Trabajo Fin de Máster

Problemas de optimización entera mixta no lineal

Presentado por:
Ángela Valdayo Raya

Supervisor:
Dr. Justo Puerto Albandoz, Universidad de Sevilla

UNIVERSIDAD D SEVILLA

14 de septiembre de 2015

Índice general

Introducción.

La investigación y el análisis de las teorías en el manejo del riesgo asociado a cualquier inversión son fundamentales para el desarrollo de nuevas ideas y aplicaciones en el área de las finanzas.

El modelo de Harry Markowitz, desde su origen en 1952, contribuyó a variados desarrollos y derivaciones, proporcionando el marco conceptual del manejo eficiente de un portafolio, maximizando la rentabilidad esperada y controlando el riesgo.
Sin embargo, este modelo se vuelve mucho más complejo cuando añadimos restricciones de cardinalidad sobre el soporte del portafolio.

El objetivo general de este trabajo será presentar dos algoritmos de programación cuadrática entera-mixta para la resolución de este modelo. El primero de ellos estará basado en cuatro tipos de planos de corte. El segundo de ellos, estará basado principalmente en el algoritmo de Lemke.

Además, en dicho trabajo se incluirán experimentos computacionales que en efecto, muestren la complejidad de dicho modelo; sobre todo cuando hacemos comparaciones con el modelo continuo.

Capítulo 1

Preliminares

En este capítulo vamos a introducir el problema de optimización esperanza-varianza de Markowitz (MVO). Así mismo, describiremos algunos modelos de programación entera. Estos modelos, integran algunas variables que toman valores enteros. Dichas variables enteras, con frecuencia, suelen tomar sólo los valores en $\{0, 1\}$, ya que este tipo de variables permiten representar condiciones lógicas.

Este tipo de modelos permite representar sistemas mucho más complejos. A cambio, la resolución de los mismos se complica, ya que no se puede utilizar la suavidad de las funciones para inferir el comportamiento de las mismas cerca del óptimo. De esta forma, problemas con unas pocas decenas de variables pueden ser casi imposibles de resolver.

En este capítulo introduciremos la optimización de carteras con cota superior en el número de variables positivas. Este problema lo trataremos con más detenimiento en el capítulo siguiente como un modelo de programación entera-mixta. Por último, veremos otras dos aplicaciones de la programación entera en el *problema de las subastas combinatorias* y en el *problema de la caja de seguridad*.

1.1. Modelos de programación: optimización de carteras

1.1.1. Optimización media-varianza

Sean S_1, S_2, ... S_n $(n \geq 2)$ n acciones con ganancias aleatorias (variables aleatorias), con:

$$\begin{array}{lcl} \mu_i & = & \text{Ganancia esperada de cada } S_i \\ \sigma_i & = & \text{Desviación estándar (típica) de cada } S_i \\ \rho_{i,j} & = & \text{Coeficiente de correlación de las variables } S_i, S_j \end{array}$$

Consideramos:

$$\mu = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_n \end{bmatrix}$$

$Q = (\sigma_{i,j})_{nxn}$ la matriz de covarianzas, simétrica, con

$$\sigma_{i,j} = \begin{cases} \sigma_i^2 & i = j \\ \rho_{i,j}\sigma_i\sigma_j & i \neq j \end{cases}$$

$x_i =$ la proporción del total de fondos invertidos en el valor $i\,(i = 1, ..., n)$

Recordatorio 1.1. Recordamos que el coeficiente de correlación es un operador simétrico,

$$\rho_{i,j} = \rho(S_i, S_j) = \frac{\sigma(S_i, S_j)}{\sigma_i\sigma_j} = \frac{\sigma(S_j, S_i)}{\sigma_j\sigma_i} = \rho(S_j, S_i) = \rho_{j,i}$$

lo que garantiza la simetría de Q.

Observación 1.2. Podemos representar la ganancia esperada y la varianza de la cartera resultante $x = (x_1, ..., x_n)$ como:

$$\boxed{E(x) = x_1\mu_1 + ... + x_n\mu_n = \mu^T x}$$

$$\boxed{Var(x) = \sum_{i,j} \rho_{i,j}\sigma_i\sigma_j x_i x_j = x^T Q x} \text{ donde } \rho_{i,i} = 1$$

Definición 1.3. *Diremos que una* acción es redundante *cuando no sea requerida específicamente por una empresa para generar ganancias y flujo de caja a partir de sus operaciones. Aquellas que son "redundantes" a las operaciones de la empresa.*

Observación 1.4. $Var(x) \geq 0 \implies x^T Q x \geq 0 \quad \forall x \implies Q$ semidefinida positiva. En este trabajo vamos a suponer que Q es definida positiva, lo que equivale a asumir que no hay acciones redundantes en nuestra solección $S_1, ...S_n$

Definición 1.5. *Definimos el* conjunto de carteras admisibles *como el conjunto*

$$\chi := \{x : Ax = b,\, Cx \geq d\}$$

donde:

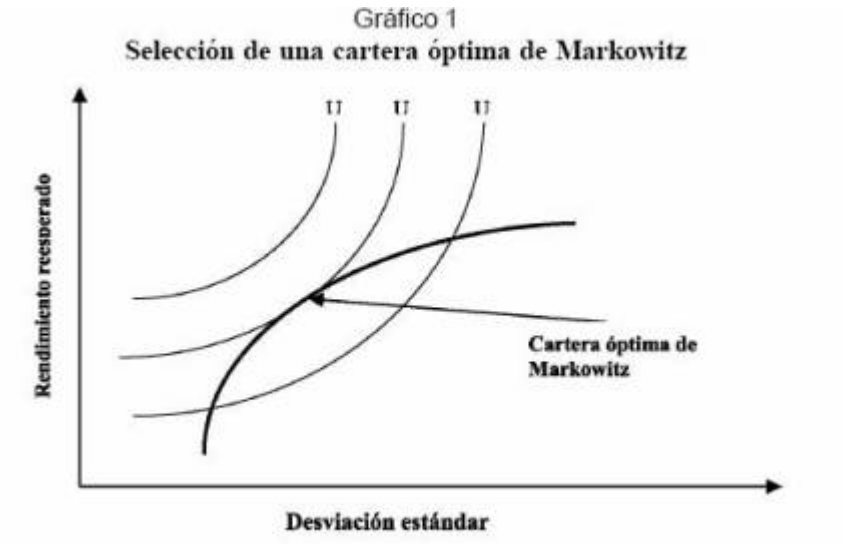
- *A es una matriz m x n*
- *b es un vector m-dim*
- *C es una matriz p x n*
- *d es un vector p-dim*

En particular, una de las restricciones de este conjunto es

$$\sum_{i=1}^{n} x_i = 1$$

Definición 1.6.

- Cartera eficiente*: puede verse como la cartera que tiene el máximo rendimiento esperado de todas las carteras con la misma varianza, o bien, puede verse como aquella que tiene la mínima varianza entre todas aquellas que tienen al menos una cierta ganancia esperada.*
- Frontera eficiente*: Aquella formada por todas las carteras eficientes de la cartera universo. Esta frontera es a menudo representada como una curva en un grafo 2-dimensional donde las coordenadas de un punto corresponden a la ganancia esperada y la desviación típica de la ganancia de una cartera eficiente.*



Supongamos que queremos buscar aquella cartera eficiente con varianza mínima, entre todas aquellas que tienen al menos una cierta ganancia esperada. Como Q es definida positiva, la varianza es una función esctrictamente convexa y entonces $\exists! x \in \chi$ tal que tiene mínima varianza .
Denotemos esta cartera por x_{min} y su ganancia $\mu^T x_{min} = R_{min}$. Denotemos también como R_{max} = la máxima ganancia para una cartera admisible. Siguiendo esta notación, podemos definir el siguiente problema:

PROBLEMA DE OPTIMIZACIÓN ESPERANZA-VARIANZA DE MARKOWITZ (M.V.O.)

El problema de optimización esperanza-varianza de Markowitz consiste en encontrar la cartera de varianza mínima que tenga una ganancia mínima esperada; o en otras palabras, minimizar el riesgo para una rentabilidad mínima esperada. Matemáticamente, esta formulación produce el siguiente **problema de programación cuadrática**:

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} x^T Q x \\ \text{s.a.} \quad & \mu^T x \geq R \\ & Ax = b \\ & Cx \geq d \end{aligned} \tag{1.1}$$

La primera restricción indica que la ganancia esperada no puede ser menor que un cierto valor prefijado R. Esta restricción es necesaria, ya que cuando minimizamos la varianza (el riesgo) también estamos minimizado la ganancia. Si resolvemos este problema para valores de R entre R_{min} y R_{max}, obtendremos el conjunto de todas las carteras eficientes. Por otro lado, la función objetivo corresponde a un medio de la varianza total de la cartera. El hecho de añadir la constante $\frac{1}{2}$ es para simplicar las ecuaciones correspondientes a las condiciones de optimalidad. Obviamente, esto no afecta a la solución óptima.

Como podemos observar, estamos ante un problema de programación cuadrática convexo, en el cual las condiciones de primer orden son necesarias y suficientes para la optimalidad. Veamos cuáles son:

- Primero, transformamos el problema 1.1 en:

$$\begin{aligned} \min_{x} \quad & \frac{1}{2}x^T Q x \\ \text{s.a.} \quad & -\mu^T x + R \leq 0 \\ & Ax = b \\ & -Cx + d \leq 0 \end{aligned} \tag{1.2}$$

- Como tenemos restricciones lineales, éstas forman un recinto convexo. En este caso (caso de restricciones lineales) no hace falta exigirle regularidad a la solución óptima x_R, es decir, no tenemos que exigir que los gradientes de las funciones de las restricciones sean l.i. en x_R, y además, las condiciones de primer orden son necesarias y suficientes para la optimalidad. Estas condiciones son:

x_R es solución óptima de 1,1 $\iff$ $\exists \lambda_R \in \mathbb{R}, \gamma_E \in \mathbb{R}^m, \gamma_I \in \mathbb{R}^p$ tal que:

$$\nabla_x L(x_R, \lambda_R, \gamma_E, \gamma_I) = 0$$

$$\begin{cases} \nabla_\lambda L(x_R, \lambda_R, \gamma_E, \gamma_I) = 0; \\ \nabla_\gamma L(x_R, \lambda_R, \gamma_E, \gamma_I) = 0; \\ \nabla_{\gamma'} L(x_R, \lambda_R, \gamma_E, \gamma_I) = 0 \end{cases}$$

$$\lambda_R, \gamma_I \geq 0$$

$$\text{C. de Holgura complementaria} \quad \begin{cases} \lambda_R(\mu x_R - R) = 0 \\ \gamma_I^T(cx_r - d) = 0 \end{cases}$$

Donde $\boxed{L(x, \lambda, \gamma, \gamma') = \frac{1}{2}x^T Q x + \lambda(R - \mu^T x) + (Ax - b)\gamma + (d - cx)\gamma'}$

1.2. Modelos de programación entera

1.2.1. Optimización de carteras con niveles mínimos de transación y con cota superior en el número de variables positivas

Cuando resolvemos el modelo clásico de Markowitz, la cartera óptima, a menudo, contiene posiciones x_i que son demasiado pequeñas para ejecutar. En la práctica, uno tomaría una solución de (1.1) con la resticción adicional:

$$\{\ x_j > 0 \quad \Longrightarrow \quad x_j \geq l_j\ \} \tag{1.3}$$

donde l_j son los niveles mínimos de transación. Esta restricción establece que, si se realiza una inversión en una acción, entonces ésta debe ser "suficientemente grande". Como podemos observar, esta restricción no es una simple restricción lineal, por lo que no puede ser directamente manejada por la programación cuadrática. Este problema es considerado por Bienstock [2]. Él también considera el problema de optimización de cartera donde hay una cota superior en el número de variables positivas, esto es:

$$\{\ x_j > 0 \text{ para, como mucho, } K \text{ distintos } j \quad (j \in \{1, ..., n\}; \quad n \geq K)\ \} \tag{1.4}$$

La restricción 1.3 puede ser fácilmente incorporada dentro del algoritmo de ramificación y acotación de la siguiente forma:

1. Resolvemos el modelo de Markowitz básico (1.1)
2. Supongamos x^* la solución óptima. Si x^* cumple 1.3, entonces x^* es también solución del problema 1.1-1.3 y podemos parar. Si no es así, sea j un índice para el cual x^* no cumple la restricción 1.3, formamos los dos subproblemas siguientes:

$$\left\{ \begin{array}{c} 1{,}1 \\ + \\ x_j = 0 \end{array} \right\} \qquad\qquad \left\{ \begin{array}{c} 1{,}1 \\ + \\ x_j \geq l_j \end{array} \right\}$$

3. Por último verificamos si la solución de estos dos problemas satisfacen 1.3. Si no la cumplen para el índice k, se vuelve a repetir el paso 2. De esta manera se obtiene el árbol del método de ramificación y acotación (Branch and Bound).

Por otro lado, la restricción 1.4 es más complicada de manejar. Asumimos que tenemos una cota superior dada u_j que nos indica hasta cuánto puede ser invertido en la acción j. Vamos a asumir que las restriciones $0 \leq x_j \leq u_j$ son parte de la formulación de 1.1. Entonces, claramente, la restricción 1.4 implica la restricción:

$$\sum_{j=1}^{n} \frac{x_j}{u_j} \leq K \tag{1.5}$$

Añadimos esta nueva restricción al problema 1.1 y resolvemos el problema cuadrático resultante.

Así, sea x^* la solución óptima encontrada. Si x^* satisfacede 1.4, x^* es la solución óptima de 1.1-1.4 y podemos parar. Si no, sea k un índice para el cual $x_k > 0$, formamos los dos subproblemas siguientes:

$$\left\{ \begin{array}{c} 1{,}1 \\ + \\ x_k = 0 \end{array} \right\} \qquad \left\{ \begin{array}{c} 1{,}1 \\ + \\ \sum_{j \neq k} \frac{x_j}{u_j} \leq K - 1 \end{array} \right\}$$

Cuando un conjunto T de variables ha sido ramificado, las restricciones añadidas al modelo básico 1.1 se convierten en

$$\sum_{j \notin T} \frac{x_j}{u_j} \leq K - |T|$$

Como veremos en el capítulo 2 de este trabajo, este último problema puede manejarse introduciendo variables dicotómicas, convirtiéndose así en un problema de programación cuadrática entero-mixto.

1.2.2. Experimentos computacionales

Con el fin de sensibilizarnos con la nueva restricción 1.5, hemos realizado algunos experimentos computacionales con el servidor NEOS. Este servidor, organizado por el Instituto de Wisconsin, nos ofrece un servicio gratuito basado en Internet para resolver problemas de optimización numérica.

En primer lugar, hemos resuelto el MVO (1.1) sin considerar la restricción de igualdad e incluyendo cotas superiores en las variables. Esto es,

$$\begin{array}{lrcll} \min\limits_{x} & \frac{1}{2} x^T Q x & & & \\ \text{s.a.} & \mu^T x & \geq & R & \\ & Cx & \geq & d & \\ & 0 \leq & x_i & \leq u_i & \forall i \in \{1...N\} \end{array} \tag{1.6}$$

Donde hemos considerado

- $N = 4$

- Q una matriz 4x4, simétrica, aleatoria, cuyos elementos están generados con una Uniforme(0,3.5) y sus elementos diagonales están multiplicados por 10. El hecho de multiplicar los elementos de la diagonal por este número no es más que para asegurarnos que la matriz Q sea definida positiva.
- C es una matriz 3x4, aleatoria, cuyos elementos también están generados con una Uniforme(0,3.5)
- d es un vector 3x1 generado de forma aleatoria con una Uniforme (0,3.5)
- R es un escalar aleatorio generado con una Uniforme(0,3.5)
- μ es un vector 1x4 aleatorio generado con una Uniforme(0,3.5)
- u es un vector 4x1 aleatorio generado con una Uniforme(10,20)

Además, hemos fijado como semilla el número 20. El objetivo de fijar esta semilla es que en experimentos repetidos los resultados sean los mismos. De esta forma, usando el solucionador MINOS, nos resuelve el problema en 6 iteraciones con valor objetivo 4.87199 y

$$x = \begin{pmatrix} 0{,}513341 \\ 0{,}153387 \\ 0{,}449663 \\ 0{,}193711 \end{pmatrix}$$

A continuación, vamos a añadirle a este problema la restriccion 1.5.Esto es,

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} x^T Q x \\ \text{s.a.} \quad & \mu^T x \geq R \\ & Cx \geq d \\ & 0 \leq x_i \leq u_i \qquad \forall i \in \{1...N\} \\ & \sum_{j=1}^{N} \frac{x_j}{u_j} \leq K \end{aligned} \tag{1.7}$$

Hemos tomado $K = 3$, $N = 4$ y hemos generado Q, C, d, R, μ y u de la misma forma que hicimos anteriormente. Además, también hemos tomado como semilla el número 20. De esta forma, el solucionador MINOS nos resuelve el problema en 6 iteraciones, con el mismo valor objetivo y las mismas variables que nos devolvió anteriormente, sin que hubiésemos añadido dicha restricción.

$$\text{Objetivo } = 4{,}87199 \qquad x = \begin{pmatrix} 0{,}513341 \\ 0{,}153387 \\ 0{,}449663 \\ 0{,}193711 \end{pmatrix}$$

Nótese que este hecho no ocurre de manera general, ya que estamos acotando aún más la región factible y nos podría dar otros resultados distintos, eso sí, con un valor objetivo menor al anterior.

En efecto, esta solución cumple la restricción 1.5, ya que, fijando la semilla como 20,

$$u = \begin{pmatrix} 15{,}5042 \\ 13{,}7741 \\ 13{,}6673 \\ 14{,}584 \end{pmatrix}$$

lo que implica

$$\frac{0{,}513341}{15{,}5052} + \frac{0{,}153387}{13{,}7741} + \frac{0{,}449663}{13{,}6673} + \frac{0{,}193711}{14{,}584} < 3$$

Sin embargo, la solución no cumple la restricción 1.4:

$$\{\ x_j > 0 \text{ para, como mucho, 3 valores distintos de } j \quad (j \in \{1, ..., 4\})\ \}$$

ya que todas las variables de x son positivas. Este hecho prueba, en efecto, que

$$1{,}5 \not\Longrightarrow 1{,}4$$

lo que hace insuficiente manejar la restricción 1.4 con tan sólo la adición de 1.5.

1.2.3. Otras aplicaciones de programación entera: subastas combinatorias

Sean:

$M = \{1, 2, ..., m\}$ un conjunto de elementos a subastar

$B_j = (S_j, p_j)$ una apuesta, donde $S_j \subseteq M$; $S_j \neq \emptyset$ y p_j el precio ofrecido por S_j.

Supongamos que el subastador ha recibido n ofertas $B_1, ..., B_n$

¿ Cómo debería determinar el subastador al ganador o ganadores, de manera que maximice su ingreso?

Para la resolución de este problema, se toman variables dicotómicas

$$x_j = \begin{cases} 1 & \text{si gana la oferta } j \\ 0 & \text{en caso contrario} \end{cases}$$

y a continuación se considera el problema

$$\begin{array}{lrcll} \max\limits_{x} & \sum\limits_{j=1}^{n} p_j x_j & & & \\ \text{s.a.} & \sum\limits_{j:i\in S_j} x_j & \leq & 1 & i = 1, ..., m \\ & x_j & \in & \{0,1\} & j = 1, ..., n \end{array} \tag{1.8}$$

Estas dos restricciones nos indican que cada item/elemento $i \in M$ es vendido como máximo una sola vez.

Ejemplo 1.7. Supongamos que tenemos cuatro items para vender, y que hemos recibido las siguientes ofertas:

$$\begin{array}{rcl} B_1 & = & (\{1\}, 6) \\ B_2 & = & (\{2\}, 3) \\ B_3 & = & (\{3,4\}, 12) \\ B_4 & = & (\{1,3\}, 12) \\ B_5 & = & (\{2,4\}, 8) \\ B_6 & = & (\{1,3,4\}, 16) \end{array}$$

Los ganadores pueden ser elegidos mediante la resolución de el siguiente programa entero:

$$\begin{aligned} \max_x \quad & 6x_1 + 3x_2 + 12x_3 + 12x_4 + 8x_5 + 16x_6 \\ \text{s.a.} \quad & \begin{aligned} x_1 + x_4 + x_6 &\leq 1 \\ x_2 + x_5 &\leq 1 \\ x_3 + x_4 + x_6 &\leq 1 \\ x_3 + x_5 + x_6 &\leq 1 \\ x_j &\in \{0,1\} \quad j = 1, ..., 6 \end{aligned} \end{aligned} \tag{1.9}$$

En algunas subastas hay varias unidades de cada elemento en venta. Así, definiríamos una apuesta por:

$$B_j = (\lambda_1^j, \lambda_2^j, ..., \lambda_m^j, p_j)$$

donde λ_i^j es el número de unidades del elemento i, y p_j el precio ofrecido. De esta forma, el problema a resolver sería:

$$\begin{aligned} \max_x \quad & \sum_{j=1}^{n} p_j x_j \\ \text{s.a.} \quad & \begin{aligned} \sum_{j:i \in S_j} x_j &\leq u_i \qquad i = 1, ..., m \\ x_j &\in \{0,1\} \quad j = 1, ..., n \end{aligned} \end{aligned} \tag{1.10}$$

Donde u_i es el número de unidades que están en venta del elemento i

1.2.4. Otras aplicaciones de programación entera: El problema de la caja de seguridad

Considera una empresa estadounidense que recibe cheques de todos las provincias. Hay una demora desde que el cheque es sellado (y por lo tanto el cliente ha cumplido con su obligación) hasta que el cheque es abonado en cuenta (y la empresa puede usar el dinero). Con el fin de acelerar este abono en cuenta, las empresas abren oficinas (llamadas cajas de seguridad) en diferentes ciudades para manejar los cheques.

Ejemplo 1.8. Supongamos que recibimos pagos de 4 regiones (Oeste, Medio oeste, Este, y Sur). El valor medio diario de cada región es:

Oeste $\rightarrow$ 300,000 $

Medio Oeste → 120,000 $

Este → 360,000 $

Sur → 180,000 $

Tener abierta una caja de seguridad cuesta 90,000 $ por año. La media de días desde que el dinero se envía hasta que aparece en la cuenta viene dada en la siguiente tabla. ¿Qué cajas de seguridad deberíamos abrir?

Desde	Los Ángeles	Cincinnati	Boston	Houston
Oeste	2	4	6	6
Medio Oeste	4	2	5	5
Este	6	5	2	5
Sur	7	5	6	3

Cuadro 1.1: Tiempos de ingreso en cuenta

Primero debemos calcular las pérdidas de interés para cada posible asignación. Por ejemplo, si el Oeste envía a Boston, entonces como promedio, en cualquier día del año, habrá en proceso 1,800,000 $ (6 x 300,000 $).Asumiendo una tasa de inversión del 10 %, esto corresponde a una pérdida anual de 180,000 $ anuales. De esta misma forma, podemos calcular las pérdidas para las otras posibilidades:

Desde	Los Ángeles	Cincinnati	Boston	Houston
Oeste	60	120	180	180
Medio Oeste	44	24	60	60
Este	216	180	72	180
Sur	126	90	108	54

Cuadro 1.2: Pérdida de intereses (,000)

Sea ahora y_j una variable 0-1 que tomará el valor 1 si la caja de seguridad j está abierta y 0 si no lo está. Sea $x_{i,j}$ otra variabe 0-1 que es 1 si la región i envía a la caja de seguridad j. Nuestro **objetivo** es minimizar nuestros costes anuales. Esto es:

$$\text{mín}\, 60x_{1,1} + 120x_{1,2} + 180x_{1,3} + ... + 54x_{4,4} + 90y_1 + 90y_2 + 90y_3 + 90y_4$$

Un conjunto de **restricciones** es el siguiente:

$$\sum_j x_{i,j} = 1 \text{ para cada } i \text{ (cada región debe ser asignada a una caja de seguridad)}$$

Otro conjunto de **restricciones** es el que refleja que una región puede sólo ser asignada a una caja de seguridad ABIERTA. Así, para la caja de seguridad 1 (la de Los Ángeles), esto puede ser escrito como:

$$x_{1,1} + x_{2,1} + x_{3,1} + x_{4,1} \leq 100y_1$$

Hemos elegido el número 100, pero podríamos haber elegido cualquier número mayor o igual que 4. De esta forma, si suponemos que la caja de seguridad de Los Ángeles no está abierta, entonces y_1 será 0, y por lo tanto

$$x_{1,1} + x_{2,1} + x_{3,1} + x_{4,1} \leq 0 \implies x_{1,1} = x_{2,1} = x_{3,1} = x_{4,1} = 0$$

Por otro lado, si $y_1 = 1$ no hay ninguna restricción sobre los valores $x_{i,1}$.
Nótese además, que si tomamos como coeficiente de la y_1 un número menor que cuatro, estaríamos imponiendo que al menos una de las variables $x_{i,1}$ fuese igual a cero, lo cual no tendría ningún sentido.

Concretamente, para este problema, tendríamos 20 variables (4 variables y y 16 variables x), y 8 restricciones:

$$\begin{aligned}
\text{mín} \quad & 60x_{1,1} + 120x_{1,2} + ... + 54x_{4,4} + 90y_1 + 90y_2 + 90y_3 + 90y_4 \\
\text{s.a.} \quad & x_{1,1} + x_{1,2} + x_{1,3} + x_{1,4} = 1 \\
& x_{2,1} + x_{2,2} + x_{2,3} + x_{2,4} = 1 \\
& x_{3,1} + x_{3,2} + x_{3,3} + x_{3,4} = 1 \\
& x_{4,1} + x_{4,2} + x_{4,3} + x_{4,4} = 1 \\
& x_{1,1} + x_{2,1} + x_{3,1} + x_{4,1} - 100y_1 \leq 0 \\
& x_{1,2} + x_{2,2} + x_{3,2} + x_{4,2} - 100y_2 \leq 0 \\
& x_{1,3} + x_{2,3} + x_{3,3} + x_{4,3} - 100y_3 \leq 0 \\
& x_{1,4} + x_{2,4} + x_{3,4} + x_{4,4} - 100y_4 \leq 0 \\
& x_{i,j}, y_j \in \{0,1\} \qquad \forall i,j \in \{1,2,3,4\}
\end{aligned} \tag{1.11}$$

Esta es una formulación de programación 0-1 del problema de la caja de seguridad. Cabe notar que no es sólo la única formulación posible. Por ejemplo, podemos obtener otra distinta si consideramos las dieciseis restricciones de la forma

$$x_{i,j} \leq y_j$$

Estas restricciones, también obligan a que una región pueda sólo ser asignada a una caja de seguridad ABIERTA.

Capítulo 2

Algoritmo para la resolución de problemas de programación cuadrática entera-mixta (I)

En este capítulo vamos a presentar un algoritmo de ramificación y corte para resolver problemas de programación cuadrática, en los que hay una cota superior en el número de variables positivas. Concretamente, estamos interesados en la optimización de problemas de programación cuadrática entera mixta (QMIP) de la forma:

$$\begin{aligned} \min_{x} \quad & \frac{1}{2}x^T Q x + c^T x \\ \text{s.a.} \quad & \\ & Ax \leq b && (2.1) \\ & |supp(x)| \leq K && (2.2) \\ & 0 \leq x_j \leq u_j \qquad \forall j && (2.3) \end{aligned}$$

donde x es un n-vector, Q es una matriz simétrica, semidefinida positiva, $supp(x) = \{j : x_j > 0\}$ y $K \in \mathbb{Z}^+$.Un ejemplo de problema de este tipo es el de optimización de carteras que vimos en el capítulo anterior, con restricciones sobre el soporte (1.4), y niveles mínimos de transacción (1.3).Como ya dijimos anteriormente, las variables en el problema corresponden a productos para ser comprados; la función objetivo es una medida de riesgo la cual queremos minimizar; y las restricciones 2.1 y 2.2 indican los niveles de rendimiento y el número máximo de productos distintos que podemos elegir, respectivamente.
Anteriormente vimos cómo resolver la versión del continuo de este problema (esto es, sin la restricción 2.2). Usando esta restricción, la región factible se convierte en un

conjunto entero mixto.

2.1. Formulación y Ramificación

Una posible aproximación para resolver el QMIP es añadir variables-$\{0,1\}$; z_j $j = 1, ..., n$, de forma que:

$$z_j = \begin{cases} 1 & \text{si } x_j > 0 \\ 0 & \text{c. c.} \end{cases}$$

De forma que podamos suprimir las restricciones 2.2 y 2.3 y añadir

$$0 \leq x_j \leq u_j z_j \qquad \forall j \tag{2.4}$$

$$\sum_j z_j \leq K \tag{2.5}$$

$$z_j \in \{0,1\} \tag{2.6}$$

Para resolver el problema, estudiariamos la estructura poliédrica del conjunto entero mixto resultante. Sin embargo, al añadir n variables dicotómicas no sólo estamos duplicando el número de variables, sino que estamos aumentando drásticamente el rango de la matriz de restricciones, ya que el problema quedaría:

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^T Q x + c^T x \\ \text{s.a.} \quad & A^* \begin{pmatrix} x \\ z \end{pmatrix} \leq b^* \\ & 0 \leq x_j \leq u_j z_j \qquad \forall j \quad z_j \in \mathbb{Z}^+ \end{aligned}$$

donde

$$A^* \begin{pmatrix} x \\ z \end{pmatrix} = \begin{pmatrix} & & & 0 & \dots & 0 \\ & A_{nxn} & & \vdots & \ddots & \vdots \\ & & & 0 & \dots & 0 \\ 0 & \dots & 0 & & & \\ \vdots & \ddots & \vdots & & I_{nxn} & \\ 0 & \dots & 0 & & & \\ 0 & \dots & 0 & 1 & \dots & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \\ z_1 \\ \vdots \\ z_n \end{pmatrix} \leq \begin{pmatrix} b_1 \\ \vdots \\ b_n \\ 1 \\ \vdots \\ 1 \\ K \end{pmatrix} = b^*$$

Al aumentar drásticamente el rango de la matriz de restricciones, los programas cuadráticos se vuelven mucho más difíciles. Este hecho lo vemos en los experimentos computacionales que se adjuntan al final de este capítulo. Vale la pena señalar que los programas cuadráticos que surgen directamente de QMIP cuando tienen bajo rango cuadrático (rango de Q). Usando un código apropiado, requiere en el peor caso tan sólo unos segundos de cálculo.Sin embargo, este hecho se va complicando a medida que el rango de Q aumenta.

Como ya indicamos en el capítulo anterior, y con el fin de preservar la rápida resolución de los programas cuadráticos, aunque a costa de una mayor ramificación, podemos modelar la restricción 2.2 usando la restricción "sustituta":

$$\sum_j \frac{x_j}{u_j} \leq K \tag{2.7}$$

que es claramente válida, pero no suficiente. Por lo que es necesario recurrir a un algoritmo de ramificación

La Ramificación se puede realizar del siguiente modo:

En un nodo dado, suponemos F el conjunto de los índices de las variables que no han sido ramificadas, y sea t el número de variables que ya han sido incluidas en el soporte. Sea $h \in F$ de forma que $h \notin supp(x)$ en este nodo, tenemos dos posibles ramas.

1. $h \notin supp(x)$
   En este caso la restricción "sustituta" del nodo hijo se mantiene como

$$\sum_{j \in F} \frac{x_j}{u_j} \leq K - t \tag{2.8}$$

2. $h \in supp(x)$

En este caso la restricción "sustituta" del nodo hijo se convertirá en:

$$\sum_{j \in F-h} \frac{x_j}{u_j} \leq K - t - 1 \tag{2.9}$$

2.2. Planos de corte

A continuación, presentamos una serie de planos de corte que simplifican la resolución de nuestro problema, sobre todo cuando hablamos de dimensiones altas. Vamos a introducir **cuatro tipos** de planos de corte:

1. *Cortes de redondeo entero-mixto*
2. *Cortes de mochila*
3. *Cortes de intersección*
4. *Cortes disyuntivos*

A continuación, vamos a proceder a la explicación de cada uno de estos cortes.

2.2.1. Cortes de redondeo entero-mixto

El procedimiento de redondeo entero-mixto (MIR) es un método general propuesto para la generación de desigualdades válidas, para los programas de problemas entero-mixtos. En términos generales, se trata de tomar una combinación lineal no negativa de desigualdades válidas y fortalecerla esencialmente redondeando los coeficientes de variables enteras.
Para el problema QMIP, como explicamos antes, añadimos a la formulación {0,1}-variables z_j, que indican cuándo x_j es positivo, y las desigualdades 2.4, 2.5 y 2.6. Tras esta incorporación, usamos el procedimiento MIR para generar corte, y luego proyectamos estos cortes en el x-espacio ($\mathbb{R}^n$). La dificultad radica en cómo elegir automáticamente los multiplicadores de la combinación lineal, porque las restricciones con las que tratamos no tienen una estructura particular. Como ejemplo, consideramos el sistema:

$$
\begin{aligned}
2x_1 + 3x_2 + 4x_3 &\geq 1 \\
4x_1 + 2x_2 + 5x_3 &\geq 1 \\
6x_1 + 3x_2 + 2x_3 &\geq 1 \\
|supp(x)| &\leq 2 \\
0 \leq x_j \leq 1 \qquad &\forall j
\end{aligned}
$$

Al añadir a la formulación {0,1}-variables z_j y las desigualdades 2.4, 2.5 y 2.6 el sistema quedaría de la siguiente forma:

$$
\begin{aligned}
2x_1 + 3x_2 + 4x_3 &\geq 1 && (2.10)\\
4x_1 + 2x_2 + 5x_3 &\geq 1 && (2.11)\\
6x_1 + 3x_2 + 2x_3 &\geq 1 && (2.12)\\
0 \leq x_1 &\leq z_1 && (2.13)\\
0 \leq x_2 &\leq z_2 && (2.14)\\
0 \leq x_3 &\leq z_3 && (2.15)\\
z_1 + z_2 + z_3 &\leq 2 && (2.16)\\
z_1, z_2, z_3 &\in \{0,1\} && (2.17)
\end{aligned}
$$

De la restricción 2.10 y 2.17 tenemos que:

$$
\left.\begin{array}{l}
\text{Si } z_3 = 0 \implies x_3 = 0 \overset{(2,17)+(3x_3=z_3=0)}{\implies} 2x_1 + 3x_2 + z_3 \geq 1 \\
\text{Si } z_3 = 1 \implies 2x_1 + 3x_2 + z_3 \geq 1
\end{array}\right\} \implies 2x_1 + 3x_2 + z_3 \geq 1
$$

Aplicando un razonamiento similar con las restricciones 2.11 y 2.17, obtenemos que :

$$
\left.\begin{array}{l}
\text{Si } z_3 = 0 \implies x_3 = 0 \overset{(2,17)+(5x_3=z_3=0)}{\implies} 4x_1 + 2x_2 + z_3 \geq 1 \\
\text{Si } z_3 = 1 \implies 4x_1 + 2x_2 + z_3 \geq 1
\end{array}\right\} \implies 4x_1 + 2x_2 + z_3 \geq 1
$$

Multiplicando estas dos desigualdades por $\frac{1}{2}$ y sumándolas:

$$
\begin{array}{l}
x_1 + \frac{3}{2}x_2 + \frac{1}{2}z_3 \geq \frac{1}{2} \\
\qquad\qquad\qquad\qquad + \\
2x_1 + x_2 + \frac{1}{2}z_3 \geq \frac{1}{2} \\
\hline
3x_1 + \frac{5}{2}x_2 + z_3 \geq 1
\end{array}
$$

Obteniendo la desigualdad: $\boxed{3x_1 + \frac{5}{2}x_2 \geq 1 - z_3}$

De manera similar,

$$\left.\begin{array}{lll} 2{,}11 + 2{,}17 & \Longrightarrow & z_1 + 2x_2 + 5x_3 \geq 1 \\ 2{,}12 + 2{,}17 & \Longrightarrow & z_1 + 3x_2 + 2x_3 \geq 1 \end{array}\right\} \overset{*\frac{1}{2} \text{ y sumo}}{\Longrightarrow} z_1 + \tfrac{5}{2}x_2 + \tfrac{7}{2}x_3 \geq 1$$

Obteniendo la desigualdad: $\boxed{\frac{5}{2}x_2 + \frac{7}{2}x_3 \geq 1 - z_1}$

Por último,

$$\left.\begin{array}{lll} 2{,}10 + 2{,}17 & \Longrightarrow & 2x_1 + z_2 + 4x_3 \geq 1 \quad (A) \\ 2{,}12 + 2{,}17 & \Longrightarrow & 6x_1 + z_2 + 2x_3 \geq 1 \quad (B) \end{array}\right\} \overset{\frac{3}{4}(A)+\frac{1}{4}(B)}{\Longrightarrow} 3x_1 + z_2 + \tfrac{7}{2}x_3 \geq 1$$

Obteniendo la desigualdad: $\boxed{3x_1 + \frac{7}{2}x_3 \geq 1 - z_2}$

Además, el hecho de que al menos uno de los z_j (y su correspondiente x_j deba ser 0, (para que se de la desigualdad 2.16), tenemos:

- Si $z_1 = 0$ $(x_1 = 0) \longrightarrow \frac{5}{2}x_2 + \frac{7}{2}x_3 \geq 1 \overset{x_1=0}{\Longrightarrow} 3x_1 + \frac{5}{2}x_2 + \frac{7}{2}x_3 \geq 1$
- Si $z_2 = 0$ $(x_2 = 0) \longrightarrow 3x_1 + \frac{7}{2}x_3 \geq 1 \overset{x_2=0}{\Longrightarrow} 3x_1 + \frac{5}{2}x_2 + \frac{7}{2}x_3 \geq 1$
- Si $z_3 = 0$ $(x_3 = 0) \longrightarrow 3x_1 + \frac{5}{2}x_2 \geq 1 \overset{x_3=0}{\Longrightarrow} 3x_1 + \frac{5}{2}x_2 + \frac{7}{2}x_3 \geq 1$

Luego,

$$\boxed{\boxed{3x_1 + \frac{5}{2}x_2 + \frac{7}{2}x_3 \geq 1}}$$

es una desigualdad válida que se puede probar que junto con las restricciones

$$\begin{array}{l} 2x_1 + 3x_2 + 4x_3 \geq 1 \\ 4x_1 + 2x_2 + 5x_3 \geq 1 \\ 6x_1 + 3x_2 + 2x_3 \geq 1 \\ 0 \leq x_j \leq 1 \qquad j = 1, 2, 3 \end{array}$$

definen la proyección al x-espacio de la envolvente convexa de puntos entero-mixtos factibles. El enfoque del ejemplo puede ser generalizado para aislar un punto extremo con $K + 1$ variables positivas donde como mucho puede haber K. En cualquier caso, como muestra el ejemplo, una buena elección de los multiplicadores es crítica. Cómo encontrarlos eficientemente es un problema para una investigación más a fondo.

2.2.2. Cortes de mochila

Una estrategia clásica para resolver programas enteros puros consiste en el fortalecimiento de restricciones individuales a través del uso de desigualdades para el problema de la {0,1}-mochila.

Recordatorio 2.1 (Problema de la {0,1}-mochila)**.**
Supongamos que tenemos n ítems distintos. Cada ítem i tiene un beneficio asociado, dado por v_i, y un peso (o volumen) w_i. Usualmente se asume que el beneficio y el peso no son negativos. Para simplificar la representación, se suele asumir que los ítems están listados en orden creciente según el peso (o volumen).

Por otro lado se tiene una mochila, donde se pueden introducir los ítems, que soporta un peso máximo (o volumen máximo) W.

El problema consiste en meter en la mochila ítems de tal forma que se maximice el valor de los ítems que contiene y siempre que no se supere el peso máximo que puede soportar la misma. La solución al problema vendrá dada por la secuencia de variables $x_1, x_2, ..., x_n$ donde el valor de x_i indica si el ítem i es seleccionado.

El problema se puede expresar matemáticamente por medio del siguiente programa lineal:

$$\begin{aligned} \max_{x} \quad & \sum_{i=1}^{n} v_i x_i \\ \text{s.a.} \quad & \sum_{i=1}^{n} w_i x_i \leq W \\ & x_i \in \{0,1\} \quad i = 1, ..., n \end{aligned}$$

A continuación, vamos a discutir sobre el fortalecimiento de una única desigualdad lineal, dada cotas superiores de las variables y una cota superior del número de variables positivas. Así, vamos a considerar el siguiente sistema (P):

$$\begin{aligned} \sum_{j \in \mathbb{N}} a_j x_j &\geq \beta \\ |supp(x)| &\leq K \\ 0 \leq x_j &\leq u_j \quad \forall j \end{aligned}$$

Donde $a_j > 0 \quad \forall j \in \mathbb{N}$. Para simplificar la notación, asumimos

$$u_j = 1 \quad \forall j$$

ya que podemos realizar el cambio de variable $x_j \to \frac{x_j}{u_j}$ que no altera la estructura del problema. De esta forma, el sistema (P) quedaría como sigue:

$$\sum_{j\in\mathbb{N}} a_j x_j \geq \beta \tag{2.18}$$

$$|supp(x)| \leq K \tag{2.19}$$

$$0 \leq x_j \leq 1 \quad \forall j \tag{2.20}$$

Definición 2.2 (Conjunto crítico de índices para el sistema (P)). *Dado un subconjunto de índices $C \subseteq \mathbb{N}$, diremos que este conjuto es* crítico *si tiene la siguiente propiedad:*

$$\forall J \subset C, \ con\ |J| = K \implies \sum_{j\in J} a_j < \beta$$

Proposición 2.3. *Si C es crítico para nuestro sistema (P), debemos de tener como mucho $K-1$ variables de C que puedan ser positivas. En otras palabras,*

$$\sum_{j\in C} x_j \leq K-1 \tag{2.21}$$

es una desigualdad válida para nuestro sistema (P)

Demostración.

Vamos a demostrarlo para el caso de C crítico con K variables de C positivas (de forma análoga se tiene si el número de variables positivas es $> K$).

Por un lado, si tomamos $J = \{i \in C : 0 < x_i \leq 1\} \implies |J| = K \overset{\text{C es crítico}}{\implies} \sum_{j\in J} a_j < \beta \overset{x_j\leq 1}{\implies} \sum_{j\in J} a_j x_j \leq \sum_{j\in J} a_j < \beta \implies \sum_{j\in J} a_j x_j < \beta$

Pero, si C crítico con K variables de C positivas $\overset{J\subset C\subseteq\mathbb{N}+2,19+2,18}{\implies} \sum_{j\in J} a_j x_j = \sum_{j\in\mathbb{N}} a_j x_j \geq \beta \implies \sum_{j\in J} a_j x_j \geq \beta$ lo que nos lleva a una contradicción □

Ejemplo 2.4. Supongamos $a = \begin{pmatrix} 8 \\ 7 \\ 6 \\ 4 \\ 12 \\ 12 \end{pmatrix}$, $\beta = 22$ y $K = 3$. El sistema (P) quedaría

de la siguiente forma:

$$8x_1 + 7x_2 + 6x_3 + 4x_4 + 12x_5 + 12x_6 \geq 22$$

$$|supp(x)| \leq 3$$

$$0 \leq x_j \leq 1 \qquad j = 1, 2, ..., 6$$

Sea $C = \{1, 2, 3, 4\}$, tenemos que C es un conjunto crítico, es decir,

$$\forall J \subset C, \text{ con } |J| = 3 \implies \sum_{j \in J} a_j < 22$$

pues

$$\begin{aligned} a_1 + a_2 + a_3 &= 8 + 7 + 6 = 21 < 22 \\ a_1 + a_2 + a_4 &= 8 + 7 + 4 = 19 < 22 \\ a_1 + a_3 + a_4 &= 8 + 6 + 4 = 18 < 22 \\ a_2 + a_3 + a_4 &= 7 + 6 + 4 = 17 < 22 \end{aligned}$$

Consideramos por otro lado el punto $v = \begin{pmatrix} 1 \\ 1 \\ \frac{1}{2} \\ \frac{1}{4} \\ \frac{1}{4} \\ 0 \end{pmatrix}$, el cual satisface la restricción sustituta $\sum_j x_j \leq K$ con igualdad, pues $1 + 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + 0 = 3$. Sin embargo, v no cumple la desigualdad $\sum_{j \in C} x_j \leq K - 1$, pues $v_1 + v_2 + v_3 + v_4 = 1 + 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} = 3 \not\leq 2$

Definición 2.5. *Sea $S \subseteq \mathbb{R}^n$ la envolvente convexa de las soluciones factibles de un problema (P), diremos que la desigualdad $cx \leq d$ es un facet-defining para S si $S \cap \{x \in \mathbb{R}^n : cx = d\}$ es una cara de S*

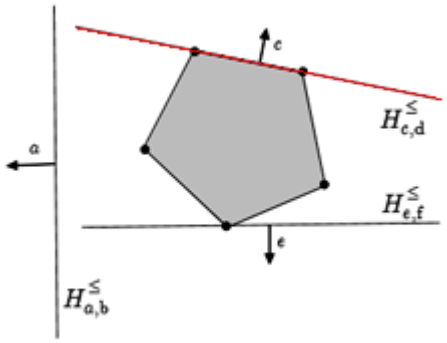
Ejemplo 2.6. Supongamos que estamos en $\mathbb{R}^2$ y S es el recinto gris del dibujo. Tenemos tres desigualdades:

$$H^{\leq}_{a,b} = \{x \in \mathbb{R}^2 : ax \leq b\}$$

$$H^{\leq}_{c,d} = \{x \in \mathbb{R}^2 : cx \leq d\}$$

$$H^{\leq}_{e,f} = \{x \in \mathbb{R}^2 : ex \leq f\}$$

De las tres, sólo $H^{\leq}_{c,d}$ es facet-defining de politopo S.



En el resto de la subsección, discutiremos cuándo 2.21 es un facet-defining para la envolvente convexa de las soluciones factibles de (P), y cómo modificarla en tiempo polinomial para obtener una cara cuando 2.21 no lo es.
Para simplificar la notación, suponemos $C = \{1, 2, ...|C|\}$ con $a_1 \geq a_2 \geq ... \geq a_{|C|}$; y que $a_{|C|+1} = \text{máx}\{a_j : j > |C|\}$

Teorema 2.7. *Suponemos que el sistema (P) es factible. La desigualdad 2.21 es facet-defining si se satisfacen una combinación de las siguientes condiciones:*

1. *C es un conjunto maximal para ser crítico y $|C| \geq K$*
2. $a_{|C|+1} + \sum_{j=2}^{K} a_j \geq \beta$
3. $a_{|C|+1} + \sum_{j=1}^{K-2} a_j + a_{|C|} \geq \beta$
4. $K > |C| + 1$, *o* $a_{|C|+1} + \sum_{j=2}^{K-1} a_j \geq \beta$

Demostración.
Sea S la envolvente convexa de las soluciones factibles de (P):

$$S = \left\{ \sum_{j \in \mathbb{N}} a_j x_j \geq \beta, \quad 0 \leq x_j \leq 1, \quad |supp(x)| \leq K \right\}$$

Consideramos que estamos en $\mathbb{R}^n$ ($x \in \mathbb{R}^n$), $dim(S) = n$. Como la dimensión de S es completa (i.e. S no está contenida en ningún hiperplano), para probar que 2.21 es un facet-defining, basta con encontrar n puntos factibles, con coordenadas 0 y 1, que

sean afinmente independientes (que es lo mismo que buscar $n-1$ puntos linealmente independientes) y que verifiquen la desigualdad 2.21 con igualdad $\left(\sum_{j\in C} x_j = K-1\right)$

Para obtener estos puntos, vamos a realizar modificaciones sobre el conjunto $C \to C'$ de forma que $x^{C'} \in S$. Donde las componentes de $x^{C'}$ son

$$x_j^{C'} = \begin{cases} 1 & \text{Si } j \in C' \\ 0 & \text{c.c.} \end{cases}$$

Sea $C = \{1, 2, ..., |C|\}$

CASO 1: $|C| \geq K$

- Si se cumple 2, quiere decir que si le quito al conjunto C el conjunto $\{1, K+1, ..., |C|\}$ y le añado el índice $|C|+1$, es decir,

  $$I_1^{'} = (C - [\{1\} \cup \{K+1, ...|C|\}]) \cup \{|C|+1\}$$

  se tiene que $x^{I_1^{'}}$ sigue siendo un punto factible del problema (P) ($x^{I_1^{'}} \in S$).

  Así, como $a_1 > ... > a_K > a_{K+1} > ... > a_{|C|} > a_{|C|+1}$

  $$I_i^{'} = (C - [\{i\} \cup \{K+1, ...|C|\}]) \cup \{|C|+1\} \implies x^{I_i^{'}} \in S \qquad \forall i \in \{1, ..., K\}$$

  Además, se cumple que

  $$\sum_{j \in I_i^{'} \cap C} x_j^{I_i^{'}} = K-1$$

  TENEMOS **K** PUNTOS DE ESTE TIPO:

$$\begin{array}{cccccccccccccc}
 & & & & \text{K} & \text{K+1} & & & \text{|C|} & \text{|C|+1} & & & & \\
(0, & 1, & 1, & ..., & 1, & 0, & 0, & ..., & 0, & 1, & 1, & 0, & ..., & 0) \\
(1, & 0, & 1, & ..., & 1, & 0, & 0, & ..., & 0, & 1, & 1, & 0, & ..., & 0) \\
 & & & \vdots & & & & \vdots & & & & \vdots & & \\
(1, & 1, & 0, & ..., & 1, & 0, & 0, & ..., & 0, & 1, & 1, & 0, & ..., & 0)
\end{array}$$

- Si se cumple 3, quiere decir que si le quito al conjunto C los índices $\{K-1, K\} \cup \{K+1, ..., |C|\}$ y le añado los índices $\{|C|, |C|+1\}$, es decir,

$$I''_{|C|} = (C - [\{K-1, K\} \cup \{K+1, ...|C|\}]) \cup \{|C|, |C|+1\}$$

se tiene que $x^{I''_{|C|}}$ sigue siendo un punto factible del problema (P) ($x^{I''_{|C|}} \in S$).

Así, como $a_1 > ... > \underbrace{a_{K-1} > a_K > a_{K+1} > ... > a_{|C|-1}}_{\text{quito}} > \overbrace{a_{|C|} > a_{|C|+1}}^{\text{añado}}$

$$I''_{|C|-1} = (C - [\{K-1, K\} \cup \{K+1, ...|C|\}]) \cup \{|C|-1, |C|+1\} \implies x^{I''_{|C|-1}} \in S$$

$$\vdots$$

$$I''_{|C|-j} = (C - [\{K-1, K\} \cup \{K+1, ...|C|\}]) \cup \{|C|-j, |C|+1\} \implies x^{I''_{|C|-j}} \in S$$
$$\forall j \in \{0, 1, ..., |C| - (K+1)\}$$

Además, se cumple que

$$\sum_{j \in I''_{|C|-j} \cap C} x_j^{I''_{|C|-j}} = K - 1$$

TENEMOS **|C|-(K+1)+1=|C|-K** PUNTOS DE ESTE TIPO:

			K-2	K-1	K	K+1		|C|-1	|C|	|C|+1		
(1,	1,	...,	1	0,	0,	0,	...,	0,	1,	1,	0,	...	0)
(1,	1,	...,	1	0,	0,	0,	...,	1,	0,	1,	0,	...	0)
			⋮				⋮					⋮	
(1,	1,	...,	1	0,	0,	1,	...,	0,	0,	1,	0,	...	0)

- El resto de los puntos los tomamos:

			K-1	K		|C|+1				n
(1,	1,	...,	1	0,	...,	0,	1,	0,	...,	0)
(1,	1,	...,	1	0,	...,	0,	0,	1,	...,	0)
			⋮						⋮	
(1,	1,	...,	1	0,	...,	0,	0,	0,	...,	1)

TENEMOS **n-(|C|+1)=n-|C|-1** PUNTOS DE ESTE TIPO:

Además todos cumplen que $\sum x_j = K - 1$.
Por el apartado 1, C es maximal para ser crítico. Este hecho implica que todos estos últimos puntos sean factibles (i.e. $\in S$)

Así, al unir todos los puntos anteriores, tenemos $K + |C| - K + n - |C| - 1 = n - 1$ puntos que ademas, por construcción son linealmente independientes.Luego 2.21 es facet-defining.

De forma análoga y usando distintos apartados en cada caso, se demuestra el teorema para los casos:

CASO 2: $|C| < K - 1$ y CASO 3: $|C| = K - 1$

□

A continuación, vamos a describir una desigualdad que es facet-defining cuando las condiciones 1 y 2 se cumplen, pero no se cumple la condición 3. Definimos:

$$Z = \{h \in C : a_{|C|+1} + \sum_{j=1}^{K-2} a_j + a_h < \beta\}$$

- Asumiendo que se da la condición 2 $\implies a_{|C|+1} + \sum_{j=2}^{K} a_j = a_{|C|+1} + \sum_{j=2}^{K-2} + a_{K-1} + a_K \geq \beta \overset{a_1>\ldots>a_{K-1}>a_K>a_{K+1}>\ldots>a_{|C|}>a_{|C|+1}}{\implies} a_{|C|+1} + \sum_{j=2}^{K-2} + a_1 + a_i \geq \beta \quad \forall 1 \leq i \leq K \implies a_{|C|+1} + \sum_{j=1}^{K-2} + a_i \geq \beta \quad \forall 1 \leq i \leq K \implies \mathbf{j > K} \quad \forall \mathbf{j \in Z}$

- Si $j \in Z \overset{j \in C}{\implies} \mathbf{j \leq |C|} \quad \forall \mathbf{j \in Z}$

Luego podemos concluir con que Z es de la forma $\{t \leq j \leq |C|, \text{ para algún } t > K\}$.

Por otro lado, para cualquier x factible para el sistema (P) que satisfaga 2.21 con igualdad, tenemos que $x_j = 0, \quad \forall j \in Z$. En consecuencia,

$$\sum_{j=1}^{t-1} x_j \leq K - 1$$

es facet-defining para el poliedro

$$Conv\{x : x \text{ factible para (P) y } x_j = 0 \quad \forall j \in Z\}$$

Así, podemos buscar desigualdades facet-defining de la forma:

$$\sum_{j=1}^{t-1} x_j + \sum_{j \in Z} \mu_j x_j \leq K - 1$$

con $\mu_j > 0$.

Una forma de obtener tal desigualdad, es usar el método de elevación. Asumiendo que tenemos calculados μ_j para $j \in W \subset Z$, entonces para un cierto $h \in Z - W$, μ_h es el valor más grande tal que:

$$\sum_{j \in C-Z} x_j + \sum_{j \in W} \mu_j x_j + \mu_h x_h \leq K - 1$$

Esta desigualdad no excluye ninguna x factible con $supp(x) \cap Z = \{j : x_j > 0\} \cup Z \subset W \cap h$. Se puede probar que para el cálculo de los μ_j podemos usar una simple recursión de programación dinámica (en tiempo polinomial). Veámoslo con un ejemplo:

Ejemplo 2.8. Supongamos $a = \begin{pmatrix} 8 \\ 7 \\ 6 \\ 2 \\ 1 \\ 11 \\ 10 \end{pmatrix}$, $\beta = 22$ y $K = 3$. El sistema (P) quedaría de la siguiente forma:

$$8x_1 + 7x_2 + 6x_3 + 2x_4 + x_5 + 11x_6 + 10x_7 \geq 22$$

$$|supp(x)| \leq 3$$

$$0 \leq x_j \leq 1 \qquad j = 1, 2, ..., 7$$

Sea $C = \{1, 2, 3, 4, 5\}$, tenemos que C es un conjunto crítico, es decir,

$$\forall J \subset C, \text{ con } |J| = 3 \implies \sum_{j \in J} a_j < 22$$

pues

$$\begin{aligned}
a_1 + a_2 + a_3 &= 8 + 7 + 6 = 21 < 22 \\
a_1 + a_2 + a_4 &= 8 + 7 + 2 = 17 < 22 \\
a_1 + a_2 + a_5 &= 8 + 7 + 1 = 16 < 22 \\
a_1 + a_3 + a_4 &= 8 + 6 + 2 = 16 < 22 \\
a_1 + a_3 + a_5 &= 8 + 6 + 1 = 15 < 22 \\
a_1 + a_4 + a_5 &= 8 + 2 + 1 = 11 < 22 \\
a_2 + a_3 + a_4 &= 7 + 6 + 2 = 15 < 22 \\
a_2 + a_3 + a_5 &= 7 + 6 + 1 = 14 < 22 \\
a_2 + a_4 + a_5 &= 7 + 2 + 1 = 10 < 22 \\
a_3 + a_4 + a_5 &= 6 + 2 + 1 = 9 < 22
\end{aligned}$$

Sea $Z = \{4, 5\}$ ($Z = \{t \leq j \leq 5, \text{ para algún } t > 3\}$). La correspondiente desigualdad facet-defining es:

$$\sum_{j=1}^{4-1} x_j + \sum_{j=4}^{5} \mu_j x_j = x_1 + x_2 + x_3 + \mu_4 x_4 + \mu_5 x_5 \leq 3 - 1 = 2$$

donde:
μ_4 es el valor más grande tal que, $x_1 + x_2 + x_3 + \mu_4 x_4 \leq 2 \implies \mu_4 = 2$
μ_5 es el valor más grande tal que, $x_1 + x_2 + x_3 + 2x_4 + \mu_5 x_5 \leq 2 \implies \mu_5 = 2$

Luego la desigualdad facet-defining será

$$x_1 + x_2 + x_3 + 2x_4 + 2x_5 \leq 2$$

2.2.3. Cortes de intersección

Para la descripción de los cortes de intersección, consideramos cualquier formulación válida $Ax \leq b$ para problemas de programación cuadrátia entera mixta (QMIP):

$$\begin{aligned}
\min_{x} \quad & x^T Q x + c^T x \\
\text{s.a.} \quad & \\
& Ax \leq b \\
& |supp(x)| \leq K \\
& 0 \leq x_j \leq u_j \qquad \forall j
\end{aligned}$$

Sea x^* un punto extremo y sea v_i, $1 \leq i \leq s$ sus vértices vecinos. Si x^* tiene más de K variables positivas, entonces todo punto de

$$Conv\left(x^* \cup \{v_1, ..., v_s\}\right) - Conv\left(\{v_1, ..., v_s\}\right)$$

tendrá igualmente más de K variables positivas. Como resultado, cualquier desigualdad que corte/separe a x^* pero no separe a ninguno de los v_i vecinos, es válida para QMIP. Para implementar esta idea, uno puede enumerar los v_i y elegir una desigualdad apropiada. La desigualdad obtenida anteriormente por medio del procedimiento MIR es un corte de intersección.

La motivación para usar cortes de intersección es la siguiente. Normalmente, los programas cuadráticos que surgen en la práctica tienen un valor objetivo no negativo. Esencialmente, como estamos minimizando, estamos buscando un punto en el poliedro $\{x : Ax \leq b, 0 \leq x \leq u\}$ que sea el más cercano al origen con alguna norma. Como resultado, el óptimo será alcanzado por algún punto x^* en el octante positivo (en el caso de $\mathbb{R}^3$. Cuadrante positivo en el caso de $\mathbb{R}^2$... , 'orthant' positivo en el caso de $\mathbb{R}^n$) con más de K coordenadas positivas, de forma que todos los puntos factibles entero-mixto estarán "más lejos" del origen que x^*. La intención de añadir un corte de intersección no es sólo "separar" a x^*, sino también es mejorar la cota más baja del QMIP. Ciertamente, este sería el caso si la normal al corte tiene producto escalar positivo con el gradiente de la forma cuadrática en x^*; esto es, si el ángulo que forman es $< 90^o$ (requisito indispensable para que sea una dirección factible).

Una complicación que surge es que el óptimo para el programa cuadrático no será en general un punto extremo. Incluso si el óptimo está en una cara de baja dimensión, esta cara puede que contenga muchos puntos extremos, y nosotros además debemos enumerar todos sus puntos vecinos. Sin embargo, esto puede ser beneficioso en el sentido de que podremos cortar una cara entera.

2.2.4. Cortes disyuntivos

El procedimiento disyuntivo fue desarrollado por Balas [3] como una propuesta para generar desigualdades válidas para programas entero-mixtos. Recientemente, este procedimiento, ha tenido gran éxito en algoritmos de corte y ramificación.

Para el caso de QMIP, vamos a describir este procedimiento como sigue:
En primer lugar vamos a considerar un nodo del árbol del branch and bound donde t es el número de variables que han sido incluidas en el soporte y consideramos F como el conjunto de los índices de las variables que no se han ramificado, y sea la

formulación actual $Ax \leq b$ junto con la restricción sustituta 2.8, es decir,

$$S = \left\{ x : Ax \leq b, \quad \sum_{j \in F} \frac{x_j}{u_j} \leq K - t \right\}$$

Para $h \in F$, sea $P^0(h)$ el poliedro definido por S añadiendo $x_h = 0$

$$P^0(h) = \left\{ x : Ax \leq b, \quad \sum_{j \in F} \frac{x_j}{u_j} \leq K - t, \quad x_h = 0 \right\}$$

y sea $P^1(h)$ el poliedro definido por $Ax \leq b$ y el fortalecimiento 2.9, es decir,

$$P^1(h) = \left\{ x : Ax \leq b, \quad \sum_{j \in F - h} \frac{x_j}{u_j} \leq K - t - 1 \right\}$$

Nota 2.9. Si estamos en un nodo en el que F es el conjunto de los índices de las variables que no se han ramificado y t el número de variables que han sido incluidas en el soporte, entonces

- $P^0(h)$ corresponde al poliedro del problema resultante en el nodo hijo en el que $h \notin supp(x)$
- $P^1(h)$ corresponde al poliedro del problema resultante en el nodo hijo en el que $h \in supp(x)$

Supongamos ahora que x^* es una solución óptima para el programa cuadrático en este nodo (nodo padre), y $h \in F$ es tal que $0 < x_h^* < u_h$. Si x^* está en la envolvente convexa de las soluciones factibles de QMIP, debe ser el caso que es una combinación convexa de un punto de $P^0(h)$ y otro punto de $P^1(h)$. Esta condición puede ser descrita por un sistema de desigualdades lineales; cuando este sistema no es factible una aplicación del lema de Farkas produce una desigualdad válida para QMIP que es violada por x^*.

Nota 2.10. El procedimiento disyuntivo puede fácilmente manejar restricciones entera-mixtas adicionales para el problema QMIP. Supongamos, por ejemplo, que tenemos niveles mínimos de transación. Esto implica que queremos hacer cumplir (con la notación anterior) $x_h \geq \alpha$ para algún $\alpha > 0$. Esta restricción puede ser añadida directamente a la formulación de $P^1(h)$.

2.2.5. Experimentos computacionales

En este apartado, nuestra pretensión es comparar el problema continuo y el problema al que se le ha añadido una restricción sobre el soporte (problema entero-mixto). Así, en primer lugar, hemos implementado con el lenguaje AMPL ambos problemas, es decir,

$$\text{CONTINUO} \begin{cases} \min\limits_{x} & \frac{1}{2}x^T Q x & & & & \\ \text{s.a.} & \mu^T x & \geq & R & & \\ & Cx & \geq & d & & \\ & 0 \leq & x_i & \leq u_i & \forall i \in \{1...N\} \end{cases}$$

$$\text{ENTERO-MIXTO} \begin{cases} \min\limits_{x} & \frac{1}{2}x^T Q x & & & & \\ \text{s.a.} & \mu^T x & \geq & R & & \\ & Cx & \geq & d & & \\ & z_i & \in & \{0,1\} & \forall i \in \{1...N\} & \\ & 0 \leq & x_i & \leq u_i z_i & \forall i \in \{1...N\} & \\ & \sum\limits_i z_i & \leq & K & & \end{cases}$$

Para ello, hemos considerado:

- Q una matriz NxN simétrica, aleatoria, cuyos elementos están generados con una Uniforme(0,3.5) y sus elementos diagonales están multiplicados por 10.
- C es una matriz 3xN, aleatoria, cuyos elementos también están generados con una Uniforme(0,3.5)
- d es un vector 3x1 generado de forma aleatoria con una Uniforme (0,3.5)
- R es un escalar aleatorio generado con una Uniforme(0,3.5)
- μ es un vector 1xN aleatorio generado con una Uniforme(0,3.5)
- u es un vector Nx1 aleatorio generado con una Uniforme(10,20)

Como hicimos anteriormente, realizaremos estos experimentos con el servidor NEOS. En el caso del continuo, nos centraremos principalmente en la variable que indica la dimensión del problema (N). En el caso del problema entero-mixto, añadiremos además la variable que indica la cota sobre el cardinal del soporte (K), considerando en

este caso dos variables; N y K.
Iremos variando estas variables con el fin de ver cómo evolucionan los experimentos. Cabe notar que en ningún momento estaremos interesados en las soluciones, sino en el tiempo de ejecución que emplee el servidor en resolver ambos problemas, o lo que es lo mismo, en el número de iteraciones que emplee el algoritmo. Para ello, realizaremos 3 experimentos en los que,

- **EXPERIMENTO 1:** No fijaremos ninguna semilla para generar aleatoriamente los datos anteriores.
- **EXPERIMENTO 2:** Fijaremos como semilla el valor 20.
- **EXPERIMENTO 3:** Fijaremos como semilla el valor 3.

De esta forma, para el problema continuo y usando el solucionador MINOS, obtenemos la tabla siguiente:

	$N°$ de iteraciones			
N	**EXP1**	**EXP2**	**EXP3**	Promedio
5	7	8	11	8.6
10	13	12	15	13.3
15	17	21	29	22.3
20	18	20	20	19.3
25	14	40	38	30.6
30	27	19	27	24.3
40	29	22	40	30.3
50	26	22	17	21.6
80	15	36	14	21.6
100	37	35	35	35.6
150	27	17	18	20.6

Cuadro 2.1: $N°$ de iteraciones-MINOS. Modelo CONTINUO.

Como podemos observar a partir de la tabla anterior, en el caso del problema continuo, el servidor resuelve el problema sin ningún inconveniente, sea cual sea su dimensión (llegando a ser esta hasta de 150). Además, lo resuelve en pocas iteraciones que oscilan entre los valores 7 y 40. De esta forma, podemos deducir que el modelo continuo es fácil de resolver y requiere tan sólo unos pocos segundos de cálculo.

Veamos ahora el problema entero mixto. Para ello, vamos a suponer en primer lugar que el soporte está acotado por **la cuarta parte de la dimensión del problema** (el

25 % de la cartera, como mucho, puede tomar valores > 0). De esta forma, y usando el solucionador COUENNE, obtenemos la siguiente tabla:

$K = \frac{N}{4}$	$N°$ de iteraciones			
N	**EXP1**	**EXP2**	**EXP3**	Promedio
5	26	55	4	28.3
10	655	803	861	773
15	1776	10189	2806	4923.6
20	39311	262328	182849	161496
25	208065	10697956	5160081	5355367.3

Cuadro 2.2: $N°$ de iteraciones-COUENNE. Modelo ENTERO-MIXTO. $K = \frac{N}{4}$

El motivo de haber llegado sólo a la dimensión 25, no es más que el solucionador, para $N = 30$, se quedó bloqueado en los tres experimentos. Además, como podemos deducir de la tabla anterior, para la dimensión $N = 5$, el número de iteraciones del algoritmo es relativamente bajo; sin embargo, a medida que la dimensión se va incrementando, éste aumenta drásticamente, y por consiguiente, aumenta el tiempo de ejecución. En este caso, el algoritmo llega a usar hasta 10697956 iteraciones, para $N = 25$, en el experimento 2. El tiempo que el solucionador empleó en este caso fue de 1528,09 segundos, o lo que es lo mismo, 25,46 minutos.

Suponiendo el soporte acotado por la **mitad de la dimensión del problema** (el 50 % de la cartera, como mucho, puede tomar valores > 0), obtenemos los siguientes resultados:

$K = \frac{N}{2}$	$N°$ de iteraciones			
N	**EXP1**	**EXP2**	**EXP3**	Promedio
5	76	172	201	149.6
10	8383	2315	4779	5159
15	292929	753930	128587	391815.3
20	6351814	-	77123351	41737582.5
25	-	-	-	-

Cuadro 2.3: $N°$ de iteraciones-COUENNE. Modelo ENTERO-MIXTO. $K = \frac{N}{2}$

En efecto, para el caso $K = \frac{N}{2}$, y tal y como ocurría en el caso $K = \frac{N}{4}$, el número de iteraciones aumenta radicalmente a medida que se va incrementando la dimensión del problema. Este incremento llega hasta 77123351 iteraciones, para N=20, en el experimento 3. El tiempo que el solucionador empleó en este caso fue de 12270.75

segundos; lo que equivale a 3.45 horas. Mucho más que el tiempo máximo que se empleó en el caso de $K = \frac{N}{2}$ (25,46 minutos), a pesar de que la dimesión era mayor (N=25).
Además, en este caso, el solucionador se quedó bloqueado para N=25 en los tres experimentos, y para N=20 en el experimento 2.

Por último, si suponemos el soporte acotado por las **tres cuartas partes de la dimensión del problema** (el 75 % de la cartera, como mucho, puede tomar valores > 0), obtenemos los siguientes resultados:

$K = \frac{3N}{4}$	$N°$ de iteraciones			
N	**EXP1**	**EXP2**	**EXP3**	Promedio
5	80	339	236	28.3
10	74923	20713	43568	773
15	869906	56670587	115309	4923.6
20	255	-	-	-
25	-	-	-	-

Cuadro 2.4: $N°$ de iteraciones-COUENNE. Modelo ENTERO-MIXTO. $K = \frac{3N}{4}$

Tal y como ocurría en los casos anteriores, el incremento del número de iteraciones sigue aumentando, de forma violenta, a medida que se aumenta la dimensión. En este caso, el solucionador quedó bloqueado para N=25 en los tres experimentos. También quedó bloqueado para N=20 en dos de ellos, dándonos, sorprendentemente, 255 iteraciones en el experimento que no se bloqueó para esta dimensión. Podemos considerar que esta cifra no es significativa, pues en los otros dos experimentos, el solucionador no fue capaz de darnos una solución.
Por otro lado, el número máximo de iteraciones que usó el solucionador, en el caso $K = \frac{3N}{4}$, fue de 56670587, para N=15, en el experimento 2. Sin embargo, en el caso $K = \frac{N}{2}$, la media de iteraciones, para N=15, era 391815.3. Una cifra significativamente más pequeña que el valor 56670587.

Para obtener una visión global de los resultados anteriores, concluimos este apartado con una tabla resumen que recoge todos los promedios de los experimentos anteriores.

N	N° de iteraciones			
	Promedios-Continuo	**Promedios-Ent-mixt** $K = \frac{N}{4}$	**Promedios-Ent-mixt** $K = \frac{N}{2}$	**Promedios-Ent-mixt** $K = \frac{3N}{4}$
5	8.6	28.33	149.6	218.3
10	13.3	773	5159	46401.3
15	22.3	4923.6	391815.3	19218600.67
20	19.3	161496	41737582.5	-
25	30.6	5155367.3	-	-

Cuadro 2.5: TABLA RESUMEN

Así, a partir de esta tabla y de las reflexiones anteriores, podemos deducir dos conclusiones claras:

- Mientras el modelo continuo se resuelve rápidamente, incluso en los casos en los que trabajamos con dimensiones altas, el modelo entero-mixto se complica enormemente a medida que realizamos pequeños incrementos en la dimensión del problema. Este hecho era de esperar, ya que al introducir variables dicotómicas, duplicamos la dimensión del problema, renunciando además a la continuidad del mismo.

- En el caso del problema entero-mixto, para una dimensión fija, el problema se complica a medida que aumentamos la cota superior en el número de variables estrictamente positivas. Este hecho puede deberse a que la región entera-mixta factible aumenta, lo que dificulta la búsqueda del óptimo.

Capítulo 3

Un algoritmo alternativo para la resolución de problemas de programación cuadrátia entera-mixta (II)

En este capítulo vamos a describir otro algoritmo para problemas de optimizacón cuadrática con restricciones de cardinalidad. Tal y como hicimos antes, vamos a proponer un branch and bound basado en algoritmos que se aprovechan de la estructura especial de estos problemas. Usaremos una implementación del branch and bound adaptada con algoritmos pivotantes. Así, como en el capitulo anterior, consideramos el siguiente problema (CCQO):

$$\begin{aligned} \min_{x} \quad & \frac{1}{2}x^T Q x + c^T x \\ \text{s.a.} \quad & \\ & Ax \leq b && (3.1)\\ & |supp(x)| \leq K && (3.2)\\ & x_i \geq \alpha_i \qquad i \in supp(x) && (3.3)\\ & 0 \leq x_j \leq u_j \qquad \forall j && (3.4) \end{aligned}$$

donde Q es una matriz $\mathbb{R}^{dxd}$, simétrica, semidefinida positiva, $c \in \mathbb{R}^d$, $A \in \mathbb{R}^{mxd}, b \in \mathbb{R}^m, \alpha_i > 0$, y $K \in \mathbb{Z}^+$.

El segundo conjunto de restricciones 3.2 (denominado restricciones de cardinalidad), y el tercer conjunto de restricciones (denominado restricciones de la cota infe-

rior), introducen el carácter discreto al problema, haciéndolo un problema de optimización entero mixto cuadrático.

En el capítulo anterior, describíamos un branch-and-bound adaptado que reemplazaba la restricción de cardinalidad 3.2 por la restricción sustituta

$$\sum_i \frac{x_i}{u_i} \leq K$$

Por otra parte, las variables de x eran ramificadas directamente en lugar de introducir variables binarias:

1. La restricción $x_h = 0$ era añadida cuando $h \notin supp(x)$
2. La restricción $x_h \geq \alpha_i$ cuando queríamos que $h \in supp(x)$

En este capítulo vamos a extender el branch and bound, pero usando el algoritmo pivotante de Lemke para resolver los sucesivos sub-problemas en el árbol de corte y ramificación. De forma distinta que el algoritmo anterior, no añadimos explícitamente las restricciones referidas a las cotas de las variables, $x_h = 0$ y $x_h \geq \alpha_i$, así, el tamaño de los subproblemas nunca aumenta. La otra distinción con el algoritmo anterior es que usamos un algoritmo pivotante para resolver los subproblemas.

Antes de describir esta metodología de forma general para resolver CCQO, vamos a dedicar una sección al algoritmo de Lemke.

3.1. Algoritmo de Lemke

El algoritmo de Lemke es usado para resolver el problema de la complementariedad lineal

Definición 3.1 (PROBLEMA DE LA COMPLEMENTARIEDAD LINEAL)**.** *Dado $q \in \mathbb{R}^n$ y dada una matriz $M \in \mathbb{R}^{nxn}$.* El problema de la complementariedad lineal, *$LCP(q, M)$, consiste en encontrar $w, z \in \mathbb{R}^n$ tales que*

$$\begin{aligned} cw &= q + Mz \\ wz &= 0 \\ w, z &\geq 0 \end{aligned} \tag{3.5}$$

Observación 3.2. Tanto los problemas de programación lineal como los problemas de programación cuadrática convexa pueden reducirse a problemas de complementariedad lineal:

- Por ejemplo, la programación lineal puede ser vista como un caso especial de ecuación 3.5:

$$\begin{array}{lll} \text{máx} & c^T x & \\ \text{s. a.} & Ax \leq b & \Longleftrightarrow \\ & x \geq 0 & \end{array} \qquad \begin{array}{l} s = b - Ax \\ u = -c + A^T y \\ ux = 0 \\ sy = 0 \\ x, y, s, u \geq 0 \end{array}$$

  Por lo que nuestro programa lineal es equivalente a resolver la ecuación

$$w = \begin{bmatrix} u \\ s \end{bmatrix} \qquad z = \begin{bmatrix} x \\ y \end{bmatrix} \qquad q = \begin{bmatrix} -c \\ b \end{bmatrix}$$

$$M = \begin{bmatrix} 0 & A^T \\ -A & 0 \end{bmatrix}$$

ALGORITMO PARA LA COMPLEMENTARIEDAD LINEAL

Para describir este algoritmo, nosotros veremos $w = q + Mz$ como un diccionario para las variables básicas, w, en términos de las variables no básicas, z. Si $q \geq 0$ este diccionario es factible, es decir, la correspondiente solución factible básica ($z = 0$ y $w = q$) es no negativa y hemos terminado. Si no, entonces empezamos la siguiente fase del algoritmo:

- **Fase 1:** Añadimos una variable auxiliar, z_0,

$$w = q + Mz + \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} z_0$$

  Ahora hacemos a z_0 entrar en la base y elegimos la variable que sale de w con el fin de hacer el nuevo diccionario factible (esto significa que todas las variables, incluida z_0, deben ser no negativa).

- **Fase 2:** Nuestro algoritmo termina si llegamos a un diccionario con las siguientes propiedades:

1. z_0 es no básica
2. $\forall i = 1, ..., p$, o bien z_i, o bien w_i es una variable no básica

La condición 1 asegurará que $z_0 = 0$ en la solución factible básica correspondiente, y por lo tanto $w = q + Mz$. La condición 2 asegurará que $wz = 0$.

Definición 3.3. *Aquel diccionario que cumpla las condiciones 1 y 2 le daremos el nombre de* diccionario terminal.
Aquel diccionario que cumpla sólo la condición 2 le daremos el nombre de diccionario equilibrado.

Para llegar a un diccionario terminal, el algoritmo pivota a través de diccionarios factibles equilibrados. Esto se hace como sigue:
Si tenemos un diccionario no terminal, entonces z_0 es básica, entonces alguna variable, o bien w_i, o bien z_i acaba de salir del diccionario. ¿ Qué variable debería entrar en el diccionario en la próxima iteración? Sólo si w_i o z_i entran, podemos asegurar que el nuevo diccionario estará equilibrado. Como no queremos volver al mismo diccionario dos veces, insistiremos en que si w_i salió, entrará z_i en la próxima iteración y viceversa. Diremos que w_i es el complemento de z_i y viceversa.

Ejemplo 3.4.

PROBLEMA PRIMAL

$$\begin{array}{ll} \text{máx} & 2x_1 + x_2 \\ s.a. & x_1 + x_1 \leq 3 \\ & x_1, x_2 \geq 0 \end{array}$$

PROBLEMA DUAL

$$\begin{array}{ll} \text{mín} & 3u_3 \\ & u_3 \geq 2 \\ & u_3 \geq 1 \\ & u_3 \geq 0 \end{array}$$

DICCIONARIO PRIMAL

$$x_1 + x_2 + x_3 = 3 \implies \boxed{x_3 = 3 - x_1 - x_2}$$

DICCIONARIO DUAL

$$\begin{array}{l} -u_3 + u_1 = -2 \\ -u_3 + u_2 = -1 \end{array} \implies \boxed{\begin{array}{l} u_1 = -2 + u_3 \\ u_2 = -1 + u_3 \end{array}}$$

$$w_1 = u_1, w_2 = u_2, w_3 = x_3$$

$$\downarrow \text{ duales}$$

$$z_1 = x_1, z_2 = x_2, z_3 = u_3$$

x_i y u_i son complemetarios

$$\underbrace{\begin{pmatrix} u_1 \\ u_2 \\ x_3 \end{pmatrix}}_{w} = \underbrace{\begin{pmatrix} -2 \\ -1 \\ 3 \end{pmatrix}}_{q} + \underbrace{\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix}}_{M} \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ u_3 \end{pmatrix}}_{z} \underbrace{\left[\begin{pmatrix} z_0 \\ + \ z_0 \\ z_0 \end{pmatrix} \right]}_{\text{Primera fase}}$$

- **Fase 1**

$$\left\{ \begin{array}{c} u_1 = -2 + z_0 + u_3 \\ u_2 = -1 + z_0 + u_3 \\ x_3 = 3 + z_0 - x_1 - x_2 \end{array} \right\}$$

$$r = arg\left(\text{máx}\left\{\frac{2}{1}, \frac{1}{1}\frac{-3}{1}\right\}\right) = 1 \rightarrow u_1$$

Entra z_0 y sale u_1, para conseguir:

$$\begin{array}{l} z_0 = 2 + u_1 - u_3 \\ u_2 = -1 + 2 + u_1 - u_3 + u_3 = 1 + u_1 \\ x_3 = 3 + 2 + u_1 - u_3 - x_1 - x_2 = 5 + u_1 - u_3 - x_1 - x_2 \end{array}$$

- **Fase 2**
Como ha salido u_1, entra ahora $x_1 \implies$ sale x_3

$$\begin{array}{l} z_0 = 2 + u_1 - u_3 \\ u_2 = 1 + u_1 \\ x_1 = 5 + u_1 - u_3 - x_2 - x_3 \end{array}$$

Como ha salido x_3, entra ahora $u_3 \implies$ sale z_0

$$\begin{array}{l} u_3 = 2 + u_1 - z_0 \\ u_2 = 1 + u_1 \\ x_1 = 5 + u_1 - (2 + u_1 - z_0) - x_2 - x_3 = 3 + z_0 - x_3 - x_2 \end{array}$$

Así, nuestra solución óptima primal será: $(x_1, x_2) = (3, 0)$ y nuestra solución óptima dual será: $u_3 = 2$

3.2. Metodología general

En nuestro branch and bound ajustado, nosotros resolvemos la relajación convexa del problema CCQO (esto es, sin la restricción 3.2 ni 3.3) por el método de Lemke. Una vez resuelta elegimos una variable a ramificar x_s. Al ramificar, tenemos dos posibles ramas; podemos elegir o bien que x_s esté en el soporte, o bien que x_s no esté en el soporte. Cuando ramificamos de forma que esta variable no esté en el soporte, actualizaremos el subproblema mediante la supresión de los datos asociados a x_s, y cuando queramos incluir esta variable en el soporte, modificaremos el método de Lemke

de manera que $x_s \geq \alpha_s$ será aplicada durante el pivote. De esta forma, la relajación que nosotros resolvemos en cada nodo cuando queremos incluir x_s en el soporte es:

$$\begin{array}{ll} \text{mín} & \frac{1}{2}x^T Q x + c^T x \\ s.a. & Ax \leq b \\ & x \geq 0 \\ & x_i \geq \alpha_i, \quad i \in U \end{array} \tag{3.6}$$

Donde la restricción de cardinalidad es eliminada y U es el conjunto de índices de las variables que han sido incluídas en el soporte por esa rama. Las restricciones de cota inferior $x_i \geq \alpha_i$ para $\alpha_i > 0$, son aplicadas mediante la implementación del método de Lemke con cotas inferiores distintas de cero (análogo para el método del simplex con cotas inferiores y superiores).A continuación, vamos a describir cómo se usa el método de Lemke para resolver el problema 3.6 en el contexto del branch and bound. Posteriormente, describiremos el procedimiento que se usa para actualizar el problema del nodo hijo cuando la variable de ramificación es eliminada.

3.2.1. Método de Lemke como optimizador cuadrático subyascente

Como hemos dicho antes, utilizaremos el método pivotante de Lemke para optimizar la relajación convexa en cada nodo del branch and bound. Este método que fue desarrollado originalmente para resolver problemas de complementariedad lineal (de los cuales los programas cuadráticos son un caso especial) a través de un pivote similar al del método del simplex. Al igual que el método del simplex dual en la optimización lineal, la ventaja clave del método de Lemke es su facilidad y eficiencia a partir de una solución básica no factible . Esto es crítico en este branch and bound ajustado, ya que la solución óptima del nodo padre puede ser utilizada como el punto inicial para resolver el problema del nodo hijo.

El método de Lemke primero comprueba si $q \geq 0$, en cuyo caso $z = 0$ y $w = q$ es una solución. De lo contrario, aumenta $LCP(q, M)$ a

$$w = q + hz_0 + Mz \geq 0, \qquad z_0 \geq 0, \qquad z \geq 0, \qquad z^T w = 0 \tag{3.7}$$

donde h es un vector dado cuyas componentes son positivas.

Necesitamos empezar el algoritmo con una base complementaria, que no tiene porque satisfacer necesariamente la restricción de la no negatividad. Una base por

defecto sencilla es tener todas las z variables no básicas y todas las w variables básicas. A continuación, establecemos como variable auxiliar z_0 al valor positivo más pequeño tal que $w \geq 0$ cuando $z = 0$, es decir,

$$w = q + hz_0 + 0 \geq 0 \Leftrightarrow w = q + hz_0 \geq 0 \rightarrow z_0 \geq \frac{-q}{h}$$

$$\boxed{z_0 = \max_i \frac{-q_i}{h_i}, \quad i = 1, ..., d.}$$

Así, $z_i w_i = 0$, $i = 1, ..., n$ y $z_0 > 0$, y z_0 es pivotado a la base en lugar de w_r, donde $r = argmax_i \frac{-q_i}{h_i}$. Este punto se llama un punto casi complementario para el problema aumentado 3.7.

El algoritmo sigue una trayectoria desde una solución básica casi complementaria a la siguiente, hasta que z_0 pivota hacia fuera para ser una variable no básica, o por otro lado, se demuestra que $LCP(q, M)$ no es factible.

Durante el procedimiento branch and bound, queremos resolver un nuevo nodo a partir de la solución básica del nodo padre. Sean $\overline{M}$ y $\overline{q}$ los datos modificados para el nodo hijo, y sean $\overline{B}$ y $\overline{N}$ las correspondientes columnas de las variables básicas y no básicas, respectivamente, del nodo padre. Queremos el método de Lemke para resolver $LCP(M_{\overline{B}}, q_{\overline{B}})$, donde $M_{\overline{B}} = -\overline{B}^{-1}\overline{N}$ y $q_{\overline{B}} = \overline{B}^{-1}\overline{q}$, ya que

$$\left.\begin{array}{rcl} \overline{M}x & = & \overline{q} \\ \overline{M} & = & [\overline{B}, \overline{N}] \end{array}\right\} \implies [\overline{B}, \overline{N}] \begin{bmatrix} x_B \\ x_N \end{bmatrix} = \overline{q} \implies \overline{B}x_B + \overline{N}x_N = \overline{q} \implies$$

$$\implies \overline{B}x_B = \overline{q} - \overline{N}x_N \implies x_B = \overline{B}^{-1}q - \overline{B}^{-1}\overline{N}x_N$$

$$\implies \boxed{\begin{array}{l} q_{\overline{B}} = \overline{B}^{-1}\overline{q} \geq 0 \\ \\ M_{\overline{B}} = -\overline{B}^{-1}\overline{N} \end{array} , \qquad x_B = q_{\overline{B}} + M_{\overline{B}}x_N}$$

y así, tenemos $\overline{B}$ como su matriz de base complementaria inicial. Esta base es muy probable que no sea factible para $LCP(M_{\overline{B}}, q_{\overline{B}})$, por lo que el problema se ve aumentado por la variable auxiliar z_0, z_0 se incrementa hasta que la base inicial es factible para el problema aumentado, y entonces ejecutamos secuencia de pivotes hasta que z_0 pivota hacia fuera o el LCP se considera no factible.

3.2.2. Actualización del nodo hijo cuando la variable de ramificación no es incluida en el soporte

Como indicamos anteriormente, cuando al ramificar sobre x_s no queremos incluirla en el soporte, en lugar de añadir de forma explícita la restricción $x_s = 0$ para evitar que el problema aumente de tamaño, eliminamos todos los datos asociados a ella en los sucesivos nodos hijos. Además en cada nodo tendremos que actualizar la base como explicaremos a continuación.

Vamos a demostrar que, en la mayoría de los casos, la inversa de la matriz de la nueva base (base en el nodo hijo) se puede calcular de manera eficiente de la inversa de la matriz de la base antigua (base en el nodo padre) a través de operaciones elementales por filas.

Supongamos que x_s es una variable básica y supongamos B y N las columnas básicas y no básicas, respectivamente, de la solución anterior. Borramos la columna y la fila de B correspondiente a x_s y la columna y la fila de N correspondiente a su variable dual w_s. Aunque podemos obtener la nueva inversa de la base simplemente invirtiendo la base modificada, el cálculo de la inversa puede ser muy costoso (por ejemplo, cuando la dimensión del problema sea muy grande).

En su lugar, se calcula la nueva inversa a partir de la inversa anterior, mediante operaciones elementales por filas. Supongamos, sin pérdida de generalidad, que la columna y la fila que se necesita eliminar en B son la primera columna y primera fila y supongamos $B \in \mathbb{R}^{nxn}$

$$B = \begin{bmatrix} v & B_{row}^T \\ B_{col} & \overline{B} \end{bmatrix}; \qquad B^{-1} = \begin{bmatrix} u & U_{row}^T \\ U_{col} & \overline{U} \end{bmatrix}$$

Donde:

$\overline{B}$ y $\overline{U}$ son matrices $(n-1)$x$(n-1)$

$B_{col}, B_{row}, U_{col}, U_{row}$ son vectores columnas de dimensión $n-1$

v, u son escalares

Sabemos que

$$B^{-1}B = \begin{bmatrix} uv + U_{row}^T B_{col} & uB_{row}^T + U_{row}^T U_{col} \\ vU_{col} + \overline{U}B_{col} & U_{col}B_{row}^T + \overline{U}\,\overline{B} \end{bmatrix} = I_n$$

Luego,

$$U_{col}B_{row}^T + \overline{U}\,\overline{B} = I_{n-1} \implies \overline{U}\,\overline{B} = I_{n-1} - U_{col}B_{row}^T$$

Observación 3.5. Observamos que $U_{col}B_{row}^T$ es una matriz de rango 1, ya que

$$U_{col} = \begin{pmatrix} u_{col}^{(1)} \\ u_{col}^{(2)} \\ \vdots \\ u_{col}^{(n-1)} \end{pmatrix}; \qquad B_{row}^T = \left(b_{row}^{(1)}, ..., b_{row}^{(n-1)}\right)$$

$$U_{col}B_{row}^T = \begin{pmatrix} u_{col}^{(1)}b_{row}^{(1)} & u_{col}^{(1)}b_{row}^{(2)} & ... & u_{col}^{(1)}b_{row}^{(n-1)} \\ u_{col}^{(2)}b_{row}^{(1)} & u_{col}^{(2)}b_{row}^{(2)} & ... & u_{col}^{(2)}b_{row}^{(n-1)} \\ \vdots & \vdots & \vdots & \vdots \\ u_{col}^{(n-1)}b_{row}^{(1)} & u_{col}^{(n-1)}b_{row}^{(2)} & ... & u_{col}^{(n-1)}b_{row}^{(n-1)} \end{pmatrix}$$

El hecho de que sea de rango 1 nos indica que podemos transformar $I_{n-1} - U_{col}B_{row}^T$ en I_{n-1} mediante operaciones elementales por filas.

Así, si E es la matriz que representa estas operaciones, $E\overline{U}\,\overline{B} = I_{n-1}$ y por lo tanto, $E\overline{U}$ es la matriz inversa de $\overline{B}$, siempre y cuando $\overline{B}$ sea invertible.

Como anteriormente indicamos, tomariamos $M_{\overline{B}} = -\overline{B}^{-1}\overline{N}$ como uno de los datos de entrada en el método de Lemke. El objetivo que nos proponemos a continuación es evitar la multiplicación de estas matrices a través de operaciones elementales por filas similares a las anteriores.

Supongamos $M_B = -B^{-1}N$ en el nodo padre. Una vez más, vamos a suponer que la columna correspondiente a w_s en N es la primera. AsÃ,

$$M_B = -B^{-1}N = -\begin{bmatrix} u & U_{row}^T \\ U_{col} & \overline{U} \end{bmatrix}\begin{bmatrix} p & N_{row}^T \\ N_{col} & \overline{N} \end{bmatrix} = \begin{bmatrix} w & M_{row}^T \\ M_{col} & \overline{M} \end{bmatrix}$$

Donde:

$\overline{N}$ y $\overline{M}$ son matrices $(n-1)$x$(n-1)$

$N_{col}, N_{row}, M_{col}, M_{row}$ son vectores columnas de dimensión $n-1$

p, w son escalares

Luego,

$$-U_{col}N_{row}^T - \overline{U}\,\overline{N} = \overline{M} \implies -\overline{U}\,\overline{N} = \overline{M} + U_{col}N_{row}^T$$

.

Además, como $U_{col}N_{row}^T$ tiene rango 1, podemos tranformar $\overline{M} + U_{col}N_{row}^T$ en $\overline{M}$ mediante operaciones elementales por filas. Así, si $E^{'}$ es la matriz que representa estas operaciones, $-E^{'}\overline{U}\,\overline{N} = \overline{M} = -\overline{B}^{-1}\,\overline{N}$, por lo que $E^{'}\overline{U} = \overline{B}^{-1}$ y la nueva matriz $M_{\overline{B}}$ será

$$M_{\overline{B}} = E^{'}(\overline{M} + U_{col}N_{row}^T) \tag{3.8}$$

Antes de la ejecución de estos procedimientos, hay que probar que $\overline{B}$ no sea una matriz singular, en cuyo caso $E^{'}$ puede ser indefinida. En el caso de que fuese singular, empezaríamos el método de Lemke desde cero con la base inicial $\overline{B} = I_{n-1}$. Por otro lado, si x_s es una variable no básica en la solución anterior, podemos aplicar la siguiente metodología para su variable complementaria w_s, que debe ser una variable básica.

Para actualizar q_B , eliminamos el elemento s-ésimo de c, obteniendo $\overline{c}$. Suponemos $q_B = B^{-1}q$, donde $q = \begin{bmatrix} c \\ b \end{bmatrix}$. De nuevo, asumiendo que $s = 1$ tenemos,

$$q_B = \begin{bmatrix} \tilde{q}_s \\ \tilde{q}_B \end{bmatrix} = B^{-1}q = \begin{bmatrix} u & U_{row}^T \\ U_{col} & \overline{U} \end{bmatrix} \begin{bmatrix} q_s \\ \overline{q}_B \end{bmatrix}$$

Donde:

$\tilde{q}_B$ y $\overline{q}$ son los subectores formados por las $n-1$ filas últimas de q_B y q, respectivamente y $q_s = c_s$.

Aplicando un razonamiento similar al que hicimos anteriormente para la obtención de $M_{\overline{B}}$, obtenemos

$$q_{\overline{B}} = E^{''}(\tilde{q}_B - q_s U_{col}) \tag{3.9}$$

Si observamos 3.8 y 3.9, para actualizar M y q a lo largo del algoritmo, sólo necesitamos conocer una columna de B^{-1}; U_{col}. Así en lugar de mantener explícitamente

B^{-1}, podemos calcular la descomposición LU de la matriz B ($B = LU$, donde L es triangular inferior y U es triangular superior) y usarla sólo para obtener la columna requerida de B^{-1}.

3.3. Resultados computacionales para selección de carteras

Como puede verse con detalles en [4], esta metodología puede aplicarse a los problemas de optimización de carteras con niveles mínimos de transación y con cota superior en el número de variables positivas. En este mismo artículo, encontramos experimentos computacionales sobre este problema, en el que se compara esta metodología a aquella usada por el solucionador de programación cuadrático entero-mixto CPLEX.

Es evidente que la implementación de CPLEX de los métodos pivotantes y branch-and-bound es muy superior a la descrita anteriormente. Sin embargo, el objetivo del autor era medir las ventajas de una implementación adaptada sobre un solucionador general entero mixto para estos problemas CCQO particulares.

Para ello, extrajo los datos recogidos en la siguiente tabla, para cada d (total de número de acciones, o lo que es lo mismo, dimensión del problema), S (número de sectores) y K (cota superior de cardinalidad).

d	K	S	LemkeBnB			CplexMIQP		
			Nodes	Best node	UB	Nodes	Best node	UB
100	50	10	16992.20	16039.60	28.46	119697.00	76987.00	19.02
100	10	4	27231.40	8329.40	18.83	58530.20	12736.60	18.49
200	100	10	2952.80	2933.80	142.73	30188.60	30062.80	81.86
200	20	4	6913.20	1281.00	38.73	7236.80	1245.60	38.73
500	200	10	142.40	-	-	4864.00	4812.60	345.46
500	100	10	164.40	137.80	159.68	782.60	398.00	158.68
500	50	4	64.20	46.40	90.89	226.40	140.40	90.89

Cuadro 3.1: Resultados para selección de cartera, resueltos hasta 120 segundos CPU (tiempo de proceso)

Donde:

- Aquellas entradas "-", indican que el método no encontró una solución factible en un tiempo inferior o igual a 120 segundos.

- "LemkeBnB"se refiere a los resultados obtenidos aplicando el método descrito en este capítulo.
- “CplexMIQP"se refiere a los resultados obtenidos usando el solucionador cuadrático entero-mixto CPLEX.
- La coluna “nodes” es la media del total de número de nodos de cada método.
- “Best node” es la media de los nodos donde se encotraron las mejores soluciones factibles.
- “UB” es el mejor valor objetivo encontrado.

Capítulo 4

Conclusiones del T.F.M.

El objetivo de este trabajo era estudiar problemas de optimización que aparecen en la gestión de valores, bonos y activos financieros cuando no se hacen hipótesis distribucionales sobre las fluctuaciones del mercado. Se pretendía plantear modelos de programación matemática, estudiar sus propiedades y desarrollar algoritmos que permitan resolverlos.

Para ello, nos hemos centrado principalmente en el problema de optimización de carteras de Markowitz con niveles mínimos de transación y con cota superior en el número de variables positivas, y hemos desarrollado dos algoritmos principales para su resolución.

En el primero de ellos, y con el fin de preservar la rápida resolución de los programas cuadráticos, aunque a costa de una mayor ramificación, hemos sustituido la restricción de cardinalidad usando la restricción (2.7). Además, hicimos experimentos computacionales a partir de los cuales comprobamos que el hecho de sustituir la restricción de cardinalidad por esta nueva "restricción sustituta" era insuficiente para resolver nuestro problema. Por ello era necesario la ramificación.

También, hemos desarrollados cuatro planos de corte que facilitan la resolución de nuestro problema. En este punto, pueden abrirse numerosas líneas de investigación en la búsqueda de otros posibles planos que reduzcan aún más la región factible y de esa forma, agilicen el proceso de resolución.

Igualmente, quisimos sensibilizarnos con el problema comparándolo con su versión continua. Para ello, implementamos ambos problemas, concluyendo que en efecto, mientras la versión continua se resuelve rápidamente, incluso en los casos en los que trabajamos con dimensiones altas, el modelo entero-mixto se complica enormemente a medida que realizamos pequeños incrementos en la dimensión del problema. Ade-

más, en el caso del problema entero-mixto, para una dimensión fijada, el problema se complica a medida que aumentamos la cota superior en el número de variables estrictamente positivas.

La metodología descrita en el tercer capítulo utiliza el método pivotante de Lemke para optimizar la relajación convexa en cada nodo del branch and bound. La ventaja clave del método de Lemke es su facilidad y eficiencia a partir de una solución básica no factible. Esto es crítico en este branch and bound ajustado, ya que la solución óptima del nodo padre puede ser utilizada como el punto inicial para resolver el problema del nodo hijo.

Por último, hemos comprobando a partir del artículo [4], que en efecto, este algoritmo puede usarse para la resolución de problemas de optimización de carteras con niveles mínimos de transación y con cota superior en el número de variables positivas, y que presenta numerosas ventajas como se observa en los experimentos computacionales de dicho artículo.

Bibliografía

[1] Gerard Cornvejols: Oprimization methods in finance. Cornegie Mellon University, Pitysburgh, PA 15213 USA, Summer 2005

[2] D. Bienstock. Computational study of a family of mixed-integer quadratic programming problems. Mathematical Programming A, 74:121-140, 1996.

[3] G.L. Nemhauser and L.A. Wolsey. Integer and Combinatorial Optimization (Wiley,New York,1988)

[4] Dimitris Bertsimas and Romy Shyoda. Algorithm for cardinality-constrained quadratic optimization.Computational Optimization and Applications, 43:1-22, 2009.

[5] Joel Friedman. Linear Complementary and Mathematical (Non-linear) Programming. April 3, 1998

[6] Eduardo Ramos Méndez. Modelos y Métodos de optimización. Segunda edición, Noviembre 2009