoque de la resolución de un tipo concreto de problema), es posible abordar la resolución de problemas de programación matemática.

El paquete **ROI** aporta a R la funcionalidad para su utilización como herramienta en la programación y permite formular y resolver distintos tipos de problemas experimentando con diferentes solvers para obtener el más adecuado para cada problema concreto. El marco inicial es el paquete **ROI** implementado por Theußl, Schwendinger, Hornik y Meyer en 2017, junto con los paquetes que le acompañan.

Aunque el paquete **ROI** no pretende la creación de un nuevo lenguaje, si que suministra mecanismos de modelado que encajan perfectamente en lo que R puede ofrecer.

En cuanto al uso de **ROI** para la resolución de problemas de optimización, estos se construyen de manera consistente y se almacenan en objetos simples, lo que posibilita su fácil modificación y por tanto la posibilidad de usarlos de diferentes formas y compartirlos

antes de llegar a la función que nos permita resolverlos de manera unificada. Así pues, el uso del paquete **ROI** y de los paquetes que lo acompañan, permite la creación de nuevos solvers que pueden aumentar la capacidad inicial de **ROI**.

Actualmente, **ROI** puede usarse para resolver problemas de optimización de muchos tipos: lineales, cuadráticos, cónicos, no lineales y mixtos, por lo que puede aplicarse a muchos problemas de optimización que provienen de distintos campos como son la estadística, machine learning y data science.

4.1. Paquetes específicos

El desarrollo reciente de paquetes específicos para resolver diferentes problemas de optimización en R ha sido muy notable. Existe una cantidad de paquetes relacionados con la optimización que ronda actualmente los 100, lo que permite resolver un amplio rango de distintos problemas de optimización con solvers muy especializados que están desarrollados para resolver tipos específicos de problemas muy rápidamente.

En la siguiente tabla se clasifican los solvers disponibles para resolver problemas atendiendo a las características de sus funciones objetivo y restricciones. Se pueden clasificar los solvers en varios grupos en función del tipo de problema al que se aplican y ello permite reflejar en una tabla esta clasificación para una mejor comprensión.

Grupo 1 de solvers: **clpAPI***, **Rglpk**$^{*+}$, **lpSolve**$^{*+}$, **rcdd***, **Rsymphony**$^{*+}$.

Grupo 2 de solvers: **coneproj***, **Dykstra***, **kernlab**, **LowRankQP***, **osqp***, **quadprog***, **ROI.plugin.qpoases**.

Grupo 3 de solvers: **cccp***, **CLSOCP***, **ECOSolveR**$^{*+}$, **Rcsdp***, **Rdsdp*****scs***.

$*$ significa que el solver se restringe a problemas convexos.

$+$ significa que el solver puede aplicarse a modelos con restricciones enteras.

Restricción	Función objetivo		
	Lineal	Cuadrática	Cónica
Lineal	**Grupo 1**, **Grupo 2**, **Grupo 3**.	**Grupo 2**, **Grupo 3**.	**Grupo 3**.
Cuadrática	**Grupo 3**.	**Grupo 3**.	**Grupo 3**.
Cónica	**Grupo 3**.	**Grupo 3**.	**Grupo 3**.

Tabla 4.1: Clasificación de paquetes de R según los tipos de problema de optimización a los que se aplican.

De los solvers citados anteriormente vamos a describir aquellos que son más adecuados para la programación lineal cónica.

La mayoría de los solvers de tipo cónico usan una forma estandar de formulación del problema que se corresponde con:

$$\begin{aligned} \text{minimizar} \quad & \mathbf{a}_0^t \mathbf{x} \\ \text{sujeto a:} \quad & \mathbf{A}\mathbf{x} + \mathbf{s} = \mathbf{b} \\ & \mathbf{s} \in K, \end{aligned} \tag{4.1}$$

Los solvers de tipo cónico disponible en R, así como el tipo de problemas y conos que se soportan puede verse en la siguiente tabla en la que se adjuntan los enlaces para su descarga:

Solver	Lineal	Segundo orden	Semidefinido	Disponibilidad
CLSOCP	✓	✓		Enlace 1
cccp	✓	✓	✓	Enlace 2
ECOSolveR	✓	✓		Enlace 3
Rcsdp	✓	✓	✓	Enlace 4
Rdsdp	✓	✓	✓	Enlace 5
scs	✓	✓	✓	Enlace 6
Enlace 1: https://cran.r-project.org/src/contrib/Archive/CLSOCP/CLSOCP_1.0.tar.gz				
Enlace 2: https://cran.r-project.org/src/contrib/cccp_0.2-9.tar.gz				
Enlace 3: https://cran.r-project.org/src/contrib/ECOSolveR_0.5.5.tar.gz				
Enlace 4: https://cran.r-project.org/src/contrib/Rcsdp_0.1.57.5.tar.gz				
Enlace 5: https://cran.r-project.org/src/contrib/Rdsdp_1.0.5.2.tar.gz				
Enlace 6: Repositorio de R del paquete ROI				

Tabla 4.2: Solvers de resolución de problemas cónicos y tipos de conos en los que se aplican.

4.2. Construcción y resolución de problemas

Para abordar un problema de programación lineal mediante **ROI** tenemos que:

1. Definir el problema como un objeto OP (Problema de Optimización).
2. Elegir el solver más adecuado para resolver el problema planteado, así como los parámetros de configuración del mismo.
3. Usando el solver elegido, resolver el problema.

Aquellos solvers que no están disponibles en el repositorio de ROI pueden instalarse mediante la aplicación RTools tras su descarga desde el enlace que se indica en la tabla 4.2.

Una vez instalados pueden usarse indistintamente tanto los solvers disponibles en ROI como los descargados para la resolución de problemas.

A continuación vamos a ilustrar con ejemplos la aplicación de los pasos anteriormente descritos a diferentes tipos de problemas.

4.2.1. Problema Lineal

Consideremos el siguiente ejemplo:

$$\begin{aligned} \text{maximizar} \quad & 3x_1 + 7x_2 - 12x_3 & \\ \text{sujeto a:} \quad & 5x_1 + 7x_2 + 2x_3 & \leq 61 \\ & 3x_1 + 2x_2 - 9x_3 & \leq 35 \\ & x_1 + 3x_2 + x_3 & \leq 31 \\ & x_1, x_2 \geq 0, & x_3 \in [-10, 10]. \end{aligned}$$

El proceso descrito anteriormente se puede detallar en forma secuencial:

- Se carga la libreria **ROI** en el entorno R.

```
library("ROI")
```

- Definimos las restricciones del problema mediante:

```
A <- rbind(c(5, 7, 2), c(3, 2, -9), c(1, 3, 1))
dir <- c("<=", "<=", "<=")
rhs <- c(61, 35, 31)
```

- Planteamos el problema usando las definiciones anteriores:

```
lp <- OP(objective = L_objective(c(3, 7, -12)),
constraints = L_constraint(A, dir = dir, rhs = rhs),
bounds = V_bound(li = 3, ui = 3, lb = -10, ub = 10, nobj = 3),
maximum = TRUE)
```

Alternativamente podemos definir el problema paso a paso:

```
lpalt <- OP()
objective(lpalt) <- L_objective(c(3, 7, -12))
constraints(lpalt) <- L_constraint(A,
dir = c("<=", "<=", "<="), rhs = rhs)
bounds(lpalt) <- V_bound(li = 3, ui = 3, lb = -10, ub = 10,
nobj = 3)
maximum(lpalt) <- TRUE
```

- En ambos casos ROI nos describe el problema planteado mediante:

```
lp
```

  o bien, en el caso alternativo

```
lpalt
```

  obteniendo la descripción que facilita R del tipo de problema que le hemos definido:

```
## ROI Optimization Problem:
##
## Maximize a linear objective function of length 3 with
## - 3 continuous objective variables,
##
## subject to
## - 3 constraints of type linear.
## - 1 lower and 1 upper non-standard variable bound.
```

- La descripción que hace R nos facilita la búsqueda de un solver apropiado para el problema planteado. Para ver los solvers disponibles en los repositorios de ROI utilizamos:

```
ROI_available_solvers(lp)[, c("Package", "Repository")]
```

  Obteniendo:

```
##      Paquete              Repositorio
##  1   ROI.plugin.alabama   https://CRAN.R-project.org
##  2   ROI.plugin.cplex     https://CRAN.R-project.org
```

```
##  4  ROI.plugin.ecos      https://CRAN.R-project.org
##  5  ROI.plugin.glpk      https://CRAN.R-project.org
##  6  ROI.plugin.highs     https://CRAN.R-project.org
##  7  ROI.plugin.ipop      https://CRAN.R-project.org
##  8  ROI.plugin.lpsolve   https://CRAN.R-project.org
## 10  ROI.plugin.neos      https://CRAN.R-project.org
## 11  ROI.plugin.nloptr    https://CRAN.R-project.org
## 13  ROI.plugin.osqp      https://CRAN.R-project.org
## 14  ROI.plugin.qpoases   https://CRAN.R-project.org
## 16  ROI.plugin.scs       https://CRAN.R-project.org
## 17  ROI.plugin.symphony  https://CRAN.R-project.org
## 18  ROI.plugin.alabama   https://gitlab.com/roigrp/solver
## 19  ROI.plugin.cccp      https://gitlab.com/roigrp/solver
## 21  ROI.plugin.ecos      https://gitlab.com/roigrp/solver
## 22  ROI.plugin.gurobi    https://gitlab.com/roigrp/solver
## 23  ROI.plugin.highs     https://gitlab.com/roigrp/solver
## 24  ROI.plugin.mosek     https://gitlab.com/roigrp/solver
## 26  ROI.plugin.neos      https://gitlab.com/roigrp/solver
## 27  ROI.plugin.nloptr    https://gitlab.com/roigrp/solver
## 29  ROI.plugin.osqp      https://gitlab.com/roigrp/solver
## 30  ROI.plugin.qpoases   https://gitlab.com/roigrp/solver
## 32  ROI.plugin.scs       https://gitlab.com/roigrp/solver
## 33  ROI.plugin.symphony  https://gitlab.com/roigrp/solver
## 34  ROI.plugin.cbc       https://github.com/dirkschumacher
## 35  ROI.plugin.clp       https://github.com/datastorm-open
```

Podemos comprobar los solvers que tenemos instalados en nuestro entorno por si es necesario descargar e instalar desde los repositorios en caso de no tener instalado el solver adecuado mediante:

```
ROI_installed_solvers()
```

En nuestro caso tenemos:

```
##             nlminb
## "ROI.plugin.nlminb"
```

Por lo tanto, dado que se va a utilizar el paquete “glpk” procedemos a instalarlo y a registrarlo mediante:

```
install.packages("ROI.plugin.glpk")
library(ROI.plugin.glpk)
```

- Podemos plantear a ROI cuáles son los solvers instalados y registrados que se pueden aplicar a este problema:

```
ROI_applicable_solvers(lp)
```

En este caso obtenemos únicamente "glpk" porque no tenemos más solvers instalados que se puedan aplicar para resolver el problema:

```
## [1] "glpk"
```

- Procedemos a resolver el problema usando el solver elegido:

```
(lp_sol <- ROI_solve(lp, solver = "glpk"))
```

obteniendo como resultado el valor óptimo de la función objetivo:

```
## Optimal solution found.
## The objective value is: 8.670149e+01
```

y los valores de las variables para ese valor óptimo de la función objetivo se obtienen mediante:

```
solution(lp_sol)
```

Dichos valores son:

```
## [1]  0.000000  9.238806 -1.835821
```

También, aunque se ha visto anteriormente, puede obtenerse el valor de la función objetivo optimizada empleando:

```
objective(lp)(solution(lp_sol))
```

Obteniendo como resultado:

```
## [1] 86.70149
```

Se puede extraer el status del solver:

```
lp_sol$status
```

Obteniendo como resultado:

```
## $code
## [1] 0
##
## $msg
##   solver glpk
##     code 5
##   symbol GLP_OPT
##  message Solution is optimal.
## roi_code 0
```

Y también un resumen de su aplicación con el uso de:

```
lp_sol$message
```

Obteniendo como resultado:

```
## $optimum
## [1] 86.70149
##
## $solution
## [1]  0.000000  9.238806 -1.835821
##
## $status
## [1] 5
##
## $solution_dual
## [1] -4.298507  0.000000  0.000000
##
## $auxiliary
## $auxiliary$primal
## [1] 61.0000 35.0000 25.8806
```

```
##
## $auxiliary$dual
## [1] 0.5820896 1.4626866 0.0000000
##
##
## $sensitivity_report
## [1] NA
```

4.2.2. Problema Lineal con Solución Múltiple

Veamos a continuación cómo se resolvería un problema con soluciones múltiples. Planteamos el siguiente problema:

$$\begin{aligned} \text{minimizar} \quad & -x_1 - x_2 - x_3 - x_4 - 99x_5 \\ \text{sujeto a:} \quad & x_1 + x_2 \leq 1 \\ & x_3 + x_4 \leq 1 \\ & x_4 + x_5 \leq 1 \\ & x_i \in \{0, 1\} \end{aligned} \tag{4.2}$$

Para obtener todas las soluciones usamos el solver específicamente desarrollado para estos casos, ya que "msbinlp" permite recuperar todas las soluciones al problema de optimización planteado en 4.2. Veámoslo en R.

En primer lugar se plantea el problema mediante:

```
A <- rbind(c(1, 1, 0, 0, 0), c(0, 0, 1, 1, 0),c(0, 0, 0, 1, 1))
dir <- c("<=", "<=", "<=")
rhs <- rep.int(1, 3)

blp <- OP(objective = L_objective(c(-1, -1, -1, -1, -99)),
constraints = L_constraint(L = A, dir = dir, rhs = rhs),
types = rep("B", 5L))

blp
```

Obteniendo como respuesta:

```
## ROI Optimization Problem:
##
## Minimize a linear objective function of length 5 with
## - 5 binary objective variables,
##
## subject to
## - 3 constraints of type linear.
## - 0 lower and 0 upper non-standard variable bounds.
```

Dado que en este caso deseamos obtener soluciones múltiples, se va a aplicar un solver específico, por lo que se le indica a R que debe resolver el problema mediante dicho solver. Previamente debemos cargar la librería correspondiente al solver "msbinlp":

```
library(ROI.plugin.msbinlp)
```

Una vez cargada, resolvemos el problema mediante:

```
(blp_sol <- ROI_solve(blp, solver = "msbinlp", method = "glpk",
nsol_max = 32))
```

R nos devuelve el valor óptimo para la función objetivo y el número de soluciones óptimas del problema:

```
## 2 optimal solutions found.
## The objective value is: -1.010000e+02
```

Para ver cuáles son las 2 soluciones óptimas encontradas usamos:

```
 solution(blp_sol)
```

Tenemos que las soluciones son:

```
## [[1]]
## [1] 0 1 1 0 1
##
## [[2]]
## [1] 1 0 1 0 1
```

4.2.3. Problema SDP

Veamos a continuación un ejemplo de problema con cono semidefinido positivo.

Tengamos el problema de programación semidefinida siguiente:

$$\begin{aligned} \text{maximizar} \quad & x_1 + x_2 - x_3 \\ \text{sujeto a:} \quad & x_1 \begin{pmatrix} 10 & 3 \\ 3 & 10 \end{pmatrix} + x_2 \begin{pmatrix} 6 & -4 \\ -4 & 10 \end{pmatrix} + x_3 \begin{pmatrix} 8 & 1 \\ 1 & 6 \end{pmatrix} \preceq \begin{pmatrix} 16 & -13 \\ -13 & 60 \end{pmatrix} \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

El planteamiento del problema con sus restricciones en R se haría mediante:

```
F1 <- rbind(c(10, 3), c(3, 10))
F2 <- rbind(c(6, -4), c(-4, 10))
F3 <- rbind(c(8, 1), c(1, 6))
F0 <- rbind(c(16, -13), c(-13, 60))
psd <- OP(objective = L_objective(c(1, 1, -1)),
constraints = C_constraint(L = vech(F1, F2, F3), cone = K_psd(3),
rhs = vech(F0)))
psd
```

Obteniéndose como respuesta:

```
## ROI Optimization Problem:
##
## Minimize a linear objective function of length 3 with
## - 3 continuous objective variables,
##
## subject to
## - 3 constraints of type conic.
##   |- 3 conic constraints of type 'psd'
## - 0 lower and 0 upper non-standard variable bounds.
```

Veamos qué solvers podemos utilizar para resolverlo:

```
ROI_available_solvers(psd)[, c("Package", "Repository")]
```

Los solvers para resolver que nos devuelve R son:

```
##      Paquete              Repositorio
## 16   ROI.plugin.scs       https://CRAN.R-project.org
## 32   ROI.plugin.scs       https://gitlab.com/roigrp/solver
```

Cargamos la librería correspondiente al solver elegido:

```
library(scs)
```

Procedemos a resolver el problema mediante el solver anterior con:

```
(psd_sol <- ROI_solve(psd, solver = "scs"))
```

Obteniendo el valor óptimo de la función objetivo:

```
## Optimal solution found.
## The objective value is: -1.486437e+00
```

Y los valores óptimos de las variables:

```
solution(psd_sol)
## [1] 5.782736e-06 1.065260e-06 1.486444e+00
```

4.2.4. Problema Cono de Segundo Orden

Veamos ahora un par de casos de problemas de cono de Segundo Orden.

Ejemplo 1

$$\begin{aligned} \text{maximizar} \quad & x_1 + x_2 \\ \text{sujeto a:} \quad & \sqrt{(2+3x_1)^2 + (4+5x_2)^2} \leq 6 + 7u \\ & x_1, x_2 \in \mathbb{R},\ u \in (-\text{ínf}, 9] \end{aligned} \tag{4.3}$$

para $x = (x_1, x_2, u)^t$, la restricción:

$$\sqrt{(2+3x_1)^2 + (4+5x_2)^2} \leq 6 + 7u$$

es equivalente a:

$$\sqrt{(b_2 - a_2^t x)^2 + (b_3 - a_3^t x)^2} \leq b_1 - a_1^t x,$$

donde

$$A = \begin{pmatrix} 0 & 0 & -7 \\ -3 & 0 & 0 \\ 0 & -5 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 6 \\ 2 \\ 4 \end{pmatrix} \tag{4.4}$$

Viendo esto en R, primero planteamos el problema mediante:

```
 soc1 <- OP(c(1, 1, 0),
C_constraint(L = rbind(c(0, 0, -7), c(-3, 0, 0), c(0, -5, 0)),
cone = K_soc(3), rhs = c(6, 2, 4)), maximum = TRUE,
bounds = V_bound(ld = -Inf, ui = 3, ub = 9, nobj = 3))
soc1
```

De donde obtenemos:

```
## ROI Optimization Problem:
##
## Maximize a linear objective function of length 3 with
## - 3 continuous objective variables,
##
## subject to
## - 3 constraints of type conic.
##   |- 3 conic constraints of type 'soc'
## - 3 lower and 1 upper non-standard variable bound.
```

Veamos qué solvers podemos utilizar para resolverlo:

```
ROI_available_solvers(soc1)[, c("Package", "Repository")]
```

Se pueden utilizar los solvers:

```
##     Paquete              Repositorio
##  4  ROI.plugin.ecos      https://CRAN.R-project.org
## 16  ROI.plugin.scs       https://CRAN.R-project.org
## 21  ROI.plugin.ecos      https://gitlab.com/roigrp/solver
## 24  ROI.plugin.mosek     https://gitlab.com/roigrp/solver
## 32  ROI.plugin.scs       https://gitlab.com/roigrp/solver
```

Cargamos la libreria correspondiente al solver elegido:

```
library(ecos)
```

Procedamos a resolver el problema con este solver mediante:

```
(soc1_sol <- ROI_solve(soc1,solver = "ecos"))
```

Obtenemos un valor óptimo de la función objetivo:

```
## Optimal solution found.
## The objective value is: 2.535571e+01
```

El cual se encuentra para los valores de las variables:

```
solution(soc1_sol)
## [1] 19.055671  6.300041  9.000000
```

Ejemplo 2 Considérese ahora el siguiente problema:

$$\begin{aligned} \text{minimizar} \quad & \sqrt{x_1^2 + x_2^2} \\ \text{sujeto a:} \quad & x_1 + x_2 = 2 \\ & x_1, x_2 \geq 0. \end{aligned}$$

El cual puede ser reformulado como:

$$\begin{aligned} \text{minimizar} \quad & u \\ \text{sujeto a:} \quad & \sqrt{x_1^2 + x_2^2} \leq u \\ & x_1 + x_2 = 2 \\ & x_1, x_2 \geq 0. \end{aligned}$$

Podemos resolverlo en R como sigue.

Planteamos el problema:

```
A <- rbind(c(0, 0, -1), c(-1, 0, 0), c(0, -1, 0))
soc2 <- OP(objective = L_objective(c(0, 0, 1)),
constraints = c(C_constraint(A, c(K_soc(3)), c(0, 0, 0)),
L_constraint(c(1, 1, 0), "==", 2)))
soc2
```

Obteniendo:

```
## ROI Optimization Problem:
##
## Minimize a linear objective function of length 3 with
## - 3 continuous objective variables,
##
## subject to
## - 4 constraints of type conic.
##   |- 3 conic constraints of type 'soc'
##   |- 1 conic constraint of type 'zero'
## - 0 lower and 0 upper non-standard variable bounds.
```

Veamos que solvers podemos utilizar para resolverlo:

```
ROI_available_solvers(soc2)[, c("Package", "Repository")]
```

Se pueden utilizar:

```
##     Paquete              Repositorio
##  4  ROI.plugin.ecos      https://CRAN.R-project.org
## 16  ROI.plugin.scs       https://CRAN.R-project.org
## 21  ROI.plugin.ecos      https://gitlab.com/roigrp/solver
## 24  ROI.plugin.mosek     https://gitlab.com/roigrp/solver
## 32  ROI.plugin.scs       https://gitlab.com/roigrp/solver
```

Cargamos la librería correspondiente al solver elegido:

```
library(ecos)
```

Procedamos a resolver el problema con este solver mediante:

```
(soc2_sol<-ROI_solve(soc2,solver = "ecos"))
```

Obteniendo como valor óptimo:

```
## Optimal solution found.
## The objective value is: 1.414214e+00
```

Para los valores de las variables usamos:

```
solution(soc2_sol)
[1] 1.000000 1.000000 1.414214
```

4.3. Resolución de un problema mediante diferentes métodos

Para ilustrar como pueden usarse diferentes métodos para la resolución de un mismo problema vamos a usar el caso del problema que se reflejó en el Ejemplo 1.

4.3.1. Resolución como Problema de Programación Lineal

En el referido ejemplo se trataba del problema que ya aparecia formulado como problema de programación lineal con el siguiente enunciado:

$$\begin{aligned} \text{minimizar} \quad & 2x_1 + x_2 + x_3 \\ \text{sujeto a:} \quad & x_1 + x_2 + x_3 = 1 \\ & (x_1, x_2, x_3) \geq \mathbf{0}. \end{aligned}$$

Planteándolo y resolviéndolo en R tendríamos que:

Se escribe el problema:

```
# Restricciones
Aej1 <- rbind(c(1, 1, 1))
direj1 <- c("==")
rhsej1 <- c(1)
# Planteamiento del problema
lpej1 <- OP(objective = L_objective(c(2, 1, 1)),
constraints = L_constraint(Aej1, dir = direj1, rhs = rhsej1))
lpej1
```

Obteniendo la descripción del mismo:

```
## ROI Optimization Problem:
##
## Minimize a linear objective function of length 3 with
## - 3 continuous objective variables,
```

```
##
## subject to
## - 1 constraint of type linear.
## - 0 lower and 0 upper non-standard variable bounds.
```

Veamos que solvers podemos utilizar para resolverlo:

```
ROI_available_solvers(lpej1)[, c("Package", "Repository")]
```

Buscamos los solvers adecuados para resolver el problema:

```
##      Paquete               Repositorio
##   1  ROI.plugin.alabama    https://CRAN.R-project.org
##   2  ROI.plugin.cplex      https://CRAN.R-project.org
##   4  ROI.plugin.ecos       https://CRAN.R-project.org
##   5  ROI.plugin.glpk       https://CRAN.R-project.org
##   6  ROI.plugin.highs      https://CRAN.R-project.org
##   7  ROI.plugin.ipop       https://CRAN.R-project.org
##   8  ROI.plugin.lpsolve    https://CRAN.R-project.org
##  10  ROI.plugin.neos       https://CRAN.R-project.org
##  11  ROI.plugin.nloptr     https://CRAN.R-project.org
##  13  ROI.plugin.osqp       https://CRAN.R-project.org
##  14  ROI.plugin.qpoases    https://CRAN.R-project.org
##  16  ROI.plugin.scs        https://CRAN.R-project.org
##  17  ROI.plugin.symphony   https://CRAN.R-project.org
##  18  ROI.plugin.alabama    https://gitlab.com/roigrp/solver
##  19  ROI.plugin.cccp       https://gitlab.com/roigrp/solver
##  21  ROI.plugin.ecos       https://gitlab.com/roigrp/solver
##  22  ROI.plugin.gurobi     https://gitlab.com/roigrp/solver
##  23  ROI.plugin.highs      https://gitlab.com/roigrp/solver
##  24  ROI.plugin.mosek      https://gitlab.com/roigrp/solver
##  26  ROI.plugin.neos       https://gitlab.com/roigrp/solver
##  27  ROI.plugin.nloptr     https://gitlab.com/roigrp/solver
##  29  ROI.plugin.osqp       https://gitlab.com/roigrp/solver
##  30  ROI.plugin.qpoases    https://gitlab.com/roigrp/solver
##  32  ROI.plugin.scs        https://gitlab.com/roigrp/solver
##  33  ROI.plugin.symphony   https://gitlab.com/roigrp/solver
##  34  ROI.plugin.cbc        https://github.com/dirkschumacher
##  35  ROI.plugin.clp        https://github.com/datastorm-open
```

Cargamos la librería correspondiente al solver elegido:

```
library(scs)
```

Procedamos a resolver el problema con este solver mediante:

```
(lpej1 <- ROI_solve(lpej1, solver = "scs"))
```

Obteniendo el valor óptimo de la función objetivo:

```
## Optimal solution found.
## The objective value is: 1.000000e+00
```

Podemos encontrar los valores de las variables para ese valor óptimo mediante:

```
solution(lpej1)
```

Estos valores son:

```
## [1] 3.417062e-09 5.000000e-01 5.000000e-01
```

4.3.2. Resolución como Problema SDP

Veamos este mismo problema planteado como un problema semidefinido positivo. Su formulación como problema SDP es:

$$\begin{aligned} \text{minimizar} \quad & \mathbf{C} \bullet \mathbf{X} \\ \text{sujeto a:} \quad & \mathbf{A}_1 \bullet \mathbf{X} = 1,\ \mathbf{X} \in S_+^2, \end{aligned}$$

donde:

$$\mathbf{C} = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 1 \end{bmatrix} \qquad \mathbf{A}_1 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}.$$

Reescribiendo este problema tenemos que:

$$\begin{aligned} \text{minimizar} \quad & 2x_1 + x_2 + x_3 \\ \text{sujeto a:} \quad & x_1 \begin{pmatrix} 0 & 0.5 \\ 0.5 & 0 \end{pmatrix} + x_2 \begin{pmatrix} 0 & 0.5 \\ 0.5 & 0 \end{pmatrix} + x_3 \begin{pmatrix} 0 & 0.5 \\ 0.5 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0.5 \\ 0.5 & 0 \end{pmatrix} \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

Veámoslo en R. Empezamos planteando el problema:

```
F1 <- rbind(c(0, 0.5), c(0.5, 0))
F2 <- rbind(c(0, 0.5), c(0.5, 0))
F3 <- rbind(c(0, 0.5), c(0.5, 0))
F0 <- rbind(c(0, 0.5), c(0.5, 0))

psdej1 <- OP(objective = L_objective(c(2, 1, 1)),
constraints = C_constraint(L = vech(F1,F2,F3), cone = K_psd(3),
rhs = vech(F0)))

psdej1
```

De donde se obtiene que:

```
## ROI Optimization Problem:
##
## Minimize a linear objective function of length 3 with
## - 3 continuous objective variables,
##
## subject to
## - 3 constraints of type conic.
##   |- 3 conic constraints of type 'psd'
## - 0 lower and 0 upper non-standard variable bounds.
```

Veamos qué solvers podemos utilizar para resolverlo:

```
ROI_available_solvers(psdej1)[, c("Package", "Repository")]
```

Se pueden utilizar los siguientes:

```
##     Paquete              Repositorio
## 16  ROI.plugin.scs       https://CRAN.R-project.org
## 32  ROI.plugin.scs       https://gitlab.com/roigrp/solver
```

Cargamos la librería correspondiente al solver elegido:

```
library(scs)
```

Procedamos ahora a resolverlo mediante el solver elegido con:

```
(psdej1_sol <- ROI_solve(psdej1, solver = "scs"))
```

Obteniendo la solución óptima:

```
## Optimal solution found.
## The objective value is: 1.000002e+00
```

Para los valores de las variables:

```
solution(psdej1_sol)
## [1] 5.848197e-06 4.999950e-01 4.999950e-01
```

4.3.3. Resolución como Problema de Cono de Segundo Orden

Por último, veamos el problema planteado como un problema de cono de segundo orden aunque la región factible se modifica puesto que x_2 y x_3 pueden tomar en este caso valores negativos:

$$\begin{aligned} \text{minimizar} \quad & 2x_1 + x_2 + x_3 \\ \text{sujeto a:} \quad & x_1 + x_2 + x_3 = 1 \\ & \sqrt{x_2^2 + x_3^2} \leq x_1. \end{aligned}$$

Planteando el problema en R tenemos que:

```
Aej13 <- rbind(c(-1, 0, 0), c(0, -1, 0), c(0, 0, -1))
soej13 <- OP(objective = L_objective(c(2, 1, 1)),
constraints = c(C_constraint(L = Aej13, cone = c(K_soc(3)),
rhs = c(0, 0, 0)), L_constraint(c(1, 1, 1), "==", 1)))
soej13
```

Obteniendo como descripción del problema:

```
## ROI Optimization Problem:
##
## Minimize a linear objective function of length 3 with
## - 3 continuous objective variables,
##
## subject to
## - 4 constraints of type conic.
```

```
##   |- 3 conic constraints of type 'soc'
##   |- 1 conic constraint of type 'zero'
## - 0 lower and 0 upper non-standard variable bounds.
```

Veamos qué solvers utilizar para resolverlo:

```
ROI_available_solvers(soej13)[, c("Package", "Repository")]
```

Los solvers adecuados son:

```
##     Paquete              Repositorio
##  4  ROI.plugin.ecos      https://CRAN.R-project.org
## 16  ROI.plugin.scs       https://CRAN.R-project.org
## 21  ROI.plugin.ecos      https://gitlab.com/roigrp/solver
## 24  ROI.plugin.mosek     https://gitlab.com/roigrp/solver
## 32  ROI.plugin.scs       https://gitlab.com/roigrp/solver
```

Cargamos la librería correspondiente al solver elegido:

```
library(ecos)
```

Procedemos a resolver el problema con dicho solver:

```
(soej13_sol <- ROI_solve(soej13, solver = "ecos"))
```

Obteniendo el valor óptimo de la función

```
## Optimal solution found.
## The objective value is: 1.414214e+00
```

Los valores de las variables para dicho valor óptimo se obtienen mediante:

```
solution(soej13_sol)
```

Y estos valores son:

```
## [1] 0.4142136 0.2928932 0.2928932
```

Conclusiones

El propósito de este trabajo fin de grado referente al análisis de la programación lineal cónica se ha abordado desde el punto de vista teórico consultando las fuentes bibliográficas que se citan, de las que nos hemos centrado en los aspectos más relevantes de su aplicación y la relación entre problema primal y dual. También se ha realizado un análisis de las distintas propiedades, algorítmos y métodos de resolución en los problemas de programación lineal cónica. A lo largo del trabajo, cada uno de los casos se ha ido ilustrando con distintos ejemplos.

Los problemas analizados son aquellos que se corresponden con tres tipos de cono: el ortante n-dimensional no negativo, el cono de matrices semidefinidas positivas y el cono de orden-p.

Desde el punto de vista práctico se ha abordado la resolución mediante software de los problemas que se han mostrado como ejemplos de los distintos casos citados para problemas de programación lineal, programación semidefinida positiva y cono de segundo orden formulándose los problemas de la manera adecuada a cada caso.

El software usado para la resolución de problemas de optimización ha sido ROI, un paquete de R destinado a tal fin que contiene distintos tipos de solvers, cada uno de los cuales se orienta a la resolución de un tipo de problema con características concretas y diferenciadas.

Por último se ha ilustrado mediante un caso concreto la aplicación de ROI para diferentes métodos de resolución del mismo caso, lo que permite realizar una comparación de los distintos métodos usados.

Bibliografía

[1] David G. Luenberger, Yinyu Ye, *Linear and Nonlinear Programming. Fifth Edition.* Springer, 2021.

[2] W.F. Sheppard, *On the calculation of the double integral expressing normal correlation.*. Trans. Camb. Philos. Soc. 19, 23–66, 1900.

[3] Theußl, S., Schwendinger, F., & Hornik, K., *ROI: The R Optimization Infrastructure Package*. Research Report Series / Department of Statistics and Mathematics No. 133, 2017.

[4] Boyd S., Vandenberghe L., *Convex Optimization*. Cambridge University Press, 2009.

[5] Nemirovski A., *Advances in Convex Optimization: Conic Programming*. Proceedings of International Congress of Mathematicians, Madrid, 2006.

[6] Huertas V., *Historia de la Programación Matemática*, 2017, https://www.reduccionalabsurdo.es/downloads/historia-pm/histPM.pdf.

[7] Hornik K., Meyer D., Schwendinger F., Stefan Theussl, Diethelm Wuertz, *R Optimization Infrastructure*. 2023. https://cran.r-project.org/web/packages/ROI/ROI.pdf.

[8] Ronald Hochreiter, Florian Schwendinger, *ROI Plug-in NEOS*. 2020. https://cran.r-project.org/web/packages/ROI.plugin.neos/vignettes/ROI.plugin.neos_Introduction.pdf.

[9] Universidad de Valencia, *Dualidad en programación lineal.* https://www.uv.es/~sala/Clase11.pdf.

[10] Blekherman G., Parrilo P.A., Thomas R.R., *Semidefinite OptimizatiOn and Convex algebraiC geometry*. Siam, 2013.