Investigación Operativa

Álvaro García Tenorio [1]

11 de septiembre de 2017

[1] alvgar14@ucm.es

Índice general

Prefacio

Estas notas son una transcripción (libremente adaptada) de las clases de la asignatura "*Investigación Operativa*", impartidas por María Inés Sobrón Fernández en el curso 2016–2017 a los cursos de cuarto y tercero de los dobles grados de Matemáticas – Física e Ingeniería Informática – Matemáticas (respectivamente) en la facultad de Ciencias Matemáticas de la Universidad Complutense de Madrid (UCM).

Cualquier aportación o sugerencia de mejora es siempre bienvenida.

Requisitos previos

Para comprender estas notas en su totalidad es necesario tener soltura a la hora de trabajar con matrices, y, en general, haber entendido bien el álgebra lineal. También es bastante recomendable recordar algunos aspectos del cálculo diferencial en varias variables y teoría de grafos, no obstante, el texto es bastante autocontenido en ese aspecto.

Agradecimientos

La existencia de estas notas es debida a la amabilidad de Clara Rodríguez Núñez, quien me cedió sus apuntes tomados durante el curso, en los cuales se basa el núcleo de este texto.

Licencia

Parte I

Programación lineal

Capítulo 1

Fundamentos teóricos

La disciplina de la programación lineal cogió fuerza en la segunda guerra mundial, pues era necesario optimizar los escasos recursos de los que se disponían para realizar operaciones militares. En un principio fue impulsada por el ejército británico, con un grupo de investigación dirigido por ***Blackett***, quien más adelante ganaría el premio Nobel de física por otros motivos.

Cabe destacar la gran importancia que tuvo la programación lineal en países como Albania, debido, entre otras cosas, a su utilidad a la hora de optimizar los procesos de producción de carbón.

1.1. Planteamiento del problema

De ahora en adelante, consideraremos fijadas las bases canónicas tanto de $\mathbb{R}^n$ como de $\mathbb{R}$, a no ser que se especifique lo contrario.

Antes de plantear formalmente el problema de la programación lineal, introduzcamos unas definiciones.

Definición 1.1.1 (Vector no negativo)**.** Un vector $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ se dice ***no negativo*** si todas sus componentes son, como su propio nombre indica, no negativas. Es decir, $x_i \geq 0$ para todo $i \in \{1, \ldots, n\}$. Si un vector x es no negativo escribiremos $x \geq 0$.

Al hilo de esta definición, vamos a generalizar ese concepto que lleva presente en nuestras vidas desde que aquel docente de primaria nos dibujó dos rectas perpendiculares en la pizarra, los cuadrantes.

Definición 1.1.2 (2^n–ante positivo)**.** Definimos el 2^n–***ante positivo*** de $\mathbb{R}^n$ como el conjunto de los vectores no negativos de $\mathbb{R}^n$.

Se deja como ejercicio al lector justificar el nombre de 2^n–ante.

Dicho esto, ya podemos formular el problema de la programación lineal.

Problema 1.1 (Formulación general)**.** La disciplina de la ***programación lineal*** estudia procedimientos (implementables en ordenador) para calcular los extremos absolutos de una aplicación lineal $f : \mathbb{R}^n \to \mathbb{R}$ cuando restringimos su dominio a una variedad afín de $\mathbb{R}^n$ cortada con el 2^n–ante positivo.

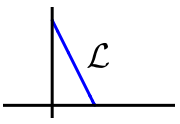

Figura 1.1: Ilustración del problema de programación lineal.

En el caso de la ilustración 1.1 suponemos que f es una aplicación lineal definida sobre todo el plano a la cual restringimos el dominio a la variedad afín $\mathcal{L}$ cortada con el cuadrante positivo (línea azul). Nuestro deber es encontrar los puntos (si los hay) sobre la línea azul en los cuales f alcanza un máximo o un mínimo absoluto.

En esta sección nos dedicaremos simplemente a formular de diversas maneras el problema de la programación lineal, observando que todas son equivalentes entre sí. Comencemos pues, sin más dilación nuestro viaje por lo desconocido, un viaje del que probablemente no regresaremos.

A continuación vemos una sencilla caracterización de las aplicaciones lineales $f : \mathbb{R}^n \to \mathbb{R}$ a partir de su expresión analítica.

Observación 1.1.1 (Expresión analítica)**.** Consideremos una aplicación lineal $f : \mathbb{R}^n \to \mathbb{R}$.

Por ser f una aplicación lineal, tendrá una única matriz asociada C. Como salta a la vista, C será una matriz fila con n columnas.

Entonces, dado un vector $x \in \mathbb{R}^n$ se tiene que

$$f(x) = f(x_1, \dots, x_n) = CX = \begin{pmatrix} c_1 & \cdots & c_n \end{pmatrix} \begin{pmatrix} x_1 \\ \cdots \\ x_n \end{pmatrix} = c_1 x_1 + \cdots + c_n x_n \tag{1.1}$$

De esta forma se concluye que toda aplicación lineal $f : \mathbb{R}^n \to \mathbb{R}$ tiene una expresión analítica de la forma de la ecuación (1.1). Recíprocamente, toda función $g : \mathbb{R}^n \to \mathbb{R}$ con una expresión analítica como la de (1.1) es una ecuación lineal, para demostrarlo, basta con leer como los árabes la ecuación (1.1). ♢

Antes de continuar es importante hacer un inciso acerca de la notación.

Observación 1.1.2 (Notación matricial)**.** Siempre que escribamos matrices fila o columna lo haremos con letras minúsculas. Asimismo, siempre que estemos escribiendo una matriz fila añadiremos el símbolo de trasposición, para dejarlo claro implícitamente. De esta manera, a las funciones lineales las denotaremos por

$$f(x_1 \dots, x_n) = c^t x$$

donde c^t es la matriz asociada a f y x es el vector columna con las coordenadas de x respecto de la base canónica (nótese el pequeño abuso de notación). ♢

1.1.1. Forma estándar y formas canónicas

Sea $f(x_1, \dots, x_n) = c_1 x_1 + \cdots + c_n x_n = c^t x$ una función lineal a la que a partir de ahora llamaremos ***función objetivo***. A la matriz c se la denomina ***vector de coeficientes de la función objetivo***, mientras que x recibe el nombre de ***vector de variables de decisión***.

Consideremos asimismo la variedad afín de $\mathbb{R}^n$ dada por el conjunto de soluciones al sistema no necesariamente homogéneo $Ax = b$. Escrito de forma desarrollada

$$\begin{cases} a_{11}x_1 & + \cdots + & a_{1n}x_n & = b_1 \\ \vdots & \ddots & \vdots & \vdots \\ a_{1m}x_1 & + \cdots + & a_{mn}x_n & = b_m \end{cases} \qquad \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

A la matriz $A \in \mathfrak{M}_{m \times n}(\mathbb{R})$ usualmente la llamaremos ***matriz de coeficientes de las restricciones***, además, a cada una de las ecuaciones del sistema las llamaremos ***restricciones***. Asimismo, el vector b será conocido como ***vector de términos independientes***.

Expongamos ahora la llamada "*formulación estándar*" del problema de la programación lineal.

Problema 1.2 (Forma estándar)**.** El problema de programación lineal en ***forma estándar*** consiste en hallar los vectores $x \geq 0$ tales que sean solución de $Ax = b$ y además sean mínimos absolutos de la función f. Usualmente se escribe de forma sintética (aunque no lo usaremos mucho)

$$\begin{aligned} & \min c^t x \\ \text{Sujeto a:} \quad & Ax = b, \qquad x \geq 0 \end{aligned}$$

Nótese que la formulación del problema en forma estándar es exactamente la misma que la formulación general 1.1 con la única diferencia de que ya no buscamos extremos absolutos en general, sino únicamente mínimos.

La formulación estándar del problema de programación lineal puede parecer muy inflexible, en el sentido de que, a simple vista, no parece ser demasiado útil para modelizar problemas reales. Es esta aparente rigidez la que lleva a plantearse otras formulaciones más laxas del problema de programación lineal. Las llamadas "*formulaciones canónicas*" que exponemos a continuación.

Problema 1.3 (Primera forma canónica)**.** Esta nueva formulación a la que llamamos ***primera forma canónica*** plantea el problema de hallar los vectores $x \geq 0$ tales que satisfagan la ecuación $Ax \leq b$ y que además sean máximos absolutos de la función objetivo f. Escrito de forma compacta

$$\begin{array}{ll} & \max c^t x \\ \text{Sujeto a:} & Ax \leq b, \qquad x \geq 0 \end{array}$$

Problema 1.4 (Segunda forma canónica)**.** El problema de programación lineal en ***segunda forma canónica*** consiste en encontrar los vectores $x \geq 0$ tales que satisfagan la ecuación $Ax \geq b$ y que además sean mínimos absolutos de la función objetivo f. En resumen

$$\begin{array}{ll} & \min c^t x \\ \text{Sujeto a:} & Ax \geq b, \qquad x \geq 0 \end{array}$$

Una vez vistas todas las formulaciones, hagamos una pequeña reflexión. Sería la apoteosis del tedio tener que demostrar resultados teóricos para cada una de las formulaciones. No obstante, no hay que dejar nunca de confiar en los dioses olímpicos, pues a veces son benévolos con nosotros. Esta benevolencia consiste en que todas los formulaciones son equivalentes en el sentido que veremos a continuación.

Maximización y minimización

Si se nos plantea el problema de hallar el máximo absoluto de una función objetivo f podemos transformar este problema en uno equivalente de forma que el objetivo de el nuevo problema sea hallar el mínimo de otra función objetivo g.

La utilidad de este hecho es que podemos transformar un problema de programación lineal en primera forma canónica (que surgen de forma natural a la hora de modelizar problemas reales) en un problema más parecido a la formulación estándar.

Lema 1.1.1 (Maximización–minimización)**.** *Sea A un conjunto arbitrario y una función $f : A \to \mathbb{R}$ que alcanza el máximo y el mínimo. Entonces se cumple*

$$\max f = -\min(-f)$$

Demostración. Sea ξ el máximo de f, es decir, el único número de $f(A)$ que verifica que $f(x) \leq \xi$ para todo $x \in A$. Consideremos ahora la función $g \equiv -f$, que, por las propiedades de los números reales alcanza el mínimo, al que llamaremos η.

Evidentemente η verifica que $g(x) \geq \eta$ para todo $x \in A$, o equivalentemente $-g(x) \leq -\eta$ para todo $x \in A$. Pero $-g(x) = f(x)$, luego $f(x) \leq -\eta$ para todo $x \in A$. Por unicidad del máximo $-\eta = \xi$, como queríamos demostrar. ■

Observación 1.1.3 (Aplicación)**.** Si tenemos un problema de programación lineal formulado en términos de maximización bastará con cambiar la función objetivo f por la función objetivo $-f$ para obtener un problema equivalente en términos de minimización.

Para obtener las soluciones del primer problema en términos del segundo bastará con cambiar el valor por la función objetivo de las soluciones del segundo problema por su opuesto. ♢

Igualdades y desigualdades

Una vez visto el apartado anterior, vamos a aprender a trasformar restricciones de tipo desigualdad a restricciones de tipo igualdad, con lo cual ya podremos transformar cualquier problema planteado en alguna forma canónica a un problema equivalente planteado en forma estándar.

Lema 1.1.2 (Variables de holgura)**.** *Sea una restricción de la forma*

$$a_{i1}x_1 + \cdots + a_{in}x_n \overset{(\geq)}{\leq} b_i \tag{1.2}$$

entonces la restricción

$$a_{i1}x_1 + \cdots + a_{in}x_n \overset{(-)}{+} x_i^h = b_i \tag{1.3}$$

es equivalente, en el sentido de que hay una biyección "natural" entre los conjuntos de vectores ***no negativos*** *que verifican cada una de las restricciones.*

A la variable x_i^h añadida en la restricción (1.3) *se la denomina* ***variable de holgura****.*

Demostración. Sean los conjuntos $\mathcal{S}$ y $\mathcal{S}'$ de vectores que verifican las restricciones (1.2) y (1.3) respectivamente. Consideremos la aplicación

$$\begin{aligned} \varphi : \quad & \mathcal{S} \to \mathcal{S}' \\ & (x_1, \ldots, x_n) \mapsto (x_1, \ldots, x_n, z) \end{aligned}$$

donde $z = b_i - a_{i1}x_1 + \cdots + a_{in}x_n$. Veamos que es una biyección.

La inyectividad es clara por definición, si dos vectores compartieran imagen todas sus componentes coinciden.

En cuanto a la sobreyectividad, dado un vector $(x_1, \ldots, x_n, x_{n+1})$ que verifica la restricción (1.3), es claro que el vector resultante de eliminar la última componente, $(x_1, \ldots, x_n)$, cumple la restricción (1.2).

La demostración para el caso $\geq$ es totalmente simétrica. ■

Veamos que utilidad tiene lo que acabamos de demostrar.

Observación 1.1.4 (Aplicación)**.** Si a la hora de modelizar un problema nos aparecen restricciones de tipo $\geq$ o $\leq$ basta con sustituirlas por restricciones equivalentes añadiendo variables de holgura.

Además, debemos cambiar la función objetivo f, extendiéndola de la siguiente manera

$$\begin{aligned} \widehat{f} : \quad & \mathbb{R}^{n+m} \to \mathbb{R} \\ & (x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+m}) \mapsto f(x_1, \ldots, x_n) \end{aligned}$$

donde n es el número total de restricciones originales y m el número de restricciones de tipo desigualdad.

Para obtener las soluciones del problema original en términos de este nuevo problema equivalente basta con quitar las componentes correspondientes a las variables de holgura. ♢

Con lo que sabemos hasta el momento podemos transformar cualquier problema escrito en cualquiera de las formas canónicas en un problema en forma estándar.

Positividad y negatividad

Otro caso que se presenta de manera usual es que al modelizar un problema concreto sea necesario permitir que algunas componentes de los vectores candidatos a ser solución sean negativas. Esta es una situación que no contempla la formulación estándar, sin embargo, con un simple truco podemos meter esta clase de problemas dentro de nuestro marco de actuación.

Observación 1.1.5 (Truco)**.** Si tenemos un problema en el que la componente i–ésima de los vectores puede ser negativa tomamos todas las restricciones

$$a_{j1}x_1 + \cdots + a_{ji}x_i + \cdots + a_{jn}x_n = b_j$$

que involucren a la componente x_i y las sustituimos por las restricciones

$$a_{j1}x_1 + \cdots + a_{ji}(x_i^+ - x_i^-) + \cdots + a_{jn}x_n = b_j$$

con x_i^+ y x_i^- no negativos.

Es claro que si un vector $(x_1, \ldots, x_i, \ldots, x_n)$ (donde x_i es negativo) verifica las restricciones originales, entonces el vector $(x_1, \ldots, 0, -x_i, \ldots, x_n)$ (por ejemplo) verifica las restricciones nuevas. Análogamente si x_i es positivo. Asimismo, si el vector $(x_1, \ldots, x_i^+, x_i^-, \ldots, x_n)$ cumple las restricciones nuevas, el vector $(x_1, \ldots, x_i^+ - x_i^-, \ldots, x_n)$ cumplirá las viejas, pudiendo concluir así que las restricciones viejas y nuevas son equivalentes en un sentido un poco más laxo que en el apartado anterior, ya que únicamente se tiene la sobreyectividad (pero esto es irrelevante).

De esta forma, si cambiamos la función objetivo f por la función

$$\begin{aligned} \overline{f}: \quad & \mathbb{R}^{n+1} \to \mathbb{R} \\ & (x_1 \ldots, x_i^+, x_i^-, \ldots, x_n) \mapsto f(x_1, \ldots, x_i^+ - x_i^-, \ldots, x_n) \end{aligned}$$

obtenemos un problema equivalente.

Para obtener la solución del problema original en términos de este nuevo problema basta considerar el vector $(x_1, \ldots, x_i^+ - x_i^-, \ldots, x_n)$ donde $(x_1 \ldots, x_i^+, x_i^-, \ldots, x_n)$ es el vector solución del problema con las restricciones nuevas. ♢

Hechas todas estas disquisiciones iniciales cabe mencionar que a la hora de modelizar problemas, siempre que tanto la función objetivo como las restricciones sean lineales, podremos realizar transformaciones (las vistas en esta sección) para que el problema quede planteado en forma estándar, pudiendo ser así resuelto mediante el uso de toda la artillería teórica que veremos a continuación.

1.1.2. Terminología y casuística

Dado un problema de programación lineal en forma estándar que tiene a A como matriz de coeficientes de las restricciones y a b como vector de términos independientes. Exponemos las siguientes definiciones.

Definición 1.1.3 (Soluciones)**.** Llamamos ***solución*** del problema P a cualquier vector $x \in \mathbb{R}^n$ (no necesariamente no negativo) que verifique $Ax = b$.

Sacando un poco de punta al asunto, diremos que si $x \in \mathbb{R}^n$ es solución del problema y además $x \geq 0$ entonces x es una ***solución factible***.

Llamaremos ***región factible*** al conjunto de todas las soluciones factibles del problema. Normalmente la denotaremos con la letra $\mathcal{S}$.

Una vez resuelto el problema se pueden dar las siguientes situaciones

- La región factible posee al menos un punto en el cual la función objetivo alcanza el mínimo. En esta situación diremos que el problema posee ***solución óptima***.
- La región factible del problema es el conjunto vacío. En este caso se dirá que el problema es ***infactible***.
- La función objetivo alcanza valores tan bajos como queramos en la región factible. En este caso se dice que el problema posee una solución ***no acotada***.

A lo largo del capítulo veremos por qué no se puede dar ninguna otra situación.

Teniendo en cuenta estas pequeñas observaciones podemos empezar a estudiar cómo son las regiones factibles, lo cual nos será muy útil para resolver el problema de programación lineal.

1.2. Programación lineal y conjuntos convexos

En esta sección demostraremos que las regiones factibles de un problema de programación lineal en forma estándar son siempre conjuntos convexos.

Asimismo, estudiaremos en profundidad las propiedades de estos conjuntos, explotándolas para obtener un procedimiento robusto con el cual resolver estos problemas.

1.2.1. Conjuntos convexos. Definición y propiedades

Comenzamos definiendo el concepto de conjunto convexo.

Definición 1.2.1 (Convexos)**.** Sea $\mathcal{S}$ un subconjunto de $\mathbb{R}^n$, se dice ***convexo*** si para cada par de puntos $x, y \in \mathcal{S}$ el segmento que tiene por extremos a x e y está contenido en $\mathcal{S}$. Es decir

$$\lambda x + (1-\lambda)y \in \mathcal{S}$$

para todo $\lambda \in [0,1]$. Nótese que el conjunto vacío se considera convexo.

La región factible de un problema de programación lineal en forma estándar es un conjunto convexo.

Lema 1.2.1 (Región convexa)**.** *El conjunto de los vectores no negativos $x \in \mathbb{R}^n$ que verifican la condición $Ax = b$ para cierta matriz A y cierto vector b es un conjunto convexo.*

Demostración. Si la región factible $\mathcal{S}$ es el conjunto vacío ya hemos terminado. En otro caso, basta considerar dos vectores x e y de la región factible y ver que $\alpha x + (1-\alpha)y \in \mathcal{S}$ para todo $\alpha \in [0,1]$, pero esto es inmediato ya que

$$A(\alpha x + (1-\alpha)y) = \alpha Ax + (1-\alpha)Ay = \alpha b + (1-\alpha)b = b \qquad \blacksquare$$

Una cuestión importante a destacar es que la intersección arbitraria de conjuntos convexos es convexo, tal y como muestra el siguiente lema.

Lema 1.2.2 (Intersección)**.** *Sea $\mathcal{F}$ una familia de conjuntos convexos. La intersección de la familia es un conjunto convexo.*

Demostración. Sean x e y dos puntos de la intersección de la familia, luego x e y están en todos los conjuntos de la familia. Como estos conjuntos son convexos, el segmento que une x e y estará contenido en todos los conjuntos de la familia simultáneamente, luego también en la intersección de la familia. $\blacksquare$

Un concepto por el que merece la pena pasar es el de "envoltura convexa" de un conjunto $\mathcal{S}$ de $\mathbb{R}^n$, que viene a ser el menor conjunto convexo que contiene a $\mathcal{S}$.

Esta idea es especialmente recurrente en las matemáticas, por ejemplo, en álgebra lineal teníamos el subespacio generado por un subconjunto, en teoría de grupos el subgrupo generado por un subconjunto, en topología tenemos la adherencia de un conjunto,... La lista es interminable.

Observación 1.2.1 (Justificación)**.** Es fácil darse cuenta de que, como la intersección arbitraria de convexos es convexa, el menor conjunto convexo que contiene a uno dado (al que llamaremos $\mathcal{S}$) puede ser construido de la siguiente manera

$$\mathrm{Conv}(\mathcal{S}) := \bigcap_{\mathcal{F} \supset \mathcal{S}} \mathcal{F}$$

En efecto, es un conjunto que contiene a $\mathcal{S}$, ya que todos los conjuntos de la familia a intersecar contienen a $\mathcal{S}$, además, es el menor de ellos, ya que, de haber uno más pequeño, pertenecería a la familia que se está intersecando, lo cual es absurdo (¡compruébese!).

De esta forma queda justificada la existencia del menor conjunto convexo que contiene a $\mathcal{S}$, no obstante, la construcción que hemos realizado nos da escasa información acerca de los elementos de la envoltura convexa. $\diamondsuit$

Observación 1.2.2 (Curiosidad)**.** A modo de curiosidad comentamos que el problema de calcular la envoltura convexa de un conjunto finito de n puntos en el plano o el espacio es uno de los problemas centrales de la ***geometría computacional*** por sus múltiples aplicaciones. $\diamondsuit$

Veamos a continuación una caracterización útil de la envoltura convexa de un conjunto, para la cual necesitamos introducir una definición.

Definición 1.2.2 (Combinaciones convexas)**.** Sea $\{z_1, \ldots, z_r\}$ un conjunto finito de puntos. Diremos que z es ***combinación lineal convexa*** de dichos puntos si

$$z = \lambda_1 z_1 + \cdots + \lambda_r z_r$$

donde $\lambda_i > 0$ para $i \in \{1, \ldots, r\}$ y se cumple que $\sum_{i=1}^{r} \lambda_i = 1$.

Que no asuste la longitud de la demostración de la proposición pues realmente es muy sencilla.

Proposición 1.2.3 (Caracterización)**.** *Dado un conjunto $\mathcal{S}$, su envoltura convexa es el conjunto de puntos que son una combinación lineal convexa de puntos de $\mathcal{S}$.*

Demostración. Llamemos $\mathcal{H}(\mathcal{S})$ al conjunto de puntos que son combinación lineal convexa de puntos de $\mathcal{S}$. Veamos en primer lugar que $\mathcal{H}(\mathcal{S})$ contiene a $\mathcal{S}$. Esto es claro, ya que todo punto x de $\mathcal{S}$ es combinación lineal convexa de puntos de $\mathcal{S}$, en efecto, $x = 1 \cdot x$.

Veamos ahora que $\mathcal{H}(\mathcal{S})$ es convexo. Esto lo haremos simplemente usando la definición. Sean x e y dos puntos de $\mathcal{H}(\mathcal{S})$, veamos que $\alpha x + (1-\alpha)y \in \mathcal{H}(\mathcal{S})$ para todo $\alpha \in [0, 1]$.

Por definición de $\mathcal{H}(\mathcal{S})$

$$x = \textstyle\sum_{i=1}^{r} \lambda_i x_i \qquad y = \sum_{j=1}^{s} \mu_j y_j$$

por tanto, si sustituimos esto en $\alpha x + (1-\alpha)y$ obtenemos

$$\alpha x + (1-\alpha)y = \sum_{i=1}^{r} (\alpha\lambda_i) x_i + \sum_{j=1}^{s} [(1-\alpha)\mu_j] y_j$$

Haciendo los siguientes cambios de nombre

$$\xi_k := \begin{cases} \alpha\lambda_k & \text{si } k \in \{1, \ldots, r\} \\ (1-\alpha)\mu_{k-r} & \text{si } k \in \{r+1, \ldots, r+s\} \end{cases} \qquad z_k := \begin{cases} x_k & \text{si } k \in \{1, \ldots, r\} \\ y_{k-r} & \text{si } k \in \{r+1, \ldots, r+s\} \end{cases}$$

nos queda que

$$\alpha x + (1-\alpha)y = \sum_{k=1}^{r+s} \xi_k z_k$$

luego por definición de $\mathcal{H}(\mathcal{S})$ solo queda comprobar que $\sum_{k=1}^{r+s} \xi_k = 1$, pero esto es evidente ya que

$$\sum_{k=1}^{r+s} \xi_k = \alpha \sum_{i=1}^{r} \lambda_i + (1-\alpha) \sum_{j=1}^{s} \mu_j = \alpha + (1-\alpha) = 1$$

Queda por ver pues que $\mathcal{H}(\mathcal{S})$ es el menor conjunto convexo que contiene a $\mathcal{S}$. Para ello, consideramos $\mathcal{C}$ un conjunto convexo arbitrario que contiene a $\mathcal{S}$. Demostremos que $\mathcal{H}(\mathcal{S}) \subset C$ por inducción sobre la "longitud" de la combinación lineal convexa de sus elementos.

El caso base es evidente, ya que los elementos de $\mathcal{H}(\mathcal{S})$ que son combinaciones lineales convexas de longitud unitaria son también elementos de $\mathcal{S}$ y por tanto de $\mathcal{C}$. Supongamos cierto que los elementos de $\mathcal{H}(\mathcal{S})$ que son combinaciones lineales convexas de longitud n viven también en $\mathcal{C}$, y demostremos que lo mismo sucederá para los elementos que sean combinaciones de longitud $n+1$.

Consideremos $z \in \mathcal{H}(\mathcal{S})$ un elemento que es combinación de longitud $n+1$, es decir

$$z = \textstyle\sum_{i=1}^{n+1} \lambda_i z_i \quad \text{con} \quad \sum_{i=1}^{n+1} \lambda_i = 1$$

Vamos a apañar la expresión de z para poder aplicar la hipótesis de inducción

$$z = \sum_{i=1}^{n} \lambda_i z_i + \lambda_{n+1} z_{n+1}$$

Es claro que $z_{n+1} \in \mathcal{C}$ por ser una combinación de longitud uno. Sin embargo, el primer sumando no es una combinación lineal convexa ya que $\sum_{i=1}^{n} \lambda_i = 1 - \lambda_{n+1} \neq 1$. Para arreglar esto vamos a multiplicar y a dividir el primer sumando por $1 - \lambda_{n+1}$

$$z = (1 - \lambda_{n+1}) \sum_{i=1}^{n} \frac{\lambda_i}{1 - \lambda_{n+1}} z_i + \lambda_{n+1} z_{n+1}$$

De esta forma el primer sumando (excluyendo el factor que lo multiplica) es una combinación lineal convexa de longitud n, ya que $\frac{1}{1\lambda_{n+1}} \sum_{i=1}^{n} \lambda_i = \frac{1-\lambda_{n+1}}{1-\lambda_{n+1}} = 1$.

Por hipótesis de inducción, tanto el primer sumando como el segundo (excluyendo los factores que los multiplican) son elementos de $\mathcal{C}$, y como $\mathcal{C}$ es convexo $z \in \mathcal{C}$, como queríamos. ■

1.2.2. Puntos extremos

De ahora en adelante será de utilidad tener en mente un polígono regular cuando se nos hable de conjuntos convexos, pues nos dará una idea bastante intuitiva.

Para estrenar esta nueva buena costumbre se presenta la siguiente definición.

Definición 1.2.3 (Punto extremo)**.** Sea $\mathcal{C}$ un conjunto convexo. Diremos que $\overline{x} \in \mathcal{C}$ es un ***punto extremo*** de $\mathcal{C}$ si se verifica que para cualquier par de puntos $x_1, x_2 \in \mathcal{C}$, $\overline{x}$ **no** está en el segmento generado por dichos puntos (excluyendo los extremos). Es decir

$$\overline{x} = \lambda x_1 + (1 - \lambda) x_2 \iff \begin{cases} \lambda \in \{0, 1\} \\ x_1 = x_2 \end{cases}$$

La ilustración 1.2 marca en negro los puntos extremos de un polígono regular, para coger una idea intuitiva. A lo largo de este capítulo veremos la importancia que tienen estos puntos en la resolución del problema de programación lineal. El teorema 1.2.4 nos da una caracterización de los

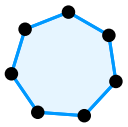

Figura 1.2: Ilustración de los puntos extremos de un heptágono regular.

puntos extremos de una región factible que, aunque muy potente, resulta poco intuitiva. Antes de enunciarlo recordemos un viejo resultado de álgebra lineal.

Observación 1.2.3 (Rango y submatrices)**.** Es conocido desde tiempos inmemoriales que $A \in \mathfrak{M}_{m \times n}(\mathbb{K})$ con $m \leq n$ y $\operatorname{rg}(A) = m$ entonces habrá al menos una submatriz cuadrada $B \in \mathfrak{M}_m(\mathbb{K})$ de A tal que $\operatorname{rg}(B) = m$. De esta forma, reordenando las columnas de A si fuera necesario, podemos hacer la siguiente división por bloques de A.

$$A = (B|N)$$

Nótese que la igualdad se da solo cuando ya hemos reordenado A. ♢

A partir de ahora trabajaremos con matrices en las condiciones de la observación 1.2.3 a no ser que se especifique lo contrario. Nótese que esto no supone ninguna pérdida de generalidad en lo que se refiere al estudio de los problemas de programación lineal ya que si se nos presentara una matriz con más filas que columnas habrá restricciones redundantes.

Definición 1.2.4 (Bases)**.** Dado un problema en forma estándar que tiene a A por matriz de coeficientes de las restricciones, llamaremos ***base*** de A a toda submatriz suya cuadrada y de rango máximo.

Teorema 1.2.4 (Caracterización). *Dada una región factible $\mathcal{S}$, $\overline{x}$ es un punto extremo de $\mathcal{S}$ si y solo si $\overline{x} \geq 0$ y hay alguna reordenación de las columnas de A de forma que*

$$A = (B|N) \qquad \overline{x} = \begin{pmatrix} \overline{x_B} \\ \hline \overline{x_N} \end{pmatrix} \text{ con } \overline{x_N} = 0$$

donde B es una base de A y $\overline{x_B}$ y $\overline{x_N}$ son los "trozos" de $\overline{x}$ acordes con la descomposición en bloques de A, es decir

$$A\overline{x} = (B|N)\begin{pmatrix} \overline{x_B} \\ \hline \overline{x_N} \end{pmatrix} = B\overline{x_B} + N\overline{x_N}$$

Demostración. Veamos las dos implicaciones

$\boxed{\Longleftarrow}$ Supongamos que $\overline{x}$ no es punto extremo, entonces habrá dos puntos distintos $x_1, x_2 \in \mathcal{S}$ tales que $\overline{x} = \lambda x_1 + (1-\lambda)x_2$ con $\lambda \in (0,1)$. Desplegando la descomposición por bloques tendríamos que

$$(\overline{x_B}|\overline{x_N})^t = \lambda(x_B^1|x_N^1)^t + (1-\lambda)(x_B^2|x_N^2)^t$$

Como por hipótesis $\overline{x_N} = 0$ y tanto λ como $1-\lambda$ son estrictamente positivos se tiene que $x_N^1 = x_N^2 = 0$, ya que x_1 y x_2 deben ser no negativos.

Como $x_1, x_2 \in \mathcal{S}$, entonces $Ax_i = b$ para $i \in \{1,2\}$, por tanto

$$(B|N)(x_B^i|x_N^i)^t = (B|N)(x_B^i|0)^t = Bx_B^i = b \implies x_B^i = B^{-1}b$$

Con lo que se concluye que $x_1 = x_2$, lo cual es absurdo.

$\boxed{\Longrightarrow}$ Sea $\overline{x}$ un punto extremo. Reordenemos el vector de forma que todas sus componentes nulas queden al final, reordenando solidariamente las columnas de la matriz A.

Podemos suponer pues que el vector $\overline{x}$ tiene $p \neq 0$ componentes no nulas (si $p = 0$ el resultado es evidente). Vamos a demostrar que las p primeras columnas de A (ya reordenada) son linealmente independientes.

A partir de ahora denotaremos a la matriz A por sus columnas, es decir $A = (a_1 \cdots a_n)$ donde a_i representa la i–ésima columna de A.

Procedamos por reducción al absurdo, es decir, supongamos que hay ciertos números reales no todos nulos λ_i tales que $\sum_{i=1}^{p} \lambda_i a_i = 0$.

Definimos el vector $\lambda := (\lambda_1, \ldots, \lambda_p, 0, \ldots, 0) \in \mathbb{R}^n$. A continuación consideramos los vectores **no negativos** $x_1 := \overline{x} + \alpha\lambda$ y $x_2 := \overline{x} - \alpha\lambda$ tomando un α lo suficientemente pequeño para que ambos vectores sean precisamente no negativos, lo cual siempre puede hacerse al ser $\overline{x}$ no negativo.

Los vectores x_1 y x_2 tienen la particularidad de que con ellos podemos obtener $\overline{x}$ como combinación lineal convexa. En efecto $\overline{x} = \frac{1}{2}x_1 + \frac{1}{2}x_2$. Si resultara que x_1 y x_2 son puntos de la región factible estaríamos entrando en contradicción con la "extremalidad" de $\overline{x}$, y eso es exactamente lo que vamos a hacer. En efecto, con $i \in \{1,2\}$ tenemos

$$Ax_i = A\overline{x} \pm \alpha A\lambda \overset{!}{=} A\overline{x} = b \implies x_i \in \mathcal{S}$$

donde la igualdad de la exclamación se debe a la composición de λ y la dependencia lineal de las p primeras columnas de A (por hipótesis). Con esto llegamos a un absurdo y concluimos que las p primeras columnas de A son linealmente independientes.

Además, de paso podemos concluir que $p \leq m$, ya que m es el rango máximo de A. Si además $p = m$ ya tenemos una base $B = (a_1 \cdots a_p)$ que cumple lo que queremos (¡compruébese!). En el caso de que $p < m$ la observación 1.2.3 nos dice que siempre podemos escoger columnas adicionales de A para completar una base y, tras una nueva reordenación de A si fuera necesario tendríamos lo que necesitamos. ■

Corolario 1.2.5 (Parte básica). *Dado un punto extremo $\overline{x} = (\overline{x_B}|0)^t$ de una región factible $\mathcal{S}$ se cumple que $\overline{x_B} = B^{-1}b \geq 0$.*

Demostración. Basta con echar las cuentas. Como $\overline{x} \in \mathcal{S}$ es claro que $A\overline{x} = b$, luego

$$A\overline{x} = (B|N)(\overline{x_B}|0)^t = B\overline{x_B} = b$$

y despejando se tiene que $\overline{x_B} = B^{-1}b$. ■

Al hilo de este teorema surge la siguiente definición.

Definición 1.2.5 (Soluciones básicas)**.** A los puntos extremos de una región factible se les suele llamar ***soluciones básicas factibles***. Esto es debido a que, por el teorema 1.2.4, están "asociados" a una base de la matriz asociada a la región factible del problema.

Observación 1.2.4 (Procedimiento constructivo)**.** Nótese que si nos dan un punto extremo de una región factible y nos piden hallar la base asociada a dicho punto extremo no tenemos más que seguir la demostración de la implicación a la derecha del teorema 1.2.4 para encontrarla, es decir, tomar las columnas de A asociadas a las componentes no nulas del punto extremo y, en su caso, completar la base.

Esto nos lleva a deducir de forma evidente que cada punto extremo puede estar asociado a más de una base (salvo reordenaciones). ♢

Una de las razones por la cual es estudio de los puntos extremos es importante es porque toda región factible de un problema de programación lineal en forma estándar tiene al menos un punto extremo, y además únicamente tiene una cantidad finita de ellos.

Observación 1.2.5 (Finitud)**.** Una región factible tiene un número finito de puntos extremos, esto es debido a que A tiene más bases que puntos extremos (hay una sobreyección que parte del conjunto de las bases y llega al conjunto de puntos extremos). Además, cada matriz tiene a lo sumo $\binom{n}{m}$ bases (salvo reordenaciones). ♢

Teorema 1.2.6 (Existencia)**.** *Toda región factible no vacía posee al menos un punto extremo.*

Demostración. Sea $\overline{x}$ un punto arbitrario de la región factible. Reordenemos sus componentes de forma que las componentes nulas queden al final, asimismo reordenemos A de forma coherente con la reordenación de $\overline{x}$. Supongamos que $\overline{x}$ posee p componentes no nulas (las p primeras tras la reordenación).

Ahora consideramos las columnas de A correspondientes a las p componentes no nulas de $\overline{x}$. Si estas columnas son linealmente independientes, por el teorema 1.2.4 habríamos terminado.

Si, por el contrario, las columnas (que serán las p primeras tras la reordenación) son linealmente dependientes, entonces existirán ciertos números reales λ_i no todos nulos tales que $\sum_{i=1}^{p} \lambda_i a_i = 0$, donde a_i representa a la i–ésima columna de A.

Lo que vamos a hacer es cambiar ligeramente el punto $\overline{x}$ de forma que siga siendo de la región factible y conserve un subconjunto estrictamente menor de las componentes no nulas de $\overline{x}$.

En primer lugar cabe mencionar que siempre podemos suponer que existe al menos un $r \in \{1, \ldots, p\}$ de forma que $\lambda_p > 0$, ya que en caso de que todos los números de la combinación lineal fueran negativos podemos considerar la combinación lineal alternativa $\sum_{i=1}^{p} -\lambda_i a_i = 0$, en la cual todos los escalares son positivos.

Definimos ahora el vector $\lambda := (\lambda_1, \ldots, \lambda_p, 0, \ldots, 0) \in \mathbb{R}^n$ y consideramos el puntos $\overline{x}' := \overline{x} - \alpha\lambda$ para cierto $\alpha > 0$ que definiremos más adelante.

El punto $\overline{x}'$ verifica la condición $Ax = b$, en efecto

$$A\overline{x}' = A\overline{x} - \alpha A\lambda = A\overline{x} = b$$

Luego únicamente queda elegir un α adecuado para que $\overline{x}' \geq 0$. Para ello elegiremos

$$\alpha := \text{mín}\left\{\frac{\overline{x}_j}{\lambda_j} \ : \ \lambda_j > 0\right\} = \frac{\overline{x}_l}{\lambda_l}$$

Veamos que $\overline{x}'$ es no negativo. Para ello distinguiremos casos componente a componente.

- Si $\lambda_j > 0$ tenemos que $\overline{x}'_j = \overline{x}_j - \frac{\overline{x}_l}{\lambda_l}\lambda_j = \frac{\overline{x}_j}{\lambda_j}\lambda_j - \frac{\overline{x}_l}{\lambda_l}\lambda_j = \lambda_j\left(\frac{\overline{x}_j}{\lambda_j} - \frac{\overline{x}_l}{\lambda_l}\right)$ y este número es no negativo por definición de α.

- Si $\lambda_j \leq 0$ evidentemente $\overline{x}'_j = \overline{x}_j - \alpha\lambda_j$ es no negativo por ser la suma de números no negativos.

Cabe destacar que si $\overline{x}_j = 0$ entonces $\overline{x}'_j = 0$, además $\overline{x}_l > 0$ mientras que $\overline{x}'_l = 0$, luego $\overline{x}'$ tiene a lo sumo $p-1$ componentes no nulas, con lo que, al volver al principio de la demostración el conjunto de columnas que queremos que sean linealmente independientes se ha visto reducido en una columna como poco. Por tanto, tras un número finito de pasos encontraremos un punto extremo. Esto proporciona un algoritmo para encontrar un punto extremo de una región factible. ■

Culminamos este teorema con una pequeña observación.

Observación 1.2.6 (Extremalidad del 0)**.** Si una región factible contiene al vector 0, este es siempre un punto extremo, basta con aplicar el teorema 1.2.6 para comprobarlo (el conjunto vacío es linealmente independiente). ♢

El siguiente teorema termina de mostrar la brutal importancia de los puntos extremos de las regiones factibles en los problemas de programación lineal, y es que, si un problema tiene solución óptima, esta se alcanza en un punto extremo o solución básica factible.

Teorema 1.2.7 (Optimalidad)**.** *Si un problema de programación lineal en forma estándar tiene solución óptima, esta se alcanza en un punto extremo.*

Demostración. Procedemos de análogamente a como hicimos en la demostración del teorema 1.2.6.

Consideremos $\overline{x}$ un punto donde se alcanza la solución óptima al problema en cuestión. Reordenemos las coordenadas de $\overline{x}$ dejando las componentes nulas al final, reordenando también A solidariamente.

Suponiendo que $\overline{x}$ tiene p componentes no nulas y que las columnas de A correspondientes a dichas columnas son linealmente dependientes (en caso contrario hemos terminado), entonces habrá ciertos números reales no todos nulos (de hecho podemos suponer que alguno será positivo) tales que $\sum_{i=1}^{n} \lambda_i a_i = 0$. Definimos el vector $\lambda \in \mathbb{R}^n$ de la manera usual y consideramos los puntos

$$x_1 = \overline{x} + \alpha_1\lambda \qquad x_2 = \overline{x} - \alpha_2\lambda$$

siendo α_1 y α_2 coeficientes positivos lo suficientemente pequeños para que x_1 y x_2 sean no negativos.

Es evidente que tanto x_1 como x_2 cumplen la condición $Ax = b$, esto se debe a la dependencia lineal de las p primeras columnas y a la definición de λ, como ya hemos hecho en otras ocasiones.

Evaluemos los puntos x_1 y x_2 por la función objetivo

$$c^t x_1 = c^t\overline{x} + \alpha_1 c^t\lambda$$
$$c^t x_2 = c^t\overline{x} - \alpha_2 c^t\lambda$$

Por la optimalidad de $\overline{x}$ se tiene que

$$c^t\overline{x} \leq c^t\overline{x} + \alpha_1 c^t\lambda \tag{1.4}$$
$$c^t\overline{x} \leq c^t\overline{x} - \alpha_2 c^t\lambda \tag{1.5}$$

De la ecuación (1.4) se deduce que $c^t\lambda \geq 0$ y de la ecuación (1.5) se desprende que $c^t\lambda \leq 0$, luego $c^t\lambda = 0$, por tanto tanto x_1 como x_2 son puntos en los que también se alcanza la solución óptima. Centrémonos en x_2. Si tomamos α_2 como

$$\alpha_2 := \text{mín}\left\{\frac{\overline{x}_j}{\lambda_j} \ : \ \lambda_j > 0\right\}$$

sabemos que x_2 sigue perteneciendo a la región factible (como demostramos en el teorema 1.2.6) luego es una elección válida. Además, por las mismas razones que en el teorema 1.2.6, el vector x_2 tiene a lo sumo $p-1$ componentes positivas, luego, tras un número finito de pasos encontraremos un punto extremo en el cual se alcance la solución óptima. ■

1.2.3. Direcciones extremas

El siguiente es el último concepto relativo a convexidad que veremos, se trata del concepto de "dirección". En palabras llanas podríamos decir que una "dirección" de un conjunto convexo es una flecha tal que si partimos de cualquier punto del conjunto y la seguimos nunca saldremos del conjunto. la siguiente definición formaliza este concepto.

Definición 1.2.6 (Dirección)**.** Dado un conjunto convexo $\mathcal{C}$ y un vector $d \in \mathbb{R}^n$ diremos que d es una ***dirección*** de $\mathcal{C}$ si para todo punto $x \in \mathcal{C}$ se verifica que

$$x + \lambda d \in \mathcal{C}$$

para todo λ no negativo. Asimismo, dadas dos direcciones d_1 y d_2 diremos que son ***equivalentes*** si son vectores proporcionales.

Observación 1.2.7 (No acotación)**.** Nótese que si un conjunto convexo es acotado no puede tener direcciones (compruébese). Luego cuando hablemos de direcciones será recomendable tener en mente conjuntos como el que representa la ilustración 1.3 (que continúa infinitamente hacia arriba y hacia la derecha). ♢

Veamos a continuación un concepto análogo al de los puntos extremos pero en el ámbito de las direcciones.

Definición 1.2.7 (Dirección extrema)**.** Sea d una dirección. Diremos que d es una ***dirección extrema*** si no puede escribirse como combinación lineal positiva de dos direcciones no equivalentes. De forma sintética

$$d = \lambda d_1 + \mu d_2 \text{ con } \lambda, \mu > 0 \iff d_1 \sim d_2$$

donde $\sim$ denota la equivalencia de direcciones.

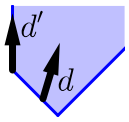

Figura 1.3: Ilustración de una dirección y una dirección extrema

Veamos una caracterización muy sencilla de las direcciones para regiones factibles.

Lema 1.2.8 (Caracterización débil)**.** *Dada una región factible $\mathcal{S}$, d es una dirección de $\mathcal{S}$ si y solo si d es un vector no negativo y no nulo que verifica que $Ad = 0$.*

Demostración. Veamos ambas implicaciones.

$\boxed{\Longrightarrow}$ Sea d una dirección, por tanto, dado $x \in \mathcal{S}$ y $\lambda \geq 0$ se cumple que $x + \lambda d \in \mathcal{S}$, luego $A(x + \lambda d) = b$, por tanto

$$A(x + \lambda d) = Ax + \lambda Ad = b + \lambda Ad = b \iff Ad = 0$$

Queda comprobar que d es un vector no negativo, para ello supongamos que tiene alguna componente negativa, digamos la i–ésima. Entonces por hipótesis $x_j + \lambda d_j \geq 0$ para todo $\lambda \geq 0$, sin embargo, tomando un λ lo suficientemente grande llegamos a un absurdo.

$\boxed{\Longleftarrow}$ Es claro que $x + \lambda d$ es un vector no negativo para todo $x \in \mathcal{S}$ (por hipótesis). Comprobemos que está en la región factible. En efecto

$$A(x + \lambda d) = Ax + \lambda Ad = Ax = b$$ ■

El lema anterior nos abre las puertas a una caracterización de las direcciones extremas mucho muy potente y extremadamente parecida al teorema 1.2.4, es decir, muy poco intuitiva. Conviene reflexionar sobre el enunciado antes de lanzarse a la demostración.

Teorema 1.2.9 (Caracterización fuerte)**.** *Dada una región factible $\mathcal{S}$, d es una dirección extrema de $\mathcal{S}$ si y solo si, tras una conveniente reordenación, hay una base B de A tal que*

$$A = (B|N) \qquad d = \alpha \begin{pmatrix} -B^{-1}a_k \\ e_t \end{pmatrix}$$

donde a_k es una columna de N, o sea, la $k = m + t$–ésima columna de A.

Además, $\alpha > 0$, $B^{-1}a_k \leq 0$ y e_t es el t–ésimo vector de la base canónica de $\mathbb{R}^{n-m}$.

Demostración. Veamos ambas implicaciones.

$\boxed{\Longleftarrow}$ Sea d un vector que cumple las propiedades del enunciado, luego no negativo y no nulo. Veamos en primer lugar que es una dirección de la región factible. En efecto, usando la única caracterización que tenemos

$$Ad = (B|N)(-B^{-1}a_k|e_t)^t = -BB^{-1}a_k + Ne_t = -a_k + a_k = 0$$

Para comprobar la extremalidad de d, supongamos que puede escribirse como combinación lineal positiva de dos direcciones d_1 y d_2 y veamos que estas direcciones son equivalentes.

Como $d = \lambda_1 d_1 + \lambda_2 d_2$, si nos fijamos únicamente en las $n - m$ últimas coordenadas se tiene que $e_t = \lambda_1 d_{1N} + \lambda_2 d_{2N}$, luego, por ser los vectores d_1 y d_2 no negativos, sus $n - m$ últimas componentes deben ser múltiplos de e_t. Es decir $d_{iN} = \mu_i e_t$ con $i \in \{1, 2\}$.

Fijémonos ahora en las m primeras componentes. Al ser los vectores d_i con $i \in \{1, 2\}$ direcciones se cumplirá que $Ad_i = 0$. Desarrollando esta cuenta tenemos

$$Ad_i = (B|N)(d_{iB}|\mu_i e_t)^t = Bd_{iB} + \mu_i Ne_t = Bd_{iB} + \mu_i a_k = 0$$

despejando obtenemos que $d_{iB} = -\mu_i B^{-1}a_k$ con lo que ambas direcciones son equivalentes.

$\boxed{\Longrightarrow}$ Sea d una dirección extrema de $\mathcal{S}$, por tanto un vector no negativo y no nulo. Podemos suponer que d tiene $p + 1$ componentes positivas con $p \in \{0, \ldots, n - 1\}$. En primer lugar, procedemos a reordenar el vector de forma que sus p primeras componentes sean positivas, colocando la última componente positiva en la posición $k > m$ (cualquiera donde quepa).

Veamos que las columnas correspondientes a las p primeras componentes del vector d reordenado son linealmente independientes. En caso contrario existirá una combinación lineal no trivial tal que $\sum_{i=1}^{p} \lambda_i a_i = 0$. Dicho esto, vamos a tratar de construir dos direcciones de $\mathcal{S}$ en función de las cuales d pueda escribirse como combinación lineal positiva, llegando a una contradicción con la extremalidad de d.

Definiendo el vector $\lambda := (\lambda_1, \ldots, \lambda_p, 0, \ldots, 0)$ y los vectores

$$d_1 = d + \delta\lambda \qquad d_2 = d - \delta\lambda$$

siendo δ suficientemente pequeño para que d_1 y d_2 sean no negativos. Nótese que $d = \frac{1}{2}d_1 + \frac{1}{2}d_2$, luego si d_1 y d_2 son direcciones tendríamos una contradicción. Y, en efecto lo son, basta echar las cuentas

$$Ad_i = A(d \pm \delta\lambda) = Ad \pm \delta A\lambda = 0$$

De esta forma se llega a la conclusión de que necesariamente $p \leq m$, lo cual es importante, pues nos dice que si un vector tiene más de m componentes positivas este no puede ser dirección extrema.

En definitiva, si $p = m$ ya tenemos una base B de A, en caso contrario siempre podemos coger columnas adicionales para completar la base (observación 1.2.3). Veamos pues que se cumple lo que se tiene que cumplir, para lo cual basta con, de nuevo, hacer las cuentas.

$$0 = Ad = (B|N)(d_B|d_N)^t = Bd_B + Nd_N = Bd_B + a_k d_k$$

despejando obtenemos que $d_B = -d_k B^{-1}a_k$ y evidentemente $d_N = d_k e_t$, luego tomando $\alpha := d_k$ se tiene el resultado. ∎

Observación 1.2.8 (Finitud)**.** Sobre la finitud de las direcciones extremas de una región factible sigue siendo válida con muy pocas modificaciones (que se dejan al lector) la observación 1.2.5. ♢

Terminado nuestro viaje por el mundo de la convexidad, podemos internarnos ya en el meollo del asunto, es decir, en desarrollar material teórico que nos permita desarrollar un algoritmo para optimizar funciones lineales condicionadas a variedades afines en el 2^n–ante positivo.

1.3. Teoremas fundamentales

Empezamos esta sección enunciando un teorema que no demostraremos, no porque sea especialmente complicado, sino porque es especialmente largo y su demostración no aporta demasiado a nuestros objetivos. Hablamos del famoso "teorema de representación".

Teorema 1.3.1 (Teorema de representación)**.** *Sea $\mathcal{S}$ una región factible con p puntos extremos $\{x_1, \ldots, x_p\}$ y l direcciones extremas $\{d_1, \ldots, d_l\}$. Un punto x está en la región factible si y solo si x es "combinación esotérica" de puntos y direcciones extremas, es decir*

$$x = \sum_{i=1}^{p} \lambda_i x_i + \sum_{j=1}^{l} \mu_j d_j$$

con $\lambda_i, \mu_j \geq 0$ y $\sum_{i=1}^{p} \lambda_i = 1$.

El siguiente teorema es un test de optimalidad para regiones factibles que poseen direcciones extremas.

Teorema 1.3.2 (Test de optimalidad)**.** *Dado un problema de programación lineal con región factible no acotada y direcciones extremas $\{d_1, \ldots, d_s\}$, el problema tiene solución optima si y solo si se cumple que $c^t d_j \geq 0$ para todo $j \in \{1, \ldots, s\}$.*

Demostración. Demostremos ambas implicaciones echando mano del teorema de representación.

$\boxed{\Longrightarrow}$ Por reducción al absurdo, si la función objetivo alcanzara la solución óptima y hubiera una dirección extrema d_r tal que $c^t d_r < 0$ podemos considerar el punto de la región factible (por el teorema de representación)

$$x_{\mu_r} = \sum_{i=1}^{p} \lambda_i x_i + \mu_r d_r$$

Veamos cómo evoluciona el valor de la función objetivo para valores grandes de μ_r

$$\lim_{\mu_r \to \infty} c^t x_{\mu_r} = \lim_{\mu_r \to \infty} \sum_{i=1}^{p} \lambda_i c^t x_i + \mu_r c^t d_r = \lim_{\mu_r \to \infty} \mu_r c^t d_r = -\infty$$

luego es un problema con solución no acotada (absurdo por hipótesis).

$\boxed{\Longleftarrow}$ Dado un punto x de la región factible por el teorema de representación se verifica que la función objetivo evaluada en x es

$$c^t x = \sum_{i=1}^{p} \lambda_i c^t x_i + \sum_{j=1}^{s} \mu_j c^t d_j$$

el resultado que arroja el segundo sumatorio es positivo, luego si consiguiéramos fusilarlo mejoraríamos el valor de la función objetivo. Para ello basta con coger el punto con representación $x' := \sum_{i=1}^{p} \lambda_i x_i$. De esta forma

$$c^t x \geq c^t x' = \sum_{i=1}^{p} \lambda_i c^t x_i$$

más aun, si consideramos el punto extremo x^* tal que $c^t x^* := \text{mín}\{c^t x_i \ : \ 1 \leq i \leq p\}$ tenemos la cadena de desigualdades

$$c^t x \geq c^t x' \geq \sum_{i=1}^{p} \lambda_i c^t x^* = c^t x^*$$

de esta forma se tiene que x^* es solución óptima del problema, ya que la función objetivo evaluada en ese punto es menor o igual a la de cualquier otro. ■

Observación 1.3.1 (No acotación)**.** La hipótesis de no acotación del teorema 1.3.2 no es necesaria para la demostración de la implicación a la izquierda.

La demostración de la implicación a la derecha exige la existencia de al menos una dirección extrema, y por tanto la no acotación de la región factible. Esta implicación constituye un criterio de no acotación de la función objetivo en regiones factibles no acotadas. ♢

Observación 1.3.2 (Algoritmo de representación)**.** Con todo lo que tenemos hasta ahora ya tenemos un algoritmo (algoritmo 1) para resolver problemas de programación lineal. Basta con calcular los puntos y direcciones extremas de la región factible, lo cual puede hacerse usando los teoremas 1.2.4 y 1.2.9 y seguir la demostración del test de optimalidad (teorema 1.3.2).

La eficiencia de este algoritmo es pésima pues obliga a calcular todos los puntos extremos, con lo que la eficiencia el algoritmo sería $\mathcal{O}\binom{n}{m}$ siendo n el número de variables de decisión y m el número de restricciones. ♢

Algoritmo 1 Algoritmo de representación.

```
Calcular los puntos y direcciones extremas.
si no hay puntos extremos entonces
    devolver problema infactible.
si no, si no hay direcciones extremas entonces
    devolver el mejor punto extremo.
si no
    {Region no acotada.}
    Comprobar las hipótesis del teorema 1.3.2.
    si hay solución óptima entonces
        devolver el mejor punto extremo.
    si no
        devolver x_{μ_r}, dirección en la que las soluciones decrecen.
    fin si
fin si
```

La pésima eficiencia del algoritmo basado en el teorema de representación incita seguir desarrollando soporte teórico para un algoritmo mejor.

El siguiente resultado actúa a modo de recopilación de todo lo visto hasta ahora y se presenta más que nada por su recurrente aparición en la literatura como un teorema con nombre propio.

Teorema 1.3.3 (Teorema fundamental de la programación lineal)**.** *Dado un problema de programación lineal en forma estándar se cumplen los siguientes asertos*

1. *Si el problema es factible, posee al menos una solución básica factible.*
2. *Si el problema tiene solución óptima, posee al menos una solución básica factible óptima.*
3. *Si el problema no tiene solución óptima, o bien es infactible o bien tiene solución no acotada.*

Demostración. El primer aserto se corresponde con el contenido del teorema 1.2.6. La segunda afirmación se demuestra en el teorema 1.2.7.

El último apartado es el que tiene algo más de chicha. Distinguimos tres casos según la naturaleza de la región factible. Si la región factible es vacía el problema es, por definición, infactible. En el caso de que la región factible sea acotada, es también compacta.

Esto se debe a que es la imagen inversa de $\{b\}$ (cerrado) vía la función lineal (continua) que tiene a A por matriz asociada, luego la región factible es cerrada y acotada, por tanto, la función objetivo alcanza el mínimo en la región factible y el problema tiene solución óptima (se sale de nuestros casos de estudio).

Si la región factible es no acotada basta con aplicar el teorema 1.3.2. ■

A continuación vamos a presentar un nuevo test de optimalidad de una solución factible que no requiere del cálculo de todas las direcciones extremas. Antes de enunciarlo presentamos la siguiente definición.

Definición 1.3.1 (Costes reducidos)**.** Definimos el ***vector de costes reducidos*** asociado a una base B como el vector $\overline{c} \in \mathbb{R}^n$ cuyas componentes son

$$\overline{c_j} := c_j - c_B^t B^{-1} a_j$$

justificaremos el nombre y la razón de ser de esta cruel definición tras la demostración del teorema 1.3.4, que consideramos bastante importante.

Teorema 1.3.4 (Test de optimalidad)**.** *Si $\overline{x}$ es un punto extremo de $\mathcal{S}$ asociado a la base B y el vector de costes reducidos asociado a B es no negativo entonces $\overline{x}$ es una solución óptima.*

Demostración. Sea x un punto arbitrario de $\mathcal{S}$. Vamos a demostrar que $c^t x \geq c^t \overline{x}$, con lo cual quedaría demostrado el resultado.

Como $x \in \mathcal{S}$ se verifica que $Ax = b$. Expresando esto en términos de la base B tenemos que

$$Ax = (B|N)(x_B|x_N)^t = Bx_B + Nx_N = b \iff x_B = B^{-1}b - B^{-1}Nx_N \tag{1.6}$$

Pasemos x por la función objetivo

$$\begin{aligned} c^t x = (c_B^t|c_N^t)(x_B|x_N)^t = c_B^t x_B + c_N^t x_N = \\ = c_B^t(B^{-1}b - B^{-1}Nx_N) + c_N^t x_N = c_B^t B^{-1}b + (c_N^t - c_B^t B^{-1}N)x_N = \\ = c_B^t B^{-1}b + \left(c_N^t x_N - c_B^t \sum_{m+1}^{n} (B^{-1}a_j)x_j\right) = c_B^t B^{-1}b + \sum_{j=m+1}^{n} (c_j x_j - c_B^t B^{-1} a_j x_j) = \\ = c_B^t B^{-1}b + \sum_{j=m+1}^{n} (c_j - c_B^t B^{-1}a_j)x_j = c_B^t B^{-1}b + \sum_{j=m+1}^{n} \overline{c_j} x_j \end{aligned} \tag{1.7}$$

Como el último sumatorio arroja por hipótesis un resultado positivo tenemos que

$$c^t x \geq c_B^t B^{-1} b$$

pero resulta que $c_B^t B^{-1} b = c^t \overline{x}$ por el teorema 1.2.4 (compruébese). ■

Observación 1.3.3 (Justificación del nombre)**.** Una interpretación que le podemos dar a las componentes del vector de costes reducidos es la siguiente.

$$\overline{c_j} = c^t e_j - c^t \overline{x} = c^t(e_j - \overline{x})$$

donde e_j es el j–ésimo vector de la base canónica de $\mathbb{R}^n$ y $\overline{x}$ es una solución básica factible (punto extremo).

Lo que quiere decir esta interpretación es que $\overline{c_j}$ es la diferencia entre valor que arroja la función objetivo cuando se la evalúa en un punto extremo y el valor que obtenemos si la evaluamos sobre un vector de la base canónica.

La comprobación de la validez de esta interpretación se hace echando mano de las identidades obtenidas en la demostración del teorema 1.3.4.

Nótese que si $j \leq m$ entonces $\overline{c_j} = 0$ (compruébese). ♢

Teorema 1.3.5 (Test de no acotación)**.** *Dada una región factible no vacía $\mathcal{S}$ y un punto extremos $\overline{x}$ asociado a la base B de manera que hay algún coste reducido $\overline{c_k} < 0$ y se verifica que $B^{-1}a_k \leq 0$, entonces el problema tiene solución no acotada.*

Demostración. Nótese que $k \in \{m+1, \ldots, n\}$ ya que los costes reducidos asociados a las componentes asociadas a B son nulos.

Como por hipótesis $B^{-1}a_k \leq 0$, luego es claro que la región factible posee una dirección extrema, la dada por $d := (-B^{-1}a_k|e_t)^t$ donde $k = m + t$.

De esta forma sabemos que el punto $x_\alpha := \overline{x} + \alpha d \in \mathcal{S}$ para todo $\alpha \geq 0$. Veamos cómo se comporta $c^t x_\alpha$ para valores grandes de α

$$c^t x_\alpha = c^t(\overline{x} + \alpha d) = c^t\overline{x} + \alpha c^t d \overset{!}{=} c_b^t B^{-1}b + \alpha c^t d$$

por otra parte tenemos que

$$c^t d = (c_B^t|c_N^t)(-B^{-1}a_k|e_t)^t = -c_B^t B^{-1}a_k + c_N^t e_t = -c_B^t B^{-1}a_k + c_k = \overline{c_k}$$

de manera que $c^t x_\alpha = c_B^t B^{-1}b + \alpha\overline{c_k}$. Al ser $\overline{c_k}$ negativo claramente $\lim_{\alpha\to\infty} c^t x_\alpha = -\infty$ ■

Observación 1.3.4 (Dirección de decrecimiento)**.** De la demostración del teorema 1.3.5 se deduce que la función objetivo decrece indefinidamente a lo largo de la recta afín de ecuación paramétrica x_α. A esto usualmente lo llamaremos ***dirección de decrecimiento*** de las soluciones.

Algo similar se desprende de la demostración del teorema 1.3.2 con x_{μ_r}. ♢

Definición 1.3.2 (Vector auxiliar)**.** Dada una matriz A y una base B de A, llamamos ***vector auxiliar*** asociado a a_k al vector

$$y_k := B^{-1}a_k \in \mathbb{R}^m$$

La única intención de esta definición es compactar un poco la notación para las direcciones extremas. Cabe destacar que no es necesario aprenderse la fórmula, simplemente hay que entender que y_k es el vector a_k escrito en coordenadas de la base B.

Denotaremos por y_{ik} a la i–ésima componente de y_k.

Presentamos a continuación el que probablemente sea el teorema más importante del curso, cuya demostración es debida a Clara Rodríguez.

Teorema 1.3.6 (Mejora de la solución)**.** *Sea una región factible $\mathcal{S}$ no vacía y un punto extremo $\overline{x}$ asociado a una base B de manera que hay algún coste reducido $\overline{c_k} < 0$ y se verifica que y_k tiene al menos una componente positiva.*

Entonces se tiene que el vector $x' = \overline{x} + \alpha d$ es un punto extremo que mejora la función objetivo respecto a $\overline{x}$, es decir $c^t x' \leq c^t\overline{x}$. Donde

$$\alpha := \text{mín}\left\{\frac{\overline{x_i}}{y_{ik}} \ : \ y_{ik} > 0\right\} =: \frac{\overline{x_l}}{y_{lk}} \qquad d := (-y_k|e_t)^t \ \textit{con } k = m + t$$

Además $c^t x' < c^t\overline{x}$ si y solo si $\overline{x_l} > 0$.

Demostración. Veamos en primer lugar que x' pertenece a la región factible. Es evidente que $Ax' = b$ (se deja al lector hacer la cuenta), luego solo queda comprobar que $x' \geq 0$. Es claro que las componentes de x' que no están asociadas a la base B son todas positivas ya que

$$x' = (x'_B|x'_N)^t = (B^{-1}b|0)^t + \alpha(-y_k|e_t)^t \iff x'_N = \alpha e_t \geq 0$$

por tanto, solo nos debemos preocupar de las componentes asociadas a la base. Veamos que son todas no negativas

$$x'_B = \overline{x_B} - \alpha y_k \geq 0 \iff \overline{x_i} - \alpha y_{ik} \geq 0 \text{ con } i \in \{1, \ldots, m\}$$

esa última condición se verifica si y solo si $\alpha \leq \frac{\overline{x_i}}{y_{ik}}$ para los $y_{ik} > 0$. Y hemos cogido α expresamente para que esto se cumpla, luego x' está en la región factible.

Veamos ahora que x' es un punto extremo, para lo cual basta con demostrar que el conjunto de vectores $(B \setminus a_l) \cup a_k$ es linealmente independiente. Para mostrar consideramos el conjunto de vectores $B \cup a_k$, que es un sistema de generadores de $\mathbb{R}^m$.

Resulta que a_l es combinación lineal de los vectores de $(B \setminus a_l) \cup a_k$ ya que

$$a_k = \sum_{i=1}^{m} y_{ik} a_i = \sum_{\substack{i=1 \\ i \neq l}}^{m} y_{ik} a_i + y_{lk} a_l$$

como $y_{lk} > 0$ podemos despejar a_l. En efecto

$$a_l = \frac{a_k}{y_{lk}} - \sum_{\substack{i=1 \\ i \neq l}}^{m} \frac{y_{ik}}{y_{lk}} a_i$$

de donde se desprende que el conjunto $(B \setminus a_l) \cup a_k$ sigue siendo un sistema de generadores de $\mathbb{R}^m$ que además tiene m elementos, luego es una base.

Veamos ahora que x' es una solución mejor que $\overline{x}$, basta echar las cuentas

$$c^t(\overline{x} + \alpha d) = c^t \overline{x} + \frac{\overline{x_l}}{y_{lk}} c^t d \overset{!}{=} c_B^t B^{-1} b + \frac{\overline{x_l}}{y_{lk}} \overline{c_k} \leq c_B^t B^{-1} b = c^t \overline{x}$$

El "además" es evidente. Con lo que concluye la prueba. ■

Con la ayuda estos últimos teoremas podemos desarrollar un algoritmo (mejor que el basado en el teorema de representación) que nos permite hallar la solución a un problema de programación lineal. Este algoritmo es conocido como "algoritmo del Símplex" (algoritmo 2), desarrollado por el matemático estadounidense ***George Dantzig*** en 1947.

Observación 1.3.5 (Justificación del nombre)**.** El nombre "símplex" que recibe el algoritmo se debe a que las regiones factibles de los problemas de programación lineal en forma estándar tienen estructura geométrica de ***poliedros*** o ***símplices***, o, en inglés ***simplex***. ♢

Algoritmo 2 Primera aproximacíon al algoritmo del Símplex.

Entrada: Punto extremo $\overline{x}$ de la región factible $\mathcal{S}$.
Salida: Punto extremo x' que mejora o iguala a $\overline{x}$.

```
Aplicar el teorema 1.3.4 para detectar la optimalidad de x̄.
si x̄ es solución óptima entonces
    devolver x̄.
si no
    Aplicar el teorema 1.3.5 para detectar la no acotación.
    si el problema es no acotado entonces
        devolver x_α, dirección de decrecimiento de las soluciones.
    si no
        Calcular una solución mejor usando el teorema 1.3.6.
        devolver x', la solución mejorada.
    fin si
fin si
```

Observación 1.3.6 (Acertijos en la oscuridad)**.** El algoritmo 2 deja ciertas lagunas. Por ejemplo, no está claro que la aplicación reiterada del algoritmo termine devolviéndonos una solución óptima (podría quedarse estancado).

Además, tampoco se especifica cómo se calcula el primer punto extremo, ni se establece ninguna regla acerca de cómo calcular los costes reducidos necesarios para aplicar los teoremas.

Este último punto puede dar lugar (con una implementación inocente) a un algoritmo extremadamente ineficiente.

Otro asunto a discutir sería el de, en caso de haber varios costes reducidos negativos, cuál elegir para aplicar el teorema 1.3.6. ♢

Todas estas cuestiones se discutirán en el capítulo siguiente.

Capítulo 2

Algoritmo del Símplex

Este es un capítulo dedicado a aclarar las lagunas que presenta el algoritmo 2, desarrollando ya un algoritmo completo que tenga en cuenta todas las contigencias habidas y por haber, el algoritmo del Símplex.

2.1. Cambios de base

Llegados a este punto nos preguntamos qué datos necesita el algoritmo 2 para ejecutarse. Para obtener la respuesta basta con mirar con cuidado los teoremas en los que se basa, con lo que concluimos que son necesarios

- Las componentes asociadas a la base (o componentes básicas) del punto extremo. Para calcular qué vector sale de la base al mejorar el punto extremo (teorema 1.3.6) y para devolverlas cuando se detecta la optimalidad.
- El vector de costes reducidos asociado a la base del punto extremo para aplicar los tests de optimalidad y no acotación. Además de para decidir qué vector entra a la base para mejorar el punto extremo.
- Los vectores auxiliares y_j para aplicar los tests de no acotación (y en su caso devolver la dirección en la que la función objetivo decrece indefinidamente) y para calcular qué vector sale de la base al mejorar el punto extremo.
- El valor de la función objetivo al ser evaluada en el punto extremo, para devolverlo cuando se halle la solución óptima.

Cuando cambiamos de punto extremo, el teorema de mejora nos da explícitamnte la nueva base asociada al nuevo punto. Este nuevo punto tendrá otras componentes asociadas a la base e inducirá nuevos vectores de costes reducidos y vectores axiliares que necesitan ser calculados para ejecutar la nueva iteración.

La forma inocente de calcular todas estas cosas pasa por invertir la base asociada al nuevo punto extremo, lo cual es pecado mortal, pues es un trabajo computacionalmente muy pesado (del orden de $\mathcal{O}(n^3)$).

En lugar de eso lo que haremos será calcular todos los nuevos valores necesarios a partir de los anteriores. A ver que esto es posible y cómo nos dedicamos en esta sección.

2.1.1. Vectores auxiliares y componentes básicas

En primer lugar nos planteamos la ecuación (1.6) respecto de la base asociada al punto extremo original, B, y la nueva base $B' = (B \setminus a_l) \cup a_k$.

$$x_B = \overline{x_B} - B^{-1}Nx_N \tag{2.1}$$

$$x_{B'} = \overline{x_{B'}} - B'^{-1}N'x_{N'} \tag{2.2}$$

Observación 2.1.1 (Vectores auxiliares)**.** Nótese que, por las propiedades del producto de matrices se tiene que

$$B^{-1}N = B^{-1}(a_{m+1}\cdots a_n) = (B^{-1}a_{m+1}\cdots B^{-1}a_n) = (y_{m+1}\cdots y_n)$$
$$B'^{-1}N' = (y'_{m+1}\cdots y'_{k-1}|y'_l|y'_{k+1}\cdots y'_n) \qquad \diamondsuit$$

Planteando las ecuaciones (2.1) y (2.2) componente a componente nos econtramos con

$$x_s = \overline{x_s} - \sum_{\substack{j=m+1\\ j\neq k}}^{n} y_{sj}x_j - y_{sk}x_k \tag{2.3}$$

$$x_s = \overline{x'_s} - \sum_{\substack{j=m+1\\ j\neq k}}^{n} y'_{sj}x_j - y'_{sl}x_l \tag{2.4}$$

Una vez organizado nuestro espacio de trabajo como lo está ahora, vayamos por partes. Por un lado consideramos la ecuación (2.3) para $s = l$. Despejando x_k de esta ecuación obtenemos (¡compruébese!)

$$x_k = \frac{\overline{x_l}}{y_{lk}} - \sum_{\substack{j=m+1\\ j\neq k}}^{n} \frac{y_{lj}}{y_{lk}}x_j - \frac{1}{y_{lk}}x_l \tag{2.5}$$

Nótese que el despeje que hemos hecho es válido, ya que por el teorema 1.3.6 tenemos garantizado que $y_{lk} > 0$. Ahora, si sustituimos el valor de x_k dado por la ecuación (2.5) en la ecuación (2.3) obtenemos

$$x_s = \overline{x_s} - \frac{y_{sk}\overline{x_l}}{y_{lk}} + \sum_{\substack{j=m+1\\ j\neq k}}^{n} \left(\frac{y_{sk}y_{lj}}{y_{lk}} - y_{sj}\right)x_j + \frac{y_{sk}}{y_{lk}}x_l$$

esta ecuación y la ecuación (2.4) son dos ecuaciones lineales equivalentes. Por tanto, son la una múltiplo de la otra, no obtante, al tener x_s el mismo coeficiente en ambas ecuaciones, estas deben ser iguales. De esto se desprende, comparando ambas ecuaciones

$$\overline{x'_s} = \overline{x_s} - \frac{y_{sk}\overline{x_l}}{y_{lk}} \qquad y'_{sj} = y_{sj} - \frac{y_{sk}y_{lj}}{y_{lk}} \qquad y'_{sl} = -\frac{y_{sk}}{y_{kl}} \tag{2.6}$$

para $s \in \{1,\ldots,m\}\setminus\{l\}$ y con $j \in \{m+1,\ldots,n\}\setminus\{k\}$. En el caso $s = l$ estas expresiones también son válidas (aunque no nos importa mucho).

Nos queda pues el trabajo de hallar extresiones para $\overline{x'_k}$, la coordenada k–ésima del nuevo punto extremo, e y'_{kj} con $j \in \{m+1,\ldots n\}\setminus\{k\}$ (las columnas de N', excepto la columna l, que ya se quedó calculada) ya que y'_j donde j es un índice correspondiente a las columnas de B' es e_j (véase definición 1.3.2).

Cosideramos ahora la ecuación (2.4) en el caso $s = k$. Dicha ecuación y (2.5) son ecuaciones lineales equivalentes, de hecho iguales (siguiendo el razonamiento anterior). Por ende basta compararlas para obtener

$$\overline{x'_k} = \frac{\overline{x_l}}{y_{lk}} \quad y'_{kj} = \frac{y_{lj}}{y_{lk}} \quad y'_{kl} = \frac{1}{y_{lk}} \tag{2.7}$$

2.1.2. Costes reducidos y función objetivo

Echando mano de la ecuación (1.7) respecto de las bases B y B' obtenemos

$$c^t x = c_B^t\overline{x_B} + \sum_{\substack{j=m+1\\ j\neq k}}^{n} \overline{c_j}x_j + \overline{c_k}x_k \tag{2.8}$$

$$c^t x = c_{B'}^t\overline{x_{B'}} + \sum_{\substack{j=m+1\\ j\neq k}}^{n} \overline{c'_j}x_j + \overline{c'_l}x_l \tag{2.9}$$

Sustituyendo el valor de x_k de (2.5) en (2.8) obtenemos

$$c^t x = c_B^t \overline{x_B} + \overline{c_k}\frac{\overline{x_l}}{y_{lk}} + \sum_{\substack{j=m+1 \\ j\neq k}}^{n} \left(\overline{c_j} - \overline{c_k}\frac{y_{lj}}{y_{lk}}\right) x_j - \overline{c_k}\frac{1}{y_{lk}} x_l$$

tanto esta ecuación como la (2.9) son expresiones analíticas de la misma función lineal respecto de las mismas bases (que no tienen nada que ver con las bases asociadas a la matriz del problema). Esto quiere decir que ambas expresiones son iguales, luego comparándolas obtenemos

$$c_{B'}^t \overline{x_{B'}} = c_B^t \overline{x_B} + \overline{c_k}\frac{\overline{x_l}}{y_{lk}} \quad \overline{c'_j} = \overline{c_j} - \overline{c_k}\frac{y_{lj}}{y_{lk}} \quad \overline{c'_l} = -\overline{c_k}\frac{1}{y_{lk}} \tag{2.10}$$

De esta forma, las ecuaciones (2.10), (2.6) y (2.7) nos proporcionan las fórmulas que andábamos buscando.

2.2. Tabla del Símplex. Pivotajes

En esta sección introducimos la llamada ***tabla del símplex***, que no es más que una forma elegante de implementar las fórmulas de cambio de base deducidas en la sección anterior. Además, evita su memorización.

Sea un problema de programación lineal en forma estándar del que suponemos conocido un punto extremo de su región factible, es decir, una base B.

Consideremos el sistema de ecuaciones lineales asociado a dicho problema y multipliquémoslo a ambos lados por B^{-1}.

$$\begin{aligned} Ax = b &\iff (B|N)(x_B^t|x_N^t)^t = b \iff B^{-1}(B|N)(x_B^t|x_N^t)^t = B^{-1}b \iff \\ &\iff (B^{-1}B|B^{-1}N)(x_B^t|x_N^t)^t = \overline{x_B} \iff (I_m|y_{m+1}\cdots y_n)(x_B^t|x_N^t)^t = \overline{x_B} \end{aligned} \tag{2.11}$$

La igualdad final de la ecuación (2.11) puede representarse como una tabla de la siguiente manera

B	x_1	$\cdots$	x_l	$\cdots$	x_m	x_{m+1}	$\cdots$	x_k	$\cdots$	x_n	$\overline{x_B}$
x_1	1	$\cdots$	0	$\cdots$	0	$y_{1,m+1}$	$\cdots$	y_{1k}	$\cdots$	y_{1n}	$\overline{x_1}$
$\cdots$	$\cdots$		$\cdots$		$\cdots$	$\cdots$		$\cdots$		$\cdots$	$\cdots$
x_l	0	$\cdots$	1	$\cdots$	0	$y_{l,m+1}$	$\cdots$	y_{lk}	$\cdots$	y_{ln}	$\overline{x_l}$
$\cdots$	$\cdots$		$\cdots$		$\cdots$	$\cdots$		$\cdots$		$\cdots$	$\cdots$
x_m	0	$\cdots$	0	$\cdots$	1	$y_{m,m+1}$	$\cdots$	y_{mk}	$\cdots$	y_{mn}	$\overline{x_m}$

Tabla 2.1: Primera aproximación a la tabla del Símplex.

La virtud de representar un problema de programación lineal de esta forma es que los cambios de base son "automáticos" en cierto sentido. Supongamos que queremos cambiar de punto extremo de forma que la nueva base es $B' = (B \setminus a_l) \cup a_k$. Siguiendo las fórmulas de la sección anterior, la tabla asociada a la nueva base sería

B'	x_1	$\vdots$	x_l	$\vdots$	x_m	x_{m+1}	$\vdots$	x_k	$\vdots$	x_n	$\overline{x_{B'}}$
x_1	1	$\vdots$	$-\frac{y_{1k}}{y_{lk}}$	$\vdots$	0	$y_{1,m+1} - y_{1k}\frac{y_{l,m+1}}{y_{lk}}$	$\vdots$	0	$\vdots$	$y_{1n} - y_{1k}\frac{y_{ln}}{y_{lk}}$	$\overline{x_1} - y_{1k}\frac{\overline{x_l}}{y_{lk}}$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
x_k	0	$\vdots$	$\frac{1}{y_{lk}}$	$\vdots$	0	$\frac{y_{l,m+1}}{y_{lk}}$	$\vdots$	1	$\vdots$	$\frac{y_{ln}}{y_{lk}}$	$\frac{\overline{x_l}}{y_{lk}}$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
x_m	0	$\vdots$	$-\frac{y_{mk}}{y_{lk}}$	$\vdots$	1	$y_{m,m+1} - y_{mk}\frac{y_{l,m+1}}{y_{lk}}$	$\vdots$	0	$\vdots$	$y_{mn} - y_{mk}\frac{y_{ln}}{y_{kl}}$	$\overline{x_m} - y_{mk}\frac{\overline{x_l}}{y_{lk}}$

Tabla 2.2: Tabla del problema respecto de la base B'.

Si nos fijamos, la nueva tabla puede verse como el resultado de aplicar el algoritmo de Gauss–Jordan en la columna k–ésima, es decir, considerando la tabla como una matriz, usar las transformaciones elementales de matrices para convertir el elemento y_{lk} en un 1 y "hacer ceros" en el resto de la columna. Esto es maravilloso porque puede implementarse en ordenador de forma casi trivial.

La mala noticia es que la tabla 2.2 no tiene todos los datos necesarios para que el algoritmo se ejecute. En concreto, faltan los costes reducidos y el valor de la función objetivo.

La gran noticia sin embargo (y prueba irrefutable de que los dioses son benévolos de cuando en cuando) es que podemos completar la tabla 2.2 de manera que lleve todos los datos necesarios y además la implementacón de los cambios de base no varíe en absoluto. En efecto, si consideramos la tabla

B	x_1	$\cdots$	x_l	$\cdots$	x_m	x_{m+1}	$\cdots$	x_k	$\cdots$	x_n	$\overline{x_B}$
x_1	1	$\cdots$	0	$\cdots$	0	$y_{1,m+1}$	$\cdots$	y_{1k}	$\cdots$	y_{1n}	$\overline{x_1}$
$\cdots$	$\cdots$		$\cdots$		$\cdots$	$\cdots$		$\cdots$		$\cdots$	$\cdots$
x_l	0	$\cdots$	1	$\cdots$	0	$y_{l,m+1}$	$\cdots$	y_{lk}	$\cdots$	y_{ln}	$\overline{x_l}$
$\cdots$	$\cdots$		$\cdots$		$\cdots$	$\cdots$		$\cdots$		$\cdots$	$\cdots$
x_m	0	$\cdots$	0	$\cdots$	1	$y_{m,m+1}$	$\cdots$	y_{mk}	$\cdots$	y_{mn}	$\overline{x_m}$
$\overline{c}$	0	$\cdots$	0	$\cdots$	0	$\overline{c_{m+1}}$	$\cdots$	$\overline{c_k}$	$\cdots$	$\overline{c_n}$	$-(c_B^t\overline{x_B})$

Tabla 2.3: Tabla del Símplex.

la tabla asociada a la base B' será

B'	x_1	$\vdots$	x_l	$\vdots$	x_m	x_{m+1}	$\vdots$	x_k	$\vdots$	x_n	$\overline{x_{B'}}$
x_1	1	$\vdots$	$-\frac{y_{1k}}{y_{lk}}$	$\vdots$	0	$y_{1,m+1}-y_{1k}\frac{y_{l,m+1}}{y_{lk}}$	$\vdots$	0	$\vdots$	$y_{1n}-y_{1k}\frac{y_{ln}}{y_{lk}}$	$\overline{x_1}-y_{1k}\frac{\overline{x_l}}{y_{lk}}$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
x_k	0	$\vdots$	$\frac{1}{y_{lk}}$	$\vdots$	0	$\frac{y_{l,m+1}}{y_{lk}}$	$\vdots$	1	$\vdots$	$\frac{y_{ln}}{y_{lk}}$	$\frac{\overline{x_l}}{y_{lk}}$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
x_m	0	$\vdots$	$-\frac{y_{mk}}{y_{lk}}$	$\vdots$	1	$y_{m,m+1}-y_{mk}\frac{y_{l,m+1}}{y_{lk}}$	$\vdots$	0	$\vdots$	$y_{mn}-y_{mk}\frac{y_{ln}}{y_{kl}}$	$\overline{x_m}-y_{mk}\frac{\overline{x_l}}{y_{lk}}$
$\overline{c'}$	0	$\vdots$	$-\overline{c_k}\frac{1}{y_{lk}}$	$\vdots$	0	$\overline{c_{m+1}}-\overline{c_k}\frac{y_{l,m+1}}{y_{lk}}$	$\vdots$	0	$\vdots$	$\overline{c_n}-\overline{c_k}\frac{y_{ln}}{y_{lk}}$	$-c_{B'}^t\overline{x_{B'}}-\overline{c_k}\frac{\overline{x_l}}{y_{lk}}$

Tabla 2.4: Tabla del problema respecto de la base B'.

Como se observa, la tabla del Símplex tiene en su última fila el vector de costes reducidos y el **opuesto** al valor que toma la función objetivo en el punto extremo asociado a la base que corresponda. La razón para guardar el opuesto y no el valor a secas es que si guardamos el opuesto podemos seguir haciendo los cambios de base como si aplicaramos parcialmente el algoritmo de Gauss-Jordan, en caso contrario no podríamos hacerlo.

Observación 2.2.1 (Notación)**.** Es habitual encontrarse con que la casilla correspondiente a la función objetivo en la tabla del Símplex contiene la expresión $z-(c_B^t\overline{x_B})$. Esto no supone ninguna variación en absolutamente nada, es símplemente notación, añadir una z al principio, sin más. ♢

A la operación de cambiar de base una tabla se la denmina ***pivotaje*** y al elemento y_{lk} se le denomina ***pivote***.

2.2.1. Criterios de entrada y salida de la base

El algoritmo 2 deja al azar qué hacer para elegir la columna que entra en la base cuando hay varias columnas con coste reducido negativo. Hay diversas heurísticas y técnicas para resolver estos empates a la hora de la implementación, no obstante, la que usaremos nosotros es la llamada ***regla de Dantzig***, que consiste en coger la columna asociada al coste reducido más pequeño, deshaciendo los empates al azar.

En cualquier caso, como hemos podido demostrar, aplicar esta regla o no da un poco igual, por tanto se dejó al azar con cierta parte de razón. No obstante, a la hora de la implementación si es muy recomendable seguir algún criterio, pues elegir la columna de entrada al tuntún puede llevarnos a hacer más pivotajes de la cuenta.

El algoritmo 2 sí que nos da un criterio de salida de la base, ya que nos pide amablemente que apliquemos el teorema 1.3.6, y este teorema nos dice que escojamos la columna que minimice el cociente $\frac{\overline{x_i}}{y_{ik}}$ con $y_{ik} > 0$ siendo k el índice de la columna que va a entrar en la base.

A este criterio se le conoce como ***regla de la razón mínima***. El problema de esta regla es que en ocasiones pueden producirse empates, que, aunque en la mayoría de los casos podrían resolverse al azar, en ciertas circunstancias pueden ocasionar graves problemas. Veremos como subsanar estas contingencias más adelante.

2.3. Punto extremo inicial

Aunque el trabajo duro ya está hecho, todavía quedan preguntas por resolver, por ejemplo, cómo obtener el punto extremo inicial (si lo hay, ya que el problema podría ser infactible) y todos los datos necesarios asociados a este de forma eficiente.

2.3.1. Método de las dos fases

Encontrar bases de matrices es un trabajo compuracionalmente costoso, en el caso peor, que no haya base alguna, es del orden de $\mathcal{O}\binom{n}{m}$ (eso si las comprobaciones intermedias tuvieran coste constante, que no lo tienen). Razón por la cual necesitamos algún tipo de sustitutivo a este método de fuerza bruta.

Lo que exponemos a continuación es conocido como ***método de las dos fases*** y únicamente es aplicable a problemas en forma estándar con $b \geq 0$. Esto no supone ninguna pérdida de generalidad ya que siempre podemos multiplicar la restricción correspondiente a un $b_i < 0$ por -1, obteniendo una restricción equivalente y, por tanto, un problema equivalente.

A un problema en las hipótesis anteriormente expuestas que tenga a A por matriz de coeficientes de las restricciones, se le asocia otro problema al que llamaremos ***problema artificial***. Dicho problema es el siguiente

$$\begin{array}{ll} & \min w(x^t|x_a^t)^t := \sum_{i=1}^m x_i^a \\ \text{Sujeto a:} & (A|I_m)(x^t|x_a^t)^t = b, \qquad x \geq 0,\ x_a \geq 0 \end{array}$$

es decir, el problema resultante de cosiderar las restricciones

$$a_{i1}x_1 + \cdots + a_{in}x_n + x_i^a = b_i$$

para $i \in \{1, \ldots, m\}$, cuya matriz asociada es $(A|I_m)$. A las nuevas variables x_i^a se las conoce como ***variables artificiales***. Además, la función objetivo se sustituye por $w \equiv \sum_{i=1}^m x_i^a$.

Entre las buenas propiedades de este nuevo problema es que tiene una base trivial, la identidad, situada convenientemente en las últimas columnas (lo cual el ordenador agradece).

Pero la cosa no queda aquí, ya que el problema artificial no es un problema no acotado. En efecto, por las restricciones de no negatividad, la función objetivo está acotada inferiormente por 0. Además, el problema artificial es factible, bastanto considerar el vector $(0|b^t)^t \in \mathbb{R}^{n+m}$ como solución factible. Esta es la razón por la que exigimos $b \geq 0$.

Con esto, por el teorema fundamental de la programación lineal (teorema 1.3.3), sabemos que el problema artificial tiene solución óptima. Por si esto fuera poco, obtenemos casi gratis un test de factibilidad para el problema original.

Lema 2.3.1 (Test de factibilidad)**.** *El problema original es factible si y solo si el problema artificial posee una solución óptima* $(\overline{x}^t|\overline{x_a}^t)^t$ *con* $\overline{x_a} = 0$.

Demostración. Demostraremos la implicación a la derecha, para descubrir después que si miramos la demostración con amor, y del revés, la otra implicación también está demostrada.

Si el problema original es factible, entonces habrá un $x \geq 0$ tal que $Ax = b$. En tal caso es claro que el vector $(x^t|0)^t$ es solución factible del problema artificial (¡compruébese!). Como resulta que

$w(x^t|0)^t = 0$, por la acotación inferior de w tenemos que $(x^t|0)^t$ es solución óptima del problema artificial con $\overline{x_a} = 0$. ■

Observación 2.3.1 (Sutileza)**.** Si el problema artificial posee una solución óptima x^* con $\overline{x_a} = 0$, todas sus soluciones óptimas x' tendrán $\overline{x_a} = 0$, ya que, en caso contrario $w(x^*) < w(x')$, lo cual es absurdo. ♢

Dicho esto, es claro que, dado un problema de programación lineal dentro de nuestras hipótesis, podemos considerar el problema artificial asociado y resolverlo mediante el algoritmo del Símplex que ya tenemos desarrollado, siendo el rellenado de la tabla inicial (y la determinación del primer punto extremo) trivial, ya que $B = I_m$, por tanto la tabla del Símplex es la correspondiente a la matriz $(A|I_m|b)$ añadiendo como última fila el vector de costes reducidos y el opuesto del valor de la función objetivo.

Cabe destacar que el cálculo del vector de costes reducidos es bastante trivial computacionalmente ya que no hay que invertir ninguna base, símplemente $\overline{w_j} = w_j - w_B^t a_j$.

Una vez hallada la solución óptima al problema artificial, podremos usar el lema 2.3.1 para decidir sobre la factibilidad o no del problema original. En caso de que este resulte factible, la base asociada a la solución óptima encontrada será base del problema original (ya que todas las columnas de la base están asociadas a columnas de la matriz original). Con esto, ya podríamos aplicar el algoritmo del Símplex para resolver el problema original.

No obstante, cabría preguntarse si es computacionalmente costoso rellenar la tabla del simplex del problema original respecto de la base B dada (es decir, ¿hay que invertir B?). La respuesta es no, ya que necesitamos calcular la matriz

$$B^{-1}(A|b) = (B^{-1}a_1 \cdots B^{-1}a_n|B^{-1}b)$$

y la matriz asociada a la tabla asociada a la solución óptima del problema artificial es

$$B^{-1}(A|I_m|b) = (B^{-1}a_1 \cdots B^{-1}a_n|B^{-1}e_1 \cdots B^{-1}e_m|B^{-1}b) = (B^{-1}A|B^{-1}|B^{-1}b)$$

luego basta con tomar la misma tabla suprimiendo las columnas asociadas a las variables artificiales.

En cuanto al cálculo de los costes reducidos, aunque ya tenemos invertida la matriz B y podríamos hacerlo "a capón", no es recomendable, pues multiplicar matrices es computacionalemente costoso.

En lugar de eso, lo que podemos hacer es añadir una fila más a la tabla del Símplex del problema artificial, donde ir calculando (mediante los pivotajes) los costes reducidos asociados a la función objetivo $z := c^t x$ del problema original (nótese que no hay ningún problema).

De esta forma, basta con consultar la tabla asociada a B del problema artificial para rellenar la tabla inicial del problema original, que ya puede ser resuelto usando el algoritmo del Símplex conocido. Organizemos toda esta literatura.

Algoritmo 3 Algoritmo de las dos fases.

```
{Primera fase}
Construir el problema artificial
Resolver el problema artificial (algoritmo 2) repedidas veces.
Aplicar el test de factibilidad (lema 2.3.1)
si es factible entonces
    {Segunda fase}
    Rellenar la primera tabla con los trucos vistos.
    Resolver el problema original (algoritmo 2) repedidas veces.
si no
    devolver Infactible
fin si
```

2.3.2. Método de las penalizaciones (Big M)

Una alternativa al método de las dos fases es el llamado ***método de las penalizaciones*** o de la M grande. Al igual que el método de las dos fases, solo es aplicable a problemas con $b \geq 0$, lo cual, como discutimos anteriormente, no supone ninguna pérdida de generalidad.

El método de las dos fases también asocia al problema que queremos resolver otro problema, al que llamaremos ***problema penalizado***, cuyas restricciones son exactamente las mismas que las del problema artificial. Lo único que los diferencia es la función objetivo, que en este caso es

$$u :\equiv z + Mw$$

donde z es la función objetivo del problema original y w la función objetivo del problema artificial de las dos fases. Por su parte, M es una constante positiva lo suficientemente grande (especificaremos después).

Antes de continuar presentemos un par de observaciones elementales.

Observación 2.3.2 (Relaciones de factibilidad)**.** Evidentemente, toda solución factible del problema original puede "extenderse" (poniendo las variables artificiales a cero) de modo que se convierte en una solución factible del problema penalizado. Este proceso puede realizarse a la inversa, en el sentido de que toda solución factible del problema penalizado que tenga a 0 todas sus variables artificiales puede "acortarse" (fusilando las componentes asociadas a las variables artificiales) de manera que pasa a ser una solución factible del problema original. ♢

Observación 2.3.3 (Sutilezas)**.** Evidentemente, el problema penalizado es factible, esto ocurre por el mismo motivo que lo es el problema artificial. No obstante, y al contrario de lo que sucedía con el problema artificial, el problema penalizado puede estar no acotado. En efecto, basta que el problema original sea no acotado. ♢

Dicho esto, si aplicamos el algoritmo del Símplex al problema penalizado, este puede acabar de dos maneras

1. El último punto extremo analizado tiene alguna componente artificial positiva.
2. El último punto extremo tiene todas las componentes asociadas a variables artificiales nulas.

Demostremos que si se da el primer caso, entonces el problema original es infactible.

Proposición 2.3.2 (Test de infactibilidad)**.** *Si el último punto extremo tiene alguna componente positiva, el problema original es infactible.*

Demostración. Distinguimos dos casos. Según el algoritmo detecte la no acotación o la optimalidad del problema penalizado.

- En caso de detectar la optimalidad (digamos para un punto $((x^*)^t|x_a^t)^t$), es claro que el problema es infactible, ya que de existir una solución factible $\overline{x}$ del problema original (sin pérdida de generalidad un punto extremo) se tendría que $c^t\overline{x} \leq c^t x^* + Mw((x^*)^t|x_a)^t$, siempre y cuando M sea adecuadamente grande (especificaremos luego).
- En caso de detectarse la no acotación tendríamos que hay un cierto k para el cual $\overline{u_k} < 0$ y además $y_k \leq 0$ (si hubiera varios escogemos es que tenga coste reducido menor). Si desarrollamos la cuenta tenemos que

  $$\overline{u_j} = u_j - \sum_{\text{no artif.} \in B} c_i y_{ij} - M \sum_{\text{artif.} \in B} y_{sj}$$

  En el caso de $\overline{u_k}$ todos los escalares y_{sk} correspondientes al último sumatorio son nulos, esto se debe a que, de haber alguno negativo tendríamos que $\overline{u_k} > 0$ suponiendo que M sea convenientemente grande.

  En el caso de otros $\overline{u_j}$ el último sumatorio no puede ser positivo, ya que, de serlo, al ser M lo suficientemente grande tendríamos que $\overline{u_j} < \overline{u_k}$, contra la minimalidad de $\overline{u_k}$.

  La ecuación (2.1) particularizada para las componentes asociadas a la base y que además están asociadas a variables artificiales nos dice que

  $$x_s = \overline{x_s} - \sum_{j \notin B} y_{sj} x_j$$

sumando todas estas ecuaciones obtenemos

$$\sum_{\text{artif.}\in B} x_s = \sum_{\text{artif.}\in B} \overline{x_s} - \sum_{j\notin B}\left(\sum_{\text{artif.}\in B} y_{sj}\right) x_j$$

si el problema original tuviera alguna solución factible, su ampliación sería solución del problema penalizado, por ende, deberá cumplir la ecuación anterior, que en su caso particular es

$$\sum_{\text{artif.}\in B} \overline{x_s} = \sum_{j\notin B}\left(\sum_{\text{artif.}\in B} y_{sj}\right) x_j$$

por hipótesis el miembro de la izquierda es estrictamente positivo, mientras que el de la derecha es no negativo, lo cual es absurdo. ■

Una vez hecho esto, es trivial demostrar (se deja al lector) que si se da el segundo caso se dan los siguientes subcasos

- Si se detecta optimalidad, la solución "recortada" es solución óptima del problema original.
- Análogamente con la no acotación.

Para casi finalizar, veamos cómo son los costes reducidos asociados a la función objetivo u, para lo cual, echaremos una cuenta rápida, pero antes, un poco de notación. En primer lugar extendemos la definición de c_j por 0 para índices $j \in \{n+1,\dots,n+m\}$. En segundo lugar, definimos d_j como 0 para $j \in \{1,\dots,n\}$ y como 1 para $j \in \{n+1,\dots,n+m\}$. De esta forma

$$u^t x = \sum_{j=1}^{n+m} u_j x_j = \sum_{j=1}^{n+m} c_j x_j + M \sum_{j=1}^{n+m} d_j x_j = \sum_{j=1}^{n+m} (c_j + d_j M) x_j$$

con lo que finalmente se concluye

$$\overline{u_j} = u_j - u_B^t y_j = (c_j + M d_j) - (c_B^t + M d_B^t) y_j = \\ = (c_j - c_B^t y_j) + M(d_j - d_B^t y_j) = \overline{c_j} + M\overline{d_j}$$

Dicho esto, las tablas del Símplex se representarán de la manera habitual con una pequeña salvedad, dividiremos (por razones de comodidad) tanto la función objetivo como el vector de costes reducidos en dos filas, una para la parte asociada a z y otra para la parte asociada a Mw (nótese que los pivotajes se pueden seguir haciendo exactamente de la misma forma).

Finalizamos este apartado con dos pequeñas observaciones.

Observación 2.3.4 (Implementación)**.** Nótese que a efectos de implementación no necesitamos saber cuánto vale M (mucho menos si somos laxos a la hora de aplicar la regla de Dantzig), lo único que tenemos que tener en cuenta es que es "dominante", en el sentido de que al saber si un coste reducido es positivo o negativo basta mirar el factor que acompaña a M, ya que lo demás lo consideraremos despreciable. ♢

Observación 2.3.5 (Valor de M)**.** Aunque para la implementación no necesitamos saber el valor de M, si que necesitamos saber que todo problema posee un valor finito de M tal que hace ciertas las demostraciones que hemos venido realizando.

Demostrar esto es sencillo, basta ir releyendo el apartado, a lo largo del cual vamos imponiendo exigencias a la M, a partir de estas exigencias podemos ir sacando cotas inferiores (se deja el trabajo al lector). ♢

2.4. Prevención de bucles

El algoritmo 2 se basa fundamentalmente en el teorema 1.3.6, el cual deja la puerta abierta a que se produzcan los temido ***pivotajes degenerados***, es decir, cambios de base que no mejoran el valor de la función objetivo.

El propio teorema 1.3.6 nos da una forma de saber cuándo un pivotaje es degenerado, pues basta fijarnos en la tabla y ver que estamos intentando sacar la columna l–ésima de la base, teniendo esta el valor $\overline{x_l} = 0$.

Los pivotajes degenerados no cambian el punto extremo, analizan el mismo punto extremo de nuevo pero respecto de otra base distinta. Esto también nos lo dice el teorema 1.3.6, ya que el nuevo punto extremo a analizar viene dado explícitamente por $\overline{x} + \frac{\overline{x_l}}{y_{lk}} d$, luego si $\overline{x_l} = 0$ el punto no cambia.

De esta forma se deduce que si el algoritmo no se encuentra con pivotajes degenerados, necesariamente termina ya que no puede examinar una base anteriormente examinada, ya que el valor del siguiente punto extremo a examinar es estrictamente menor que el anterior.

Por ende, el peligro de los bucles únicamente podría aparecer con los pivotajes degenerados, y de hecho aparece, el ejemplo más famoso se debe a Martin Beale. En esta sección desarrollaremos técnicas para prevenir estos bucles.

2.4.1. Regla lexicográfica

En este apartado exponemos un criterio de salida de la base, al que llamamos ***regla de la razón léxico–mínima*** o símplemente ***regla lexicográfica***. Lo que hace útil a este criterio (descubierto en 1955) es que, como veremos al finalizar la sección, es un criterio que evita la aparición de bucles, de forma que garantiza la convergencia del algoritmo a una solución óptima (cuando el problema la posea).

Antes de comenzar introduzcamos las siguientes definiciones.

Definición 2.4.1 (Lexicográficamente positivo)**.** Un vector $x \in \mathbb{R}^n$ se dice ***lexicográficamente positivo*** si es no nulo y su primera componente no nula es positiva.

Esto lo denotamos por $x >_L 0$.

Definición 2.4.2 (Orden lexicográfico)**.** Dados dos vectores $x, y \in \mathbb{R}^n$ decimos que x es ***lexicográficamente mayor*** que y si $x - y$ es lexicográficamente positivo. Sintéticamente

$$x >_L y \iff x - y >_L 0$$

Esta relación define un orden total estricto en $\mathbb{R}^n \setminus \{0\}$.

Hechas estas consideraciones previas, supongamos que en cierta iteración del algoritmo se decide que la k–ésima columna entra en la base, mientras que a la hora de decidir la variable que sale, se producen empates con la regla de la razón mínima.

En tal caso consideramos la matriz $V_B := (\overline{x_B} | B^{-1})$ donde B es la base "vieja" que vamos a sustituir por la base B' (aún por calcular).

Denotaremos por V_B^i a la fila i–ésima de la matriz V_B.

Observación 2.4.1 (Cálculo de V_B)**.** Nótese que el cálculo de la matriz V_B no supone ningún coste adicional para el ordenador en el caso de que la base inicial sea la identidad, cosa que siempre pasará si utilizamos el método de las dos fases o de las penalizaciones.

Esto se debe a que la tabla del símplex (salvo el vector de costes y la función objetivo) viene dada por la matriz

$$B^{-1}(A|b) = B^{-1}(N|I_m|b) = (B^{-1}N|B^{-1}|\overline{x_B})$$

con lo que tenemos B^{-1} en las columnas correspondientes a la base inicial, con lo que automáticamente tenemos V_B.

Este es un buen motivo para, cuando nos encontremos en la segunda fase del método de las dos fases, no tachar las columnas sobrantes, sino tenerlas ahí (marcadas para no pivotar por ellas) con objeto de tener B^{-1} siempre a mano por si se producen ciclos. ♢

Dicho esto, la **regla lexicográfica** dicta que saldrá de la base la columna l–ésima si y solo si se verifica

$$y_{lk} > 0 \qquad \frac{V_B^i}{y_{ik}} >_L \frac{V_B^l}{y_{lk}} \tag{2.12}$$

para todo $i \in \{1, \ldots, m\} \setminus \{l\}$ con $y_{ik} > 0$.

Antes de continuar presentemos dos observaciones interesantes.

Observación 2.4.2 (Generalización)**.** Nótese que la regla lexicográfica no es más que una generalización de la regla de la razón mínima, en la que en lugar de hacer una comparación lexicográfica de los vectores $V_B^i \frac{1}{y_{ik}}$ y $V_B^l \frac{1}{y_{lk}}$, únicamente se comparaban sus primeras componentes.

Esto sugiere una implementación natural de la comparación lexicográfica, que únicamente necesita comparar $V_B^i \frac{1}{y_{ik}}$ y $V_B^l \frac{1}{y_{lk}}$ componente a componente hasta que la comparación arroje un resultado no nulo. ♢

Observación 2.4.3 (Imposibilidad de empates)**.** Es interesante observar también que la regla lexicográfica no permite los empates, ya que, en caso de que varios vectores empataran se dería la situación

$$\frac{V_B^i}{y_{ik}} >_L \frac{V_B^l}{y_{lk}} \iff \frac{y_{lk}}{y_{ik}} V_B^i = V_B^l$$

Luego B^{-1} tendría dos filas proporcionales (lo cual va contra su invertivilidad). ♢

Para finalizar, comprobemos que, efectivamente, la regla lexicográfica es una regla de prevención de ciclos, para lo cual necesitamos introducir una definición.

Definición 2.4.3 (Valor vectorial lexicográfico)**.** Definimos el ***valor vectorial lexicográfico*** asociado a la base B como el vector $v_B := \sum_{i=1}^m c_i V_B^i$.

La idea de la demostración será comparar los valores lexicográficos de una base B y el de la base B' resultante de aplicar el cambio de base según la regla lexicográfica, viendo que $v_{B'} <_L v_B$, siendo imposible que el algoritmo analice dos veces una misma base.

Antes de pasar a demostrar esto, observemos la relación que hay entre $V_{B'}^i$ y V_B^i. Símplemente fijándonos en las fórmulas del pivotaje tenemos que

$$V_{B'}^k = \frac{V_B^l}{y_{lk}} \qquad V_{B'}^i = V_B^i - \frac{y_{ik}}{y_{lk}} V_B^l \tag{2.13}$$

Observación 2.4.4 (Positividad)**.** Nótese además que si la primera base considerada en el algoritmo es la identidad, es claro que $V_{I_m}^i >_L 0$, pero no solo eso, la aplicación de la regla lexicográfica en este supuesto asegura que $V_B^i >_L 0$ para toda base considerada.

Para probar esto basta echar un vistazo a las ecuaciones (2.12) y (2.13) (se deja al lector). ♢

Dicho esto, ya estamos listos para demostrar lo que queremos.

Proposición 2.4.1 (Buena definición)**.** *La regla lexicográfica es una regla de prevención de ciclos.*

Demostración. Basta desarrollar la siguiente cuenta

$$v_{B'} := \sum_{\substack{i=1 \\ i\neq l}}^m c_i V_{B'}^i + c_k V_{B'}^k = \sum_{\substack{i=1 \\ i\neq l}}^m c_i \left(V_B^i - \frac{y_{ik}}{y_{lk}} V_B^l \right) + c_k \frac{V_B^l}{y_{lk}} =$$

$$= \sum_{\substack{i=1 \\ i\neq l}}^m c_i V_B^i + \frac{V_B^l}{y_{lk}} \left(c_k - \sum_{\substack{i=1 \\ i\neq l}}^m c_i y_{ik} \right) = \sum_{\substack{i=1 \\ i\neq l}}^m c_i V_B^i + \overline{c_k} \frac{V_B^l}{y_{lk}} <_L \sum_{i=1}^m c_i V_B^i =: v_B \quad \blacksquare$$

2.4.2. Regla de Bland

Presentamos a continuación otra regla de prevención de ciclos, debida a Robert G. Bland (quien la publicó en 1977), y que, como es evidente, es conocida como ***regla de Bland***. Esta regla es muy sencilla y constituye un criterio de entrada y salida a la base.

Lo que dice la regla es lo siguiente: de todas las columnas candidatas a entrar en la base escogemos la de menor índice. Análogamente, de todas las columnas candidatas a salir de la base (que empatan con la regla de la razón mínima), escogemos también la de menor índice.

Probemos pues que la regla de Bland es una regla de prevención de ciclos.

Proposición 2.4.2 (Buena definición)**.** *El algoritmo del Símplex implementado con la regla de Bland siempre termina.*

Demostración. Supongamos que se produce un ciclo. Es decir, el algoritmo examina las bases $B_0, \ldots, B_k$ y tras esto se vuelve a exmaminar la base B_0.

Dicho esto llamaremos ***volátiles*** a las columnas que pertenecen a alguna base del ciclo, pero no a todas. Consideremos la columna volátil de mayor índice, a la que llamaremos a_t.

Por ser a_t volátil, habrá alguna base B para la cual a_t sea seleccionada para entrar en la base. Es claro que se debe cumplir que $\overline{c_t} < 0$, ya que si no, no sería candidata a entrar en la base. Además, por la implementación de la regla de Bland, las demás columnas no asociadas a la base deben cumplir que $\overline{a_j} \geq 0$.

Asimismo, por la volatilidad de a_t, habrá alguna base B' para la cual a_t sea seleccionada para salir de la base, siendo sustituida por la columna a_s. En estas circunstancias se debe cumplir que $y'_{ts} > 0$, además, $\overline{c'_s} < 0$ y $s < t$ (por la regla de Bland).

El punto extremo asociado a la base posterior a B' es $x' = \overline{x} + \alpha d$ (ver teorema 1.3.6), siendo $d = ((-y_s|0) + e_s)^t$. Claramente d es una dirección extrema, por tanto, se verifica que $Ad = 0$, es decir, d está en el complemento ortogonal del espacio de filas de A.

Sabiendo que $\overline{c} = c - c_B^t B^{-1} A$ y $\overline{c'} = c - c_{B'}^t B'^{-1} A$ tenemos que

$$c^* := \overline{c} - \overline{c'} \overset{!}{=} (c_{B'}^t B'^{-1} - c_B^t B^{-1}) A$$

luego c^* está en el espacio de filas de A, lo que quiere decir que d es ortogonal a c^*. En otras palabras $(c^*)^t d = 0$. No obstante además tenemos que

$$(c^*)^t d = c_s^* d_s + c_t^* d_t + \sum_{\text{no volátiles en } B'} c_j^* d_j + \sum_{\substack{\text{volátiles en } B' \\ j \neq t}} c_j^* d_j$$

El sumatorio asociado a las columnas no volátiles se anula ya que $c_j^* = 0$, los costes reducidos siempre se anulan por estar siempre en la base. Por su lado, el sumatorio asociado a las columnas volátiles es no negativo ya que $d_j \geq 0$ (en caso contrario estaríamos quebrantando la regla de Bland) y $c_j^* = \overline{c_j} \geq 0$ (como vimos antes). Además como $d_s = 1$, $\overline{c_s} \geq 0$ y $-\overline{c'_s} > 0$ tenemos que el primer sumando es positivo. Finalmente, como tanto d_t como c_t^* son negativos se tiene que el segundo sumando es positivo.

En definiva $(c^*)^t d > 0$, lo cual es absurdo, con lo que queda demostrado que los ciclos son imposibles. ■

Observación 2.4.5 (Desventajas)**.** La regla de Bland es maravillosa por su sencillez, no obstante, en la práctica puede llegar a ser peligrosa, pues suele conducir a realizar más pivotajes de los que serían necesarios usando otros criterios, como el lexicográfico. ♢

Capítulo 3

Dualidad

En este capítulo introduciremos los llamados "problemas duales", estudiando sus propiedades y relaciones con sus respectivos "problemas primales". Sacaremos jugo a estas propiedades cual Jíbaro a la cabeza de un enemigo.

3.1. Formulación canónica del problema dual

Sea un problema P de programación lineal en forma canónica de maximización. Sintéticamente

$$\begin{array}{rl} & \max c^t x \\ \text{Sujeto a:} & Ax \leq b, \qquad x \geq 0 \end{array}$$

se define el ***problema dual*** D asociado a P como

$$\begin{array}{rl} & \min b^t u \\ \text{Sujeto a:} & A^t u \geq c, \qquad u \geq 0 \end{array}$$

usualmente nos referiremos al problema P como el ***problema primal***.

Observación 3.1.1 (Involutividad)**.** La primera cosa que salta a la vista es que la dualidad es involutiva, es decir, el problema dual asociado al problema dual es el problema primal (esto lo veremos en el ejemplo 3.1.1), que es de vital importancia. ♢

Aunque esta definición de problema dual parezca estricta, por solo poder aplicarse a los problemas en forma canónica de maximización, en realidad no lo es tanto, ya que podemos pasar de una forma canónica a otra. Veamos, por ejemplo, cuál es el problema dual asociado a un problema en forma canónica de minimización.

Ejemplo 3.1.1 (Minimización)**.** Teniendo en cuenta que $\min f = -\max -f$ (se deja al lector la comprobación) tenemos que, dado el problema en forma canónica de minimización

$$\begin{array}{rl} & \min c^t x \\ \text{Sujeto a:} & Ax \geq b, \qquad x \geq 0 \end{array}$$

su transformación a forma canónica de maximización es

$$\begin{array}{rl} & -\max -c^t x \\ \text{Sujeto a:} & -Ax \leq -b, \qquad x \geq 0 \end{array}$$

donde lo único que se ha hecho es multiplicar por -1 todas las restricciones así como la función objetivo. En esta situación el problema dual ya está bien definido, y es

$$\begin{array}{rl} & -\min -b^t u \\ \text{Sujeto a:} & -A^t u \geq -c, \qquad u \geq 0 \end{array}$$

planteando esto de nuevo en forma canónica de maximización obtenemos

$$\begin{array}{c} \max b^t u \\ \text{Sujeto a:} \quad A^t u \leq c, \qquad u \geq 0 \end{array}$$

con lo que ya tenemos el problema dual asociado al problema original. ♢

Nótese que no está muy claro cómo definir el problema dual asociado a un problema en forma estándar (mucho menos en otras formas más exóticas). A esta cuestión nos decicaremos en secciones posteriores, justo después de ver por qué merece la pena plantearse estos problemas.

3.2. Relaciones de dualidad

En esta sección veremos cómo se relacionan un problema y su dual en términos de sus soluciones, viendo así parte de la utilidad de estudiar la dualidad. Comencemos con el siguiente resultado elemental.

Lema 3.2.1 (Cotas)**.** *Toda solución factible del problema dual proporciona una cota superior del valor óptimo de la función objetivo del problema primal, y viceversa. Es decir*

$$c^t x \leq b^t u$$

Demostración. Sea $x \in \mathbb{R}^n$ una solución factible del problema primal. Asimismo consideremos $u \in \mathbb{R}^m$ una solución factible del problema dual.

Tenemos que $c^t x = \sum_{j=1}^n c_j x_j$. Teniendo en cuenta la definición del problema dual ($A^t u \geq c$) tenemos que $c^t x \leq \sum_{j=1}^n (\sum_{i=1}^m a_{ij} u_i) x_j$. Intercambiando los sumatorios obtenemos la expresión $c^t x \leq \sum_{i=1}^m (\sum_{j=1}^n a_{ij} x_j) u_i$. Teniendo en cuenta la definición del problema primal ($Ax \leq b$) concluimos que $c^t x \leq \sum_{i=1}^m b u_i = b^t u$. Como queríamos demostrar. ■

Corolario 3.2.2 (Dualidad débil)**.** *Si dos soluciones factibles x y u, del problema primal y dual respectivamente, verifican que $c^t x = b^t u$, entonces, tanto x como u son soluciones óptimas de sus respectivos problemas.*

El siguiente resultado es conocido en la literatura con el nombre de "teorema de dualidad" o "teorema de dualidad fuerte". Es el recíproco del corolario 3.2.2. Veámoslo.

Teorema 3.2.3 (Dualidad fuerte)**.** *Si el problema primal tiene solución óptima x, entonces el problema dual tiene solución óptima u, y se verifica que $c^t x = b^t u$.*

Demostración. Podemos suponer si pérdida de generalidad que la solución óptima x se ha obtenido mediante el algoritmo del Símplex. En tal, caso llamaremos B a su base asociada. Asimismo consideremos el vector fila $\lambda^t := c_B^t B^{-1}$.

Como x es solución óptima de un problema de maximización encontrada mediante el algoritmo del Símplex se deberá verificar que todos los costes reducidos son no positivos, es decir, $\overline{c_j} \leq 0$.

Nótese que para que se pueda ejecutar el algoritmo del Símplex sobre el problema primal se deberán añadir m variables de holgura (una por restricción), de modo que la matriz del problema pasaría a ser $(A|I_m)$.

Estudiemos con un poco de cariño los costes reducidos.

$$\overline{c_j} = c_j - c_B^t B^{-1} a_j = c_j - \lambda^t a_j \leq 0 \iff \lambda^t a_j \geq c_j \iff \sum_{i=1}^m a_{ij} \lambda_i \geq c_j$$

La última equivalencia nos dice que el vector $\lambda \in \mathbb{R}^m$ cumple con las restricciones del problema dual. Para asegurarse de que es solución factible de este basta con comprobar la no negatividad, para ello consideramos los costes reducidos asociados a las variables de holgura ($\overline{c_j}$ con $j \in \{n+1, \ldots, n+m\}$), que se caracterizan por el hecho de que $c_{n+i} = 0$ y $a_{n+i} = e_i$ con $i \in \{1, \ldots, m\}$.

$$\overline{c_{n+i}} = -\lambda^t e_i = -\lambda_i \leq 0 \iff \lambda_i \geq 0$$

con lo que se concluye que λ es solución factible dual. Además

$$c^t x = \sum_{i=1}^{n} c_i x_i = c_B^t \overline{x_B} = c_B^t B^{-1} b = \lambda^t b = b^t \lambda$$

por el coralario 3.2.2 tenemos que λ es solución óptima del problema dual. ■

Observación 3.2.1 (Construcción)**.** Nótese que a partir de la solución óptima del primal obtenida por el algoritmo del Símplex obtenemos automáticamente la solución óptima del dual, en concreto, si la solución óptima del primal está asociada a la base B, la solución óptima del dual es $(c_B^t B^{-1})^t = -(\overline{c_{n+1}}, \ldots, \overline{c_{n+m}})^t$. Luego podemos decir que la solución del problema dual está literalmente en la tabla del Símplex del primal. ♢

Antes de continuar es necesario hacer un pequeño inciso.

Observación 3.2.2 (Teorema dual)**.** Existe un "teorema dual" al teorema de dualidad fuerte 3.2.3, cuyo enunciado es el mismo, pero permutando las palabras "primal" y "dual". La demostración es totalmente análoga y se deja al lector.

Esto viene a significar que si alguno de los dos problemas tiene solución óptima, el otro también la tendrá. ♢

Otro resultado bastante fuerte que relaciona las soluciones de los problemas primal y dual es el llamado "teorema de la holgura complementaria", que da condiciones necesarias y suficientes para que dos soluciones (una del primal y otra del dual) sean óptimas simultáneamente.

Teorema 3.2.4 (Teorema de la holgura complementaria)**.** *Sean x e u soluciones factibles de primal y dual respectivamente. Se verifica que x e u son soluciones óptimas de sus respectivos problemas si y solo si se cumplen*

1. *Para cada restricción del problema primal (suponemos la i–ésima con $i \in \{1, \ldots, m\}$) se verifica que $\sum_{j=1}^{n} a_{ij} x_j = b_i$ o bien $u_i = 0$.*
2. *Para cada restricción del problema dual (suponemos la j–ésima con $j \in \{1, \ldots, n\}$) se verifica que $\sum_{i=1}^{m} a_{ij} u_i = c_j$ o bien $x_j = 0$.*

Demostración. Como por el lema 3.2.1 sabemos que $c^t x \leq b^t u$, lo cual implica que tanto el problema primal como el problema dual tienen solución óptima (por no ser ni infactibles ni no acotados, véase teorema 1.3.3).

Dicho esto, el teorema 3.2.3 combinado con el corolario 3.2.2 nos asegura que ser soluciones óptimas simultáneas es equivalente a que se verifique la igualdad $c^t x = b^t u$. Busquemos condiciones necesarias y suficientes para que esto se cumpla.

Podemos resumir el contenido fundamental del lema 3.2.1 en la siguiente cadena de desigualdades.

$$c^t x = \sum_{j=1}^{n} c_j x_j \leq \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} u_i \right) x_j = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j \right) u_i \leq \sum_{i=1}^{m} b_i u_i = b^t u$$

Una condición suficiente para que se de la igualdad en la primera desigualdad de la cadena es que $c_j x_j = (\sum_{i=1}^{m} a_{ij} u_i) x_j$ para $j \in \{1, \ldots, n\}$, lo cual se da si y solo si $c_j = \sum_{i=1}^{m} a_{ij} u_i$ o bien $x_j = 0$. Encontramos una condición suciente análoga para la segunda desigualdad de la cadena.

La condición que hemos impuesto también es necesaria, ya que $c_j \leq \sum_{i=1}^{m} a_{ij} u_i$ y $x_j \geq 0$ para todo $j \in \{1, \ldots, n\}$, de este modo es fácil ver que si se diera la desigualdad estricta para un j en el cual $x_j > 0$ se produciría una desigualdad estricta también en el sumatorio. Análogamente se hace con la otra desigualdad. ■

Aunque este último teorema pueda parece más inútil que un cubo de tela, no es así, pues, como vemos a continuación, sirve para comprobar si una solución (no necesariamente punto extremo) es óptima o no sin necesidad de aplicar el algoritmo del Símplex. Veamos este resultado en forma de una observación y un ejemplo.

Observación 3.2.3 (Test de optimalidad)**.** Una solución factible x del problema primal es óptima si y solo si existe un $u \in \mathbb{R}^m$ que verifica las siguientes condiciones

1. Si $\sum_{j=1}^{n} a_{ij}x_j < b_i$ entonces $u_i = 0$ y si $x_j > 0$ entonces $\sum_{i=1}^{m} a_{ij}u_i = c_j$.
2. Se verifica que $\sum_{i=1}^{m} a_{ij}u_i \geq c_j$ y $u_i \geq 0$ para todo $j \in \{1, \ldots, m\}$.

La corrección de este "test" se deduce trivialmente de los teoremas 3.2.3 y 3.2.4 junto con el corolario 3.2.2. Se recomienda encarecidamente al lector poner a prueba esta observación en la práctica. ♢

Para finalizar la sección presentemos una tabla resumen que recopila casi todo lo aprendido.

	Infactible	Sol. óptima	Sol. no acotada
Infactible	**Posible**		**Posible**
Sol. óptima		**Posible**	
Sol. no acotada	**Posible**		

Tabla 3.1: Relaciones de dualidad.

Como se puede observar, la tabla es totalmente simétrica, motivo por el cual no se ha especificado a en ningún letrero si nos referimos al problema dual o al problema primal.

La demostración de que la tabla es verídica es muy sencilla, siendo la demostración de la veracidad de la fila central el contenido de la observación 3.2.2. Por su parte, la demostración correspondiente a la última fila se basa en el lema 3.2.1. Es muy sencilla y se deja al lector. En cuanto a la primera fila, se pueden encontrar ejemplos de los dos casos posibles, mientras que el caso otro caso es imposible por la observación 3.2.2.

3.3. Otras formulaciones

Vamos a extender la noción "problema dual asociado" a problemas que no necesariamente están presentados en forma canónica.

3.3.1. Formulación estándar

Dado un problema de programación lineal en forma estándar

$$\begin{array}{c} \text{mín } c^t x \\ \text{Sujeto a:} \quad Ax = b, \qquad x \geq 0 \end{array}$$

vamos a ponerlo en forma canónica de minimización usando el siguiente truco. Consideramos el problema equivalente

$$\begin{array}{c} \text{mín } c^t x \\ \text{Sujeto a:} \quad Ax \leq b \quad \& \quad Ax \geq b, \qquad x \geq 0 \end{array}$$

que a su vez transformamos en

$$\begin{array}{c} \text{mín } c^t x \\ \text{Sujeto a:} \quad -Ax \geq -b \quad \& \quad Ax \geq b, \qquad x \geq 0 \end{array}$$

que ya está en forma canónica, por simplificar un poco las cosas explicitamos que la matriz de este problema es $(A^t| - A^t)^t$ y su vector de términos independientes $(b^t| - b^t)^t$. De esta forma ya podemos calcular el problema dual con la definición usual. Este es

$$\begin{array}{c} \text{máx}(b^t| - b^t)^t u \\ \text{Sujeto a:} \quad (A^t| - A^t)u \leq c \qquad u \geq 0 \end{array}$$

Si llamamos u_1 a las m componentes de u asociadas a la submatriz A^t, mientras llamamos u_2 a las componentes asociadas a la submatriz restante tenemos que el problema dual tiene por restricción

$A^t u_1 - A^t u_2 \leq c$, que es equivalente a la restricción $A^t(u_1 - u_2) \leq c$. Algo similar se hace con la función objetivo.

Llamando $w := u_1 - u_2$ tenemos que el problema dual es

$$\begin{array}{c} \text{máx } b^t w \\ \text{Sujeto a:} \qquad A^t w \leq c \end{array}$$

Nótese que como $u_1, u_2 \geq 0$, w no tiene ninguna restricción en cuanto al signo.

Como la involutividad se sigue cumpliendo, ya tenemos una forma de "dualizar" problemas de maximización con variables no restringidas. Si uno no quiere gastar memoria en esto, basta con reproducir el procedimiento realizado y leerlo de abajo a arriba.

Un buen ejercicio consistiría en repetir el apartado entero cambiando el problema inicial a dualizar por un problema en forma estándar pero de maximización.

3.3.2. Formulación general

En este apartado vamos a ver cómo dualizar cualquier problema de programación lineal (independientemente de su formulación), para lo cual seguiremos una estrategia totalmente análoga a la del apartado anterior.

Uno de los problemas más generales que se nos puede plantear es el que tiene restricciones de todo tipo, es decir

$$\begin{array}{c} \text{máx } c^t x \\ \text{Sujeto a:} \qquad A_1 x \leq b \quad \& \quad A_2 x = b \quad \& \quad A_3 x \geq b, \qquad x \geq 0 \end{array}$$

Si lo transformamos en un problema en forma canónica obtenemos (¡compruébese!)

$$\begin{array}{c} \text{máx } c^t x \\ \text{Sujeto a:} \qquad (A_1^t | A_2^t | - A_2^t | - A_3^t)^t x \leq (b_1^t | b_2^t | b_3^t | b_4^t)^t \qquad x \geq 0 \end{array}$$

por tanto su dual será (háganse las cuentas)

$$\begin{array}{c} \text{mín}(b_1^t | b_2^t | b_3^t | b_4^t)(u_1^t | u_2^t | u_3^t | u_4^t)^t \\ \text{Sujeto a:} \qquad (A_1^t | A_2^t | - A_2^t | - A_3^t)(u_1^t | u_2^t | u_3^t | u_4^t)^t \geq c \qquad u \geq 0 \end{array}$$

Por ende la restricción del problema dual es equivalente a $A_1^t u_1 + A_2^t(u_2 - u_3) + A_3^t(-u_4) \geq c$. Llamando $w_1 := u_2 - u_3$ y $w_2 := -u_4$ obtenemos que el problema dual tiene a $A_1^t u_1 + A_2^t w_1 + A_3^t w_2$ por restricción, donde w_1 son variables no restrigidas y w_2 son variables no positivas. Algo similar se hace con la función objetivo.

Podemos realizar el mismo proceso con un problema de minimización, y en general con cualquier problema, ya que las anomalías de tener variables no restringidas o no positivas se trataron en el capítulo 1, de modo que podemos transformar cualquier problema en uno del tipo tratado en este apartado o su análogo de minimización.

3.4. Algoritmo dual

Resolver problemas de programación lineal en forma canónica de minimización con vector de términos independientes $b \geq 0$ utilizando el algoritmo de Símplex provoca un desperdicio de memoria que da pena verlo.

Esto es debido a que al realizar la transformación del problema $Ax \geq b$ a forma estándar nos encontramos con que la matriz asociada al problema pasa a ser $(A| - I_m)$, lo cual arroja una base trivial del problema, sin embargo, esta base no nos sirve, ya que no está asociada a ningún punto extremo. En efecto $(-I_m)^{-1} b = -I_m b = -b \leq 0$.

Esto nos obliga a usar algún procedimiento de inicialización como el método de las dos fases o el método de las penalizaciones, lo cual supone añadir m columnas más a la tabla, lo cual da rabia, habiendo estado tan cerca de conseguir una base trivial sin hacer nada.

Es de esta frustración que nace el llamado "algoritmo dual" en los años 90, que resuelve estas situaciones más ágilmente, exprimiendo las propiedades de dualidad vistas hasta ahora.

3.4.1. Fundamentación teórica

En este apartado desarrollaremos otro algoritmo para resolver problemas de programación lineal en forma estándar que, como se comentó antes, en ciertas situaciones es más eficiente.

Sea el problema de programación lineal en forma estándar

$$\begin{array}{c} \min c^t x \\ \text{Sujeto a:} \quad Ax = b, \qquad x \geq 0 \end{array}$$

cuyo dual, convenientemente reescrito es

$$\begin{array}{c} \max \lambda^t b \\ \text{Sujeto a:} \quad \lambda^t A \leq c^t \end{array}$$

Supongamos que disponemos de una base inicial B de la matriz A de forma que todos los costes reducidos asociados al problema primal son positivos, es decir

$$\overline{c_j} = c_j - c_B^t B^{-1} a_j \geq 0$$

De ser esto así, si consideramos el vector $\lambda^t := c_B^t B^{-1}$ es sencillo deducir que λ^t es una solución factible del problema dual. En efecto

$$\begin{aligned} \lambda^t A = c_B^t B^{-1}(B|N) = c_B^t B^{-1}(a_1 \cdots a_m | a_{m+1} \cdots a_n) = \\ = (c_B^t B^{-1} a_1 \cdots c_B^t B^{-1} a_m | c_B^t B^{-1} a_{m+1} \cdots c_B^t B^{-1} a_n) = \\ = (c_1 - \overline{c_1} \cdots c_m - \overline{c_m} | c_{m+1} - \overline{c_{m+1}} \cdots c_n - \overline{c_n}) \leq (c_1 \cdots c_n) = c^t \end{aligned}$$

Nótese que como $\overline{c_j} = 0$ cuando a_j es una columna de la base se tiene que $\lambda^t a_j = c_j$ para las columnas asociadas a B, mientras que $\lambda^t a_j \leq c_j$ para las columnas no asociadas a B.

Dicho esto, consideremos el vector $\overline{x} := ((B^{-1}b)^t | 0)^t$, que es solución del problema primal. Nótese que en ningún momento decimos que $\overline{x}$ sea solución **factible**, ya que no sabemos si $\overline{x} \geq 0$. No obstante, sería maravilloso que esto ocurriera, como vemos a continuación.

Lema 3.4.1 (Test de optimalidad)**.** *Si $\overline{x} \geq 0$ entonces $\overline{x}$ es solución óptima del problema primal.*

Demostración. Es evidente ya que en tal caso $c^t \overline{x} = c_B^t B^{-1} b = \lambda^t b$. Es decir, ambas soluciones factibles tienen el mismo valor al ser pasados por sendas funciones objetivo. El corolario 3.2.2 nos da el resto del resultado. ■

No obstante, esto no suele ser lo habitual. Así pues, la estrategia a seguir cuando tengamos que $\overline{x}$ no es solución factible será cambiar de base, es decir, construir a partir de B una nueva base B' sobre la que continuar probando suerte. Nótese que es una filosofía similar a la que sigue el algoritmo del Símplex primal. Supongamos pues que $\overline{x_l} < 0$.

En este caso consideraremos el vector $\lambda_\varepsilon^t := \lambda^t - \varepsilon u_l^t$ donde u_l^t es la l–ésima fila de B^{-1}. A lo largo del apartado determinaremos un $\varepsilon > 0$ adecuado, de forma que λ'_ε sea una solución factible del problema dual y además exista una base B' de A de forma que $\lambda_\varepsilon^t = c_{B'}^t B'^{-1}$, y, por ende, todos los costes reducidos asociados a B' sean no negativos (compruébese), pudiéndose volver a ejecutar todo el proceso de nuevo.

Observación 3.4.1 (Igualdades útiles)**.** Es sencillo comprobar que $u_l^t a_j = y_{lj}$. Esto se debe a que, por definición $B^{-1} a_j = y_j$. Si expresamos la matriz B^{-1} descompuesta en filas obtenemos la expresión $(u_1 \cdots u_m)^t a_j = y_j$, en concreto, $y_{lj} = u_l^t a_j$. Análogamente se demuestra que $u_l^t b = \overline{x_l}$. Usaremos estas igualdades a menudo. ♢

Para que λ_ε^t sea solución factible dual se debe cumplir que $\lambda_\varepsilon^t A \leq c^t$, o equivalentemente, que $\lambda_\varepsilon^t a_j \leq c_j$ para todo $j \in \{1, \ldots, n\}$. Vayamos caso por caso para ver qué ε sería bueno escoger.

- Para $j \in \{1, \ldots, m\} \setminus \{l\}$ tenemos que

$$\lambda_\varepsilon^t a_j = (\lambda^t - \varepsilon u_l^t) a_j = \lambda^t a_j - \varepsilon \cancel{y_{lj}} = \lambda^t a_j = c_j$$

  luego para este caso el ε que cojamos da un poco igual.

- Por su parte para $j = l$ tenemos

$$\lambda_\varepsilon^t a_l = \lambda^t a_l - \varepsilon \cancel{y_{ll}} = \lambda^t a_l - \varepsilon = c_j - \varepsilon \leq c_j$$

concluyendo que para este caso también da un poco igual.

- Por último consideramos el caso $j \in \{m+1, \ldots, n\}$

$$\lambda_\varepsilon^t a_j = \lambda^t a_j - y_{lj}\varepsilon \leq c_j - y_{lj}\varepsilon$$

este caso ya es más delicado, ya que si $y_{lj} \geq 0$ podemos coger el ε que queramos, no obstante, si este caso no se da, debemos coger ε con mucho cuidado.

Al hilo de esto, veamos qué ocurre si todos los y_{lj} son no negativos.

Lema 3.4.2 (Test de infactibilidad)**.** *Si $y_{lj} \geq 0$ para todo $j \in \{m+1, \ldots, n\}$ entonces el problema primal es infactible.*

Demostración. Si consideramos el vector λ_ε^t tenemos que es una solución factible del problema dual, ya que

$$\lambda_\varepsilon^t A = \lambda_\varepsilon^t (a_1 \cdots a_m | a_{m+1} \cdots a_n) = (\lambda_\varepsilon^t a_1 \cdots \lambda_\varepsilon^t a_m | \lambda_\varepsilon^t a_{m+1} \cdots \lambda_\varepsilon^t a_n)$$

y resulta que $\lambda_\varepsilon^t a_j = \lambda^t a_j - \varepsilon y_{lj} \leq \lambda^t a_j \leq c_j$. Esta desigualdad también se da para los $j \in \{1, \ldots, m\}$ ya que $y_{lj} \in \{0, 1\}$ para esos índices.

Además, se tiene que $\lambda_\varepsilon^t b = \lambda^t b - \varepsilon u_l^t b = \lambda^t b - \varepsilon \overline{x_l}$. Como $\overline{x_l} < 0$ es claro que $\lim_{\varepsilon \to \infty} \lambda_\varepsilon^t b = \infty$, siendo así el problema dual no acotado, y, por tanto (ver la tabla 3.1), el problema primal será infactible. ■

Llegados a este punto únicamente nos queda suponer que hay algún a_j con $y_{lj} < 0$ (con $j \in \{m+1, \ldots, n\}$). Veamos qué ε coger. Como ya vimos $\lambda_\varepsilon^t a_j = \lambda^t a_j - y_{lj}\varepsilon$. Desarrollando a partir de aquí tenemos que

$$\lambda^t a_j - y_{lj}\varepsilon \leq c_j \iff -\frac{c_j - \lambda^t a_j}{y_{lj}} \geq \varepsilon \iff \varepsilon \leq -\frac{\overline{c_j}}{y_{lj}}$$

En previsión de que haya varios $y_{lj} < 0$ tomaremos ε como

$$\varepsilon := -\frac{\overline{c_k}}{y_{lk}} := \text{mín}\left\{-\frac{\overline{c_j}}{y_{lj}} \ : \ y_{lj} < 0\right\}$$

De esta forma tenemos que λ_ε^t es una solución factible dual, que en particular cumple que $\lambda_\varepsilon^t a_j = c_j$ para $j \in (1, \ldots, m \setminus \{l\}) \cup \{k\}$. Esta propiedad es crucial, ya que si consideramos el conjunto de columnas $B' := (B \setminus \{a_l\}) \cup \{a_k\}$ (que es una base, se comprueba como en la demostración del teorema 1.3.6) tenemos

$$\lambda_\varepsilon^t B' = c_{B'}^t \iff \lambda_\varepsilon^t = c_{B'}^t B'^{-1}$$

Luego hemos ganado, y además, de una forma objetivamente bonita.

Falta sin embargo comprobar que con este cambio de base estamos acercándonos más a la solución óptima. Una forma de verlo, usando dualidad, es que estamos mejorando la solución del problema dual, y, en efecto

$$\lambda_\varepsilon^t b = \lambda^t b - \varepsilon \overline{x_l} \geq \lambda^t b$$

Otro detalle a tener en cuenta es que puede ocurrir que haya varias componenetes negativas de $\overline{x}$. En este caso ¿cuál elegimos para salir de la base? La respuesta es que da un poco igual, no obstante, podemos adoptar un análogo a la vieja y confiable regla de Dantzig, es decir, coger la más pequeña de las componentes negativas.

Observación 3.4.2 (Bucles)**.** Lamentablemente, en el algoritmo dual también pueden producirse bucles, en concreto en la situación en la que tomamos $\varepsilon = 0$. Para solucionar estas incómodas situaciones se recomienda usar la regla de Bland. Habría que demostrar que esta regla sigue siendo efectiva en este contexto, lo cual se deja al lector ya que la demostración es bastante simétrica. ♢

Dicho lo cual, podemos pasar a resumirlo todo en un bonito algoritmo.

Algoritmo 4 Primera aproximacíon al algoritmo del Símplex dual.

Entrada: Base B de A con $\overline{c} \geq 0$.
Salida: Solución óptima del problema.

```
Aplicar el lema 3.4.1 para detectar la optimalidad de x̄.
si x̄ es solución óptima entonces
    devolver x̄.
si no
    Aplicar el lema 3.4.2 para detectar la infactibilidad.
    si el problema es no factible entonces
        devolver infactible
    si no
        Cambiar de base según lo explicado anteriormente.
        devolver B', la nueva base.
    fin si
fin si
```

Nótese que podemos aplicar el algoritmo del Símplex dual sobre una tabla del Símplex normal y corriente, realizandose los cambios de base de la misma manera, lo cual, de nuevo, es una bendición de los dioses. Queda, no obstante, ver cómo nos las apañamos para obtener la primera base B, cosa que veremos en la siguiente sección.

Observación 3.4.3 (Maximización)**.** Se puede realizar una adaptación similar a la que ya se hizo con el problema primal para que el algoritmo dual sea también capaz de resolver problemas de maximización. Se deja como ejercicio al lector. ♢

3.4.2. Método de la restricción artificial

En este apartado veremos una técnica para conseguir la base inicial necesaria para ejecuta el algoritmo dual. Supongamos que ya disponemos de una base B de tal forma que hay algún coste reducido negativo.

Observación 3.4.4 (Suposiciones)**.** Suponer que se tiene una base inicial B no es mucho suponer, pues este suele ser el caso de los problemas en los que merece la pena aplicar el algoritmo dual (forma canónica de minimización). ♢

Supuesto esto, nos panteamos el problema, al que llamaremos ***problema artificialmente restringido***, o símplemente problema restringido, que se define como

$$\begin{array}{c} \min c^t x \\ \text{Sujeto a:} \qquad Ax \leq b \quad \& \quad \sum_{j=m+1}^{n} x_j \leq M, \qquad x \geq 0 \end{array}$$

donde M es un número lo suficientemente grande. Una vez transformado el problema restringido a forma estándar, la matriz del problema restringido tendrá una fila y una columna más correspondientes a la nueva restricción y a su correspondiente variable de holgura.

La columna añadida es linealmente independiente a las demás (compruébese), por lo que deberá ser incluída en la base.

Hecho esto, realizamos el cambio de base consistente en expulsar a la columna correspondiente a la variable de holgura asociada a la restricción artificial (a la cual le asignamos el índice 0) y admitir la columna con menor coste reducido (supondremos que es la columna a_k).

Lema 3.4.3 (Base inicial)**.** *La base obtenida tras es el cambio hace que el vector de costes reducidos sea no negativo.*

Demostración. En efecto, la matriz asociada al problema original es $A = (B|N)$, siendo la matriz asociada al problema restringido

$$\overline{A} = \left(\begin{array}{c|c|c} 1 & 0 & 1 \\ \hline 0 & B & N \end{array}\right)$$

siendo la base $\overline{B}$ del problema restringido la matriz $\overline{A}$ excluyendo la última columna por bloques.

Veamos cómo es la tabla del símplex asociada al problema restringido. Para lo cual, deberemos calcular la inversa de $\overline{B}$. Con un sencillo cálculo llegamos a que

$$\overline{B}^{-1} = \left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & B^{-1} \end{array}\right)$$

de forma que la tabla del Símplex asociada al problema restringido es

$$\overline{B}^{-1}\overline{A} = \left(\begin{array}{c|c|c} 1 & 0 & 1 \\ \hline 0 & I_m & B^{-1}N \end{array}\right)$$

de aquí concluimos que los escalares $y_{0j} = 1$ para $j \in \{m+1, \ldots, n\}$, dicho esto, de las fórmulas de cambio de base se deduce que los costes reducidos a la base $\overline{B}$ son $\overline{c_j'} = \overline{c_j} - \overline{c_k}$ (compruébese).

Como $\overline{c_k} = \text{mín}\{\overline{c_j} \ : \ \overline{c_j} < 0\}$ se tiene que todos los costes reducidos son no negativos. ■

A partir de aquí podemos aplicar el algoritmo del Símplex dual para resolver el problema restringido. Dicho algoritmo puede acabar de dos formas distintas.

1. El problema restringido es infactible.
2. El problema restringido tiene solución óptima.
   - *a*) La columna a_0 está en la última base considerada por el algoritmo.
   - *b*) La columna a_0 no está en la última base considerada por el algoritmo.

Veamos cuáles son las consecuencias de cada uno de estos finales para el problema original.

Supongamos que se da el caso 1. En tal caso tenemos el siguiente lema.

Lema 3.4.4 (Test de factibilidad)**.** *El problema original es factible si y solo si el problema restringido lo es.*

Demostración. Es claro que si el problema restringido es factible, también lo es el original, ya que tiene menos restricciones.

Recíprocamente, si el problema original es factible, digamos que $\overline{x}$ es una solución factible del problema original (sin pérdida de generalidad un punto extremo), como escogimos M lo suficientemente grande, es claro que $\overline{x}$ también verifica la restricción adicional. ■

Observación 3.4.5 (Valor de M)**.** La demostración anterior puede parecer tramposa, pero realmente no lo es, ya que, como hicimos con el método de las penalizaciones para el algoritmo del Símplex primal, para cada problema podemos escoger un M finito de forma que todos los teoremas funcionen bien.

La existencia de dicho M finito ha de ser demostrada, buscando cotas para el mismo cada vez que aludimos a su "grandeza" para demostrar algo.

Por poner un ejemplo, en el caso de 3.4.4 necesitamos que

$$M > \text{máx}\left\{\sum_{j=m+1}^{n} \overline{x_j} \ : \ \overline{x} \text{ punto extremo}\right\} \qquad \diamondsuit$$

Continuando con el estudio de la cauística, si se diera la situación $2a$ se puede demostrar (no lo haremos) que el problema original tiene solución óptima, y, de hecho, si el algoritmo determina que la solución óptima del problema restringido se alcanza en el punto $(x_0, \ldots, x_n)$, la solución óptima del problema original se alcanzará en $(x_1, \ldots, x_n)$.

Por último, si se cumpliera $2b$, cabe distinguir dos opciones

- El valor de la función objetivo depende de M.
- El valor de la función objetivo no depende de M.

Observación 3.4.6 (Dependencia de la M)**.** Las alternativas que acabamos de exponer no pueden considerarse en el caso $2a$, ya que el caso de que el valor de la función objetivo dependa de M es imposible. En efecto, si consideramos que la última base examinada es $B' := B \cup \{a_0\}$ (donde B es un conjunto de columnas linealmente independientes) tendríamos que $c^t x = c^t_{B'} \overline{x_{B'}} = (0|c_B)^t B'^{-1}(M|b^t)^t$, si miramos esto con un poco de cariño tenemos que

$$B' := \left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & B \end{array}\right), \quad B'^{-1} = \left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & B^{-1} \end{array}\right), \quad B'^{-1}(M|b^t)^t = (M|(B^{-1}b)^t)^t$$

de forma que $c^t x$ no depende de M en ningún caso, nótese que se ha usado fuertemente que $c_0 = 0$, por lo que esta demostración no sirve para los casos en los que a_0 no está en la base. ♢

Finalizado el inciso, supongamos que el valor de la función objetivo depende de la M. En tal caso, puede demostrarse, aunque no lo haremos aquí, que el problema original es no acotado.

Por último, si la función objetivo no depende de la M, puede demostrarse (no lo haremos) que el problema orginal alcanza la solución óptima, y además, en un número infinito de puntos.

Parte II

Introducción a la programación entera

Capítulo 4

El problema de asignación

El objetivo de este capítulo es desarrollar un algoritmo que permita resolver problemas de asignación, unos de los más aparentemente sencillos y más habituales problemas de programación entera.

4.1. Unimodularidad total

Definición 4.1.1 (Unimodularidad total)**.** Una matriz $A \in \mathfrak{M}_{m\times n}(\mathbb{Z})$ se dice ***totalmente unimodular*** si todas sus submatrices cuadradas tienen determinante 0, 1 ó -1.

Observación 4.1.1 (Consecuencia inmediata)**.** Dado un coeficiente a_{ij} de una matriz totalmente unimodular, se debe cumplir que $a_{ij} \in \{0, 1, -1\}$. Esto es debido a que los elementos de una matriz son considerados submatrices cuadradas de orden 1. ♢

Veamos algunas propiedades elementales de las matrices totalmente unimodulares.

Lema 4.1.1 (Trasposición)**.** *A es totalmente unimodular si y solo si A^t también lo es.*

Demostración. Sabemos que una submatriz cuadrada de orden k queda caracterizada por la elección de k filas y k columnas de la matriz original. Es sencillo demostrar (se deja al lector) que la aplicación que, dada una elección de filas y columnas $(\{i_1, \ldots, i_k\}, \{j_1, \ldots, j_k\})$ de la matriz A le hace corresponder la elección $(\{j_1, \ldots, j_k\}, \{i_1, \ldots, i_k\})$ de la matriz A^t, es una biyección.

De esta forma se tiene que si B es una submatriz cuadrada de A, entonces B^t es una submatriz cuadrada de A^t. Y como $\det(B) = \det(B^t)$ se tiene que A^t es totalmente unimodular. ■

Lema 4.1.2 (Extensión)**.** *A es totalmente unimodular si y solo si $(A|I)$ también lo es.*

Demostración. La implicación a la izquierda es evidente.

Para la otra, notemos que si tomamos una submatriz cuadrada de $(A|I)$ se pueden dar tres casos posibles. El primero de ellos es que cojamos una submatriz de A, luego no hay nada que probar. La segunda situación posible es que la matriz tenga una columna de ceros, en cuyo caso tampoco tenemos que hacer nada, ya que dicha matriz tendrá determinante nulo.

La situación interesante es la de tener una o varias columnas formadas por ceros salvo un uno. En este caso, desarrollamos el determinante por estas columnas, nótese que, una vez agotadas, nos quedará el determinante de una submatriz cuadrada de A (quizá multiplicado por -1). ■

Vistas estas propiedades elementales, veámos cuál es la verdadera razón por la cual vale la pena estudiar estas matrices.

Proposición 4.1.3 (Puntos extremos)**.** *Si $A \in \mathfrak{M}_{m\times n}(\mathbb{Z})$ es totalmente unimodular y cumple las hipótesis de 1.2.3 se tiene que, para todo $b \in \mathbb{Z}^m$, todos los puntos extremos $\overline{x} \in \mathbb{R}^n$ de la región factible asociada al problema $Ax = b$ con $x \geq 0$ son enteros, es decir $\overline{x} \in \mathbb{Z}^n$.*

Demostración. Sea B una base de A asociada a un punto extremo, como A es totalmente unimodular y B es invertible, se tiene que $\det(B) = \pm 1$. El punto extremo asociado a B es pues la única solución de la ecuación $Bx_B = b$.

Usando la regla de Cramer tenemos que $x_B^i = \frac{\det(Bi)}{\det B}$, donde B_i denota a la matriz B con la columna i–ésima sustituida por b. Evidentemente x_B^i es un entero para todo i, y, por tanto, el punto extremo es entero. ■

El recíproco de la proposición 4.1.3 se da, de hecho se da algo más fuerte. Es un teorema con nombre que no demostraremos aquí, pero sí enunciaremos.

Teorema 4.1.4 (Teorema de Hoffman–Kruskal)**.** *Si $A \in \mathfrak{M}_{m\times n}(\mathbb{Z})$ en las las hipótesis de 1.2.3 y para todo $b \in \mathbb{Z}^m$ la región factible tiene al menos un punto extremo entero, se cumple que A es totalmente unimodular.*

Tras este bonito y útil problema, pasemos a estudiar una condición suficiente para detectar la unimodularidad total de una matriz.

Proposición 4.1.5 (Condición suficiente)**.** *Si A verifica las siguientes condiciones, entonces es totalmente unimodular.*

1. *Todo coeficiente $a_{ij} \in \{0, 1, -1\}$.*
2. *Cada columna tiene a lo sumo dos elementos no nulos.*
3. *Podemos realizar una bipartición de las filas de A de manera que*
   a) *Si los dos elementos no nulos de una columna tienen el mismo signo, entonces las filas asociadas a dichos elementos pertenecen cada una a un cojunto de la bipartición.*
   b) *Si los dos elementos no nulos de una columna tienen distinto signo, entonces las filas asociadas a dichos elementos pertenecen al mismo conjunto de la bipartición.*

Demostración. Es muy sencillo probar esto por inducción sobre la dimensión de la submatriz cuadrada (se deja como ejercicio al lector). ■

La proposición 4.1.5 tiene su eco en la teoría de grafos. Como vemos a continuación.

Definición 4.1.2 (Matriz de incidencia vértices–aristas)**.** Dado un grafo dirigido G que no contiene autolazos, se define su ***matriz de incidencia vértices–aristas*** como la matriz que tiene tantas columnas como aristas tenga el grafo y tantas filas como vértices, y, además, $a_{ij} \in \{0, 1, -1\}$ donde si $a_{ij} = 0$ la arista j no sale ni entra del vértice i. En caso de que $a_{ij} = 1$, la arista j parte del vértice i, y, si $a_{ij} = -1$, la arista i muere en el vértice i.

En el caso de grafos no dirigidos la matriz se define eliminando el caso $a_{ij} = -1$, significando $a_{ij} = 1$ que la arista j incide sobre el vértice i.

Observación 4.1.2 (Grafos y unimodularidad total)**.** Es evidente que la matriz de incidencia vértices–aristas de un grafo dirigido sin autolazos es totalmente unimodular, ya que cumple las condiciones de la proposición 4.1.5. Nótese que una de las biparticiones será el vacío mientras que la otra será el total. ♢

4.2. Emparejamiento máximo de un grafo bipartito

En esta sección usaremos toda la artillería de la anterior para encontrar un algoritmo que encuentre el emparejamiento máximo de un grafo bipartito en un tiempo razonable. En primer lugar, definamos en detalle el problema a resolver, para ello, introduzcamos unas cuantas definiciones.

Definición 4.2.1 (Emparejamiento)**.** Dado un grafo arbitrario G, decimos que un conjunto M de aristas es un ***emparejamiento*** en G si para cada par de aristas de M, no son adyacentes, es decir, sus conjuntos de vértices adyacentes son disjuntos.

Definición 4.2.2 (Saturación y exposición)**.** Dado un grafo G y un emparejamiento M, se dice que un vértice v es M**−*saturado*** si hay alguna arista de M que incide sobre v. En caso contrario se dice que v es M**−*expuesto***.

Definición 4.2.3 (Emparejamiento perfecto)**.** Se dice que un emparejamiento M es ***perfecto*** si todo vértice es M–saturado. Evidentemente, un emparejamiento es perfecto si y solo si contine tantas aristas como la mitad de vértices de G.

Definición 4.2.4 (Cadenas)**.** Dado un grafo G, una ***cadena*** o ***camino*** es una sucesión finita de vértices donde todo vértice está conectado con su antecesor y su predecesor. Si el primer y último vértice de la cadena coinciden, se dice que la cadena es un ***ciclo***.

Definición 4.2.5 (Cadenas alternantes y de aumento)**.** Dado un grafo G, un emparejamiento M y una cadena σ, se dice que σ es una ***cadena*** M**−*alternante*** si las aristas que hay que recorrer para seguir el camino de principio a fin pertenecen a M y a su complementario alternativamente.

Si además se verifica que el primer y último vértice de σ son diferentes y M–expuestos, entonces se dice que σ es una ***cadena de*** M**−*aumento***.

Observación 4.2.1 (Justificación del nombre)**.** Las cadenas de M–aumento reciben ese nombre ya que, dada una cadena de M–aumento, si quitamos de M las aristas de la cadena pertenecientes a M y ponemos las no pertenecientes a M hemos conseguido aumentar en 1 el cardinal del emparejamiento M. ♢

Veamos a continuación un resultado auxiliar que será de gran utilidad en el futuro inmediato.

Lema 4.2.1 (Componentes conexas)**.** *Dado un grafo G y dos emparejamientos M_1 y M_2, considerando el grafo $\overline{G}$, idéntico a G salvo por el hecho de que se le han sustraído las aristas que no pertenecen ni a M_1 ni a M_2 o que pertenecen a ambos emparejamientos.*

Se tiene que las componentes conexas de $\overline{G}$ son de alguno de los siguientes tipos

1. *Vértice aislado.*
2. *Ciclo M_1–alternante con un número par de vértices.*
3. *Cadena M_1–alternante.*

Demostración. Es claro que tanto los ciclos como las cadenas deben ser M_1–alternantes (o M_2–alternantes), ya que si no, de un vértice saldrían dos aristas de un mismo emparejamiento, lo cual contradice la definición de emparejamiento.

Asimismo, si hubiera una componente conexa de cualquier otro tipo, habría un vértice del cual saldrían tres o más aristas, con lo que, por el lema del palomar, al menos dos deben ser de un mismo emparejamiento, lo cual es, de nuevo, absurdo.

Por último, si hubiera algún ciclo con un número impar de vértices, habría también un número impar de aristas, por lo que, forzosamente, debe haber dos aristas del mismo emparejamiento adyacentes a algún vértice (contradicción). ■

Definición 4.2.6 (Emparejamiento de máximo cardinal)**.** Un emparejamiento M se dice ***de máximo cardinal*** si no hay ningún emparejamiento con más aristas.

A continuación viene uno de los dos teorema importantes de esta sección, un teorema "con nombre" que da condiciones necesarias y suficientes para que un emparejamiento sea de máximo cardinal.

Teorema 4.2.2 (Teorema de Berge (1957))**.** *Un emparejamiento M es de máximo cardinal si y solo si no existe ninguna cadena de M–aumento.*

Demostración. La implicación a la derecha es evidente. Veamos pues el recíproco, probando el contrarrecíproco (válgame la redundancia).

Suponiendo que M no es un emparejamiento de máximo cardinal, supongamos que M' es un emparejamiento más grande que M.

Considerando el grafo del lema 4.2.1 tenemos que $\overline{G}$ tiene más aristas de M' que de M (compruébese, es claro). Adicionalmente se tiene que debe haber una componente conexa de tipo cadena que tenga más aristas de tipo M' que de tipo M.

Nótese que no puede ser una componente de tipo ciclo, pues estos deben tener el mismo número de aristas de cada emparejamiento (compruébese).

Claramente la cadena debe ser de longitud impar (por la misma razón que la compoente no puede ser de tipo ciclo), y por ser M'–alternante y tener más vértices de M' que de M, debe ser una cadena de M–aumento. ■

Antes de pasar a ver el último y más importante teorema de la sección, introduzcamos algunos conceptos nuevos.

Definición 4.2.7 (Árbol alternante)**.** Dado un grafo G y un empearejamiento M, un árbol T de G (subgrafo sin ciclos) se dice M–***alternante*** si al elegir un vértice distinguido $r \in T$ al que llamaremos ***raíz*** se tiene que

- r es M–expuesto.
- La cadena (se puede demostrar que es única) en T que une cualquier vértice v con la raíz (sin repetir vértices) es M–alternante.

Definición 4.2.8 (Recubrimiento)**.** Un recubrimiento de un grafo es un subconjunto de vértices tales que toda arista del grafo incide sobre alguno de los vértices del conjunto.

Definición 4.2.9 (Grafo bipartito)**.** Un ***grafo bipartito*** es un grafo G tal que contine una bipartición de vértices que verifica que no hay ninguna arista que una dos vértices del mismo conjunto de la partición.

Observación 4.2.2 (Biparticiones)**.** Nótese que la bipartición de una grafo bipartito coincide con la bipartición inducida por la proposición 4.1.5. ♢

Dicho esto, pasamos a enunciar el teorema y demostrar el teorema prometido, usando todo lo visto hasta ahora, cerrando un círculo virtuoso que nos hará sentirnos en paz con el universo.

Teorema 4.2.3 (Teorema de König (1931))**.** *El tamaño del emparejamiento máximo en un grafo bipartito es el mismo que el del recubrimiento mínimo.*

Demostración. El problema del máximo emparejamiento de grafos bipartitos se puede formular como un problema de programación lineal "relajado". Veámoslo.

Supongamos que el gráfo tiene m vértices y n aristas. La función objetivo a maximizar es $\sum_{i=1}^{n} x_i$, donde $x_i \in \{0, 1\}$. Donde $x_i = 0$ representa que la i–ésima arista no está en el emparejamiento, mientras que $x_i = 1$ representa lo contrario.

Las restricciones del problema son que para todo vértice se verifique que no hay más de una arista del emparejamiento incidiendo sobre él. Esto es equivalente a que para vértice se verifique que $\sum_{i \in \delta(v)} x_i \leq 1$, donde $\delta(v)$ es el conjunto de aristas que inciden sobre v.

Evidentemente esto no es un problema de programación lineal al uso ya que las variables son binarias, es decir, únicamente pueden tomar dos valores. Es por esto que consideramos el problema "relajado" en el que símplemente exigimos $x_i \geq 0$.

Una de las cosas bonitas que suceden es que la matriz de restricciones del problema es exactamente la matriz incidencia vértices–aristas del grafo, y, para colofón, esta matriz es siempre totalmente unimodular para grafos bipartitos (compruébese, de hecho, se da el recíproco), lo cual quiere decir, por la proposición 4.1.3, que todos los puntos extremos de este problema son enteros.

Además, las restricciones del problema fuerzan a que sean binarios (compruébese), con lo que la "relajación" efectuada no supone ninguna pérdida de generalidad (objetivamente bonito).

Ahora, nos planteamos el problema dual asociado a este. Este problema tendrá por función objetivo a minimizar $\sum_{j=1}^{m} u_j$. Además, sus restricciones serán que para cada arista $\sum_{j=1}^{m} u_j \geq 1$ con $u_j \geq 0$.

Si miramos esto un poco con lupa nos damos cuenta que esta es una formulación "relajada" del problema de la cobertura de vértices donde $u_j = 1$ representa que el j–ésimo vértice está en la cobertura y $u_j = 0$ representa lo contrario.

Como el problema primal tiene solución óptima, el problema dual también, y con el mismo valor de función objetivo, luego hemos terminado. ■

La belleza de esta demostración prueba que, incluso en esta oscuridad que nos envuelve, todavía hay esperanza. Dicho esto, si el lector ha decidido omitir la demostración se le recomienda encarecidamente que vuelva atrás y la lea.

Observación 4.2.3 ($\mathcal{NP}$–completitud)**.** Se sabe que el problema de la máxima cobertura de vértices es $\mathcal{NP}$–completo. Veamos qué pasaría si el problema de la cobertura de vértices también fuera $\mathcal{NP}$–completo para grafos bipartitos.

En tal caso, el teorema 4.2.3 reduce la resolución del problema de la cobertura de vértices a la del emparejamiento máximo de un grafo bipartito, quedando demostrado que el problema del emparejamiento máximo es $\mathcal{NP}$–completo, lo cual quiere decir que si encontramos un algoritmo polinómico para resolver el problema de emparejamiento máximo, habremos demostrado que $\mathcal{P} = \mathcal{NP}$ y podremos dedicarnos al crimen.

¿Lo hay? La respuesta es sí, este resultado está recogido en un teorema debido a Edmonds (1965). Así pues, todo parece indicar que el caso particular del problema de la cobertura de vértices para grafos bipartitos no es $\mathcal{NP}$–completo. Se anima al lector a investigar más sobre este asunto. ♢

4.2.1. Algoritmo de emparejamiento máximo

Vamos a desarrollar un algoritmo que, dado un emparejamiento de un grafo bipartito, obtenga el emparejamiento máximo. Por simplicidad, inicialmente supondremos que el emparejamiento inicial es vacío, aunque pronto abandonaremos esa hipótesis adicional.

Observación 4.2.4 (Algoritmo alternativo)**.** El teorema 4.2.3 nos dice que podemos aplicar el algoritmo del Símplex corriente y vulgar, no obstante, el algoritmo que presentamos a continuación es mucho más eficiente. ♢

La estraregia del algoritmo es construir un árbol alternante para cada vértice de una de las biparticiones del grafo, detectando cadenas alternantes que permitan ampliar el emparejamiento.

En primer lugar, asignaremos a cada vértice dos propiedades. La propiedad de estar o no "**etiquetado**" y la propiedad de estar o no "**examinado**". Inicialmente, todos los vértices estarán sin examinar y sin etiquetar.

La etiqueta de un vértice se compone de dos componentes, la primera de ellas indica la paridad del vértice, es decir, si quedan un número par o impar de aristas para llegar desde él a la raiz del árbol. La segunda componente indica quién es su predecesor en la cadena que lleva a la raíz.

En nuestra situación inicial, tomamos un vértice arbitrario r (nuestra raíz) y le asignamos la eiqueta $(P, -)$, nótese que no tiene predecesor en la cadena. En este constexto se introduce la definición.

Definición 4.2.10 (Paridad)**.** Un vértice se dice ***par*** si la primera componente de su etiqueta es P, si, por el contrario, la primera componente de la etiqueta es I, se dirá que el vértice es ***impar***.

Dicho esto, el vértice r pasa a estar etiquetado pero no examinado. A continuación, tomamos el conjunto de vértices adyacentes a r y los etiquetamos con (I, r), quedando r examinado. Ahora escogemos uno de los vértices etiquetados (que están en el árbol) y no examinados (que no son la raíz).

Entonces es claro que tenemos una cadena de aumento, la formada por la arista que une la raiz con el vértice escogido, de forma que añadimos la arista al emparejamiento y borramos todas las etiquetas (vamos a pasar a construir un árbol para otro vértice, y las etiquetas son información intrínseca del árbol anterior).

En esta segunda iteración del algoritmo lo primero que hay que comprobar es si quedan vértices de la bipartición de las raices sin examinar (que no hayamos construido ya árboles para todos), en caso contrario, todos los vértices de una de las biparticiones serían saturados, siendo el emparejamiento construído ya óptimo.

En caso de que esto último no suceda, repetimos la misma operación de antes, eligiendo una raiz y etiquetando a ella y a sus hijos debidamente, quedando la raiz examinada (nótese que el proceso de examinado, al contrario que el de etiquetado, es irreversible).

Ahora se pueden dar varios casos antes imposibles, pudiera ocurrir que al elegir uno de los adyacentes (supongamos el i–ésimo), este fuera saturado (y por tanto inelegible para una cadena

de aumento). En este caso examinamos dicho vértice y asignamos la etiqueta (P, i) al vértice emparejado con i mediante el emparejamiento.

Este último etiquetado se realiza por si se da el caso de que no comenzamos con el emparejamiento vacío, en cuyo caso, podría haber vértices saturados en la bipartición de las raíces que no estuvieran examinados.

Una nueva posibilidad que se da en este caso es que al escoger un vértice etiquetado, este sea par, en cuyo caso, etiquetaremos debidamente todos sus adyacentes no etiquetados ya y continuamos el proceso, hasta que escojamos un etiquetado impar no saturado, en cuyo caso detectaríamos una cadena alternante, y, siguiendo la observación 4.2.1 modificaríamos el emparejamiento y borraríamos todas las etiquetas.

Si este caso no se llegara a dar, es claro que no hay cadenas alternantes para ese vértice raíz, por lo que pasamos a coger otro. En esta situación tenemos que tener cuidado también de que la nueva raíz no esté etiquetada (pues eso indica que es una raíz "fracasada").

Repetimos con este nuevo vértice toda la operación teniendo en cuenta que no debemos considerar adyacentes ya etiquetados por la raiz anterior, ya que ya están saturados.

Repetiremos este proceso mientras haya algún vértice en la bipartición escogida que esté no saturado y que no esté etiquetado.

Observación 4.2.5 (Voraz)**.** Podemos considerar a este algoritmo como un algoritmo voraz, requiriendo por tanto de una prueba de corrección. Obviamente la tiene, y se anima al lector a investigar sobre ello, no obstante, la omitiremos aquí al tener este capítulo mero carácter introductorio. ♢

Para terminar vamos a recopilar el todo el ladrillo anterior en forma de algoritmo.

Algoritmo 5 Algoritmo de emparejamiento máximo en un grafo bipartito.

Entrada: Emparejamiento inicial M (puede que vacío).
Salida: Emparejamiento de máximo cardinal.

```
1: repetir
2:   Escoger algún vértice no saturado y sin etiquetar.
3:   Etiquetar vértice elegido con (P, −)
4:   repetir
5:     Escoger un vértice de entre los etiquetados no examinados.
6:     si es impar saturado (supongamos vértice i) entonces
7:       Examinar vértice.
8:       Etiquetar vértice adyacente por M con (P, i).
9:     si no, si es par (supongamos vértice i) entonces
10:      Etiquetar los adyacentes sin etiquetar con (I, i)
11:    fin si
12:  hasta que no es impar no saturado ó no quedan vértices etiquetados no examinados
13:  si impar no saturado entonces
14:    Modificar emparejamiento.
15:    Borrar etiquetas.
16:  fin si
17: hasta que no existen más
18: devolver emparejamiento actual (es óptimo).
```

Llegados a este punto, ya estamos listos para entrar en el núcleo del capítulo, el problema de asignación.

4.3. Problema de asignación

En primer lugar formularemos con sumo cuidado el problema a resolver, tras lo cual daremos un algoritmo de resolución casi totalmente basado en el problema de emparejamiento máximo de un grafo bipartito.

4.3.1. Formulación del problema de asignación

Consideremos un ***grafo bipartito completo*** y ***valorado*** G, es decir, un grafo bipartito con n vértices en cada conjunto de la bipartición (a partir de ahora los llamaremos V_1 y V_2) tal que, dado un vértice de V_1, este está conectado con todos los vértices de V_2 (y viceversa).

Que el grafo sea "valorado" significa que cada arista del mismo tiene asociado un valor $c \in \mathbb{R}$, que en la práctica suele ser interpretado o bien como un "coste" o bien como un "beneficio".

Estas "valoraciones" de las aristas se suelen representar con una matriz $C \in \mathfrak{M}_n(\mathbb{R})$, donde c_{ij} representa el valor asociado con la arista que une el i–ésimo vértice de V_1 con el j–ésimo vértice de V_2. A la matriz C se la suele llamar ***matriz de valoraciones***.

Hechos los preámbulos, pasemos a formular el problema.

Problema 4.1 (Problema de asignación)**.** Dado un grafo bipartito completo valorado, queremos obtener el emparejamiento de cardinal n de mínimo coste.

Para nosotros una solución a este problema vendrá dado por una n^2–tupla $x^t := (x_{11}, x_{12}, \ldots, x_{nn})$ donde $x_{ij} \in \{0, 1\}$. El hecho de que $x_{ij} = 1$ representa que la arista que une el i–ésimo vértice de V_1 con el j–ésimo de V_2 está en el emparejamiento óptimo, en caso contrario, la arista no pertenece al emparejamiento óptimo.

De esta forma, queremos minimizar la función objetivo

$$\sum_{i=1}^{n} \left(\sum_{j=1}^{n} c_{ij} x_{ij} \right)$$

evidentemente, la función a minimizar es una función lineal, de hecho, si definimos $c^t := (c_{11}, c_{12}, \ldots, c_{nn})$ tenemos que, escrita de forma compacta, la función objetivo es símplemente $c^t x$.

Veamos ahora cómo son las restricciones del problema. Lo único que exigimos es que el emparejamiento óptimo sea un emparejamiento, es decir, que cada vértice tenga exactamente una arista incidiento sobre él. Esto podemos expresarlo mediante las siguientes restricciones lineales.

$$\sum_{j=1}^{n} x_{ij} = 1 \quad i \in \{1, \ldots, n\} \quad \sum_{i=1}^{n} x_{ij} = 1 \quad j \in \{1, \ldots, n\}$$

El primer conjunto de restricciones representa que todos los vértices de V_1 tienen una única arista incidente, mientras que el segundo conjunto de restricciones indica lo mismo pero para los vértices de V_2. Nótese que ambos conjuntos de restricciones son necesarios.

Si miramos las restricciones con especial cariño nos damos cuenta de que la matriz asociada a todas ellas se corresponde con la matriz de incidencia vértices–arcos de un grafo bipartito completo de orden n (es decir, con n vértices en cada V_i). Poniendo esto en detalle

$$A = \left(\begin{array}{c|c|c|c} \xi_1 & \xi_2 & \cdots & \xi_n \\ \hline I_n & I_n & \cdots & I_n \end{array} \right)$$

donde I_n es la matriz identidad de orden n y ξ_i es la matriz nula salvo en su i–ésima fila, la cual es una fila de unos (compruébese, inducción).

Observación 4.3.1 (Programación lineal)**.** Nótese que las restricciones asociadas al problema de asignación son rectricciones lineales, luego, evidentemente, nos encontramos ante un problema de programación lineal, con la salvedad de que las variables de decisión son binarias.

No obstante, como acabamos de ver, la matriz asociada a las restricciones es la matriz de incidencia vértices–aristas de un grafo bipartito, y, por tanto, totalmente unimodular, lo cual quiere decir que podemos formular el problema "relajado", es decir, permitiendo $x_{ij} \in \mathbb{R}$ con $x_{ij} \geq 0$ y todos sus puntos extremos serían enteros, de hecho, por cómo son las restricciones, binarios.

De esta forma se concluye que podemos usar el algoritmo del Símplex (nuestro viejo confiable) para resolver el problema de asignación. ♢

Dualización del problema de asignación

Sabiendo lo que sabemos, ahora vamos a dualizar el problema de asignación, para lo cual seguiremos casi los mismos pasos que para dualizar un problema en forma estándar.

Dada la formulación de un problema de asignación

$$\begin{array}{c} \text{mín } c^t x \\ \text{Sujeto a:} \quad Ax = b, \qquad x \geq 0 \end{array}$$

donde b es un vector formado enteramente por unos, vamos a reformularlo con el procedimiento habitual para que se encuentre en forma canónica de minimización

$$\begin{array}{c} \text{mín } c^t x \\ \text{Sujeto a:} \quad (A^t| - A^t)^t x \geq (b^t| - b^t)^t, \qquad x \geq 0 \end{array}$$

dualizando este problema obtenemos

$$\begin{array}{c} \text{máx}(b^t| - b^t)(w_1^t|w_2^t)^t \\ \text{Sujeto a:} \quad (A^t| - A^t)(w_1^t|w_2^t)^t \leq c, \qquad (w_1^t|w_2^t)^t \geq 0 \end{array}$$

reorganizando la expresión se tiene

$$\begin{array}{c} \text{máx } b^t w_1 - b^t w_2 \\ \text{Sujeto a:} \quad A^t w_1 - A^t w_2 \leq c, \qquad (w_1^t|w_2^t)^t \geq 0 \end{array}$$

la clave del asunto está en fijarse en cómo es la matriz A^t, y esta es de la forma (compruébese)

$$A^t = \left(\begin{array}{c|c} \xi_1^t & I_n \\ \hline \xi_2^t & I_n \\ \hline \cdots & \cdots \\ \hline \xi_n^t & I_n \end{array}\right) =: (\Xi|\mathcal{I})$$

reordenando la expresión queda

$$\begin{array}{c} \text{máx } b^t(w_{11}^t|w_{12}^t)^t - b^t(w_{21}^t|w_{22}^t)^t \\ \text{Sujeto a:} \quad (\Xi|\mathcal{I})(w_{11}^t|w_{12}^t)^t - (\Xi|\mathcal{I})(w_{21}^t|w_{22}^t)^t \leq c, \qquad (w_1^t|w_2^t)^t \geq 0 \end{array}$$

desarrollando los productos por bloques y despejando

$$\begin{array}{c} \text{máx } b^t(w_{11} - w_{21}) + b^t(w_{12} - w_{22}) \\ \text{Sujeto a:} \quad c - \Xi(w_{11} - w_{21}) - \mathcal{I}(w_{12} - w_{22}) \geq 0, \qquad (w_1^t|w_2^t)^t \geq 0 \end{array}$$

Llamando $v := w_{11} - w_{21}$ y $u := w_{12} - w_{22}$ tenemos que

$$\begin{array}{c} \text{máx } b^t u + b^t v \\ \text{Sujeto a:} \quad c - \Xi v - \mathcal{I} u \geq 0 \end{array}$$

Fijándonos en la forma que tienen los bloques podemos obtener una expresión para las restricciones del problema dual "fila por fila" (compruébese).

$$\begin{array}{c} \text{máx } \sum_{i=1}^n u_i + \sum_{i=1}^n v_i \\ \text{Sujeto a:} \quad c_{ij} - u_i - v_j \geq 0 \end{array}$$

Con esto, acabamos de obtener una expresión bastante manejable del problema dual de asignación, expresión que será vital para el dasarollo del algoritmo para resolver este tipo de problemas (uno más eficiente que el viejo Símplex).

4.3.2. Algoritmo para el problema de asignación

El algoritmo que presentamos a continuación, del que tampoco comprobaremos su corrección (se deja al lector investigar más acerca del tema, esto es únicamente una introducción) tiene una filosofía similar a la del algoritmo del símplex. Partiendo de una solución factible inicial del problema, irá "cambiándola ligeramente" de forma que cada vez se aproxima más a la solución óptima.

Al contrario de lo que ocurría con el algoritmo para el emparejamiento máximo, este no hay ninguna forma elemental de razonar (aunque sea informalmente) sobre su corrección, motivo por el cual lo presentamos directamente en forma de pseudocódigo, comentando después los aspectos más delicados.

Algoritmo 6 Algoritmo de emparejamiento máximo en un grafo bipartito.

Entrada: Solución factible incial del problema dual (u, v).
Salida: Asignación óptima (de mínimo coste).

repetir
  Calculamos la "matriz auxiliar" asociada a la solución dual actual (u, v).
  Dicha matriz está definida por $c'_{ij} := c_{ij} - u_i - v_j$
  Consideramos el grafo original eliminando todas las aristas que no cumplan $c'_{ij} = 0$.
  Hallar el emparejamiento de máximo cardinal de dicho grafo usando el algoritmo 5.
  si el cardinal del emparejamiento no es n **entonces**
    Guardar el emparejamiento y sus etiquetas asociadas.
    Cambiamos a una nueva solución factible dual, a esto se le suele llamar "cambio dual".
    Calculamos $\delta := \text{mín}\{c'_{ij} \ : \ i \in V_1 \text{ par }, j \in V_2 \text{ sin etiquetar}\}$
    La nueva solución es $u'_i := u_i + \delta$ para $i \in V_1$ par, $v'_j := v_j - \delta$ para $j \in V_2$ impar.
    Borramos todo lo referente al emparejamiento.
  fin si
hasta que el emparejamiento tiene cardinal n.
devolver emparejamiento actual (es óptimo).

El único aspecto que el algoritmo no deja del todo claro es cómo calcular la solución dual inicial, cuestión que pasamos a comentar en el siguiente apartado.

Solución dual inicial

Una forma de encontrar una solución dual inicial es la siguiente. Tomamos

$$u_i = \text{mín}\{c_{ij} \ : \ j = 1, \ldots, n\}$$

Y, una vez hecho esto, calculamos

$$v_j = \text{mín}\{c_{ij} - v_i \ : \ i = 1 \ldots, n\}$$

Por definición del problema dual esto es, evidentemente, una solución factible dual (compruébese).

Todo este desarrollo teórico puede hacerse exactamente igual para problemas de asignación de maximización. La estrategia de elección de la solución dual inicial variaría ligeramente.

Observación 4.3.2 (Método húngaro)**.** Existen diversas variantes para el algoritmo de asignación, entre ellas el llamado "método húngaro", que no veremos aquí. El motivo de su omisión es que, a pesar de ser aparentemente más sencillo, su coste computacional es mucho mayor, haciéndolo prácticamente inservible. ♢

Capítulo 5

Problemas enteros

En el capítulo anterior vimos un tipo de problema entero muy especial. Eran problemas enteros tales que su "relajación continua" ofrecía también soluciones óptimas enteras (gracias a la unimodularidad total). En este capítulo nos olvidaremos de todas esas situaciones maravillosas y nos pondremos manos a la obra (de un modo meramente introductorio) a desarrollar métodos generales para resolver problemas de programación entera.

5.1. Ramificación y poda

El primero de los métodos que veremos pertenece a la familia de algoritmos de "ramificación y poda" ó "branch & bound". Veamos como aplicarlo a este caso. En primer lugar planteamos el problema a resolver.

$$\begin{array}{rc} & \min c^t x \\ \text{Sujeto a:} & Ax = b, \quad x \geq 0, \quad x_j \in \mathbb{Z}, \quad j \in J \subset \{1, \ldots, n\} \end{array}$$

Es decir, un problema de programación lineal en forma estándar salvo por el hecho de que tenemos la restricción de que algunas de las variables de decisión deben tomar valores enteros.

Observación 5.1.1 (Filosofía)**.** La filosofía de los algoritmos basados en ramificación y poda consiste en ir explorando todas las soluciones posibles a un problema de manera estructurada (usualmente recorriendo un árbol de exploración), dejando de explorar determinada zona de la estructura cuando es claro que la solución óptima no se encuentra ahí. Asimismo, no se explora "al tuntún" sino que se elije la zona de la estructura que resulta "más prometedora". ♢

La idea general del algoritmo es que, dado un problema de programación entera, lo relajemos (omitamos las restricciones de "enteridad") y lo resolvamos. Obviamente, si la solución óptima es entera, hemos terminado, pero esto nunca pasa, salvo en casos muy concretos como los ya vistos en el capítulo anterior.

Lo que haremos pues, en caso de que la solución no será entera, será considerar dos problemas auxiliares (tal y como veremos en detalle en el algoritmo 7). Estos problemas auxiliares los iremos almacenando en una especie de "bolsa" de problemas, de la cual los iremos sacando para ir resolviendo los más "prometedores".

Para esto, se debe establecer un orden entre problemas que nos permita decidir cuál es el mejor de todos. Esto lo haremos asignando a cada problema un número al que llamaremos "prioridad", que no será otra cosa que el valor de la solución óptima de dicho problema.

Es claro que la solución óptima de un problema relajado, será una cota inferior de la solución óptima de su problema entero asociado. Por este motivo, si en nuestro proceso de búsqueda encontramos una solución entera podremos descartar todos los problemas de la bolsa cuya prioridad sea peor que la solución entera encontrada.

Vista la filosofía abstracta, pasemos a ver el esquema concreto que seguiremos.

Algoritmo 7 Esquema general de ramificación y poda.

Entrada: Problema original.
Salida: Solución óptima del problema.

```
Metemos el problema original en una "bolsa de problemas".
mientras la bolsa no esté vacía hacer
    Tomar el problema de la bolsa con menor prioridad.
    si la prioridad no mejora la mejor solución encontrada entonces
        Sacar el problema de la bolsa.
    si no
        Relajar el problema y resolverlo.
        si es factible entonces
            Tomar su solución óptima como prioridad.
            si la prioridad no mejora a la mejor solución entera encontrada entonces
                Quitar el problema de la bolsa.
            si no, si la solución no es entera entonces
                Elegimos alguna de las componentes no enteras de la solución (por ejemplo x_j) y
                consideramos los siguientes problemas.
                Mismo problema con la restricción x_j ≤ [x_j]
                Mismo problema con la restricción x_j ≥ [x_j] + 1
                Sacamos al problema original de la bolsa y metemos a sus dos problemas derivados,
                ambos con la misma prioridad que su "problema padre".
            si no, si la solución es entera entonces
                Será la mejor solución entera hasta ahora.
                Quitamos el problema de la bolsa.
            fin si
        si no
            Sacar el problema de la bolsa.
        fin si
    fin si
fin mientras
si no encontramos ninguna solución entonces
    devolver Infactible.
si no
    devolver Mejor solución encontrada.
fin si
```

Aunque pueda parecer un algoritmo largo y complicado, realmente es símple y llanamente una aplicación directa del esquema general de ramificación y poda, por lo que, en el fondo, es más simple que el asa de un cubo.

Observación 5.1.2 (Complejidad)**.** Una de las cosas bonitas que tiene la programación lineal es que, como acabamos de ver, también sirve para resolver problemas de programación entera (aunque sea aplicándola varias veces).

En el caso particular de los algoritmos de ramificación y poda podemos llegar a aplicar el algoritmo del Símplex o del Símplex dual un número absurdamente grande de veces, lo cual sugiere que necesitamos una alternativa. ♢

5.2. Hiperplanos de corte

Esta sección no pretende ser más que una brevísima introducción al método de los hiperplanos de corte para resolver problemas de programación entera. No se debe tomar más que como un recetario útil.

Observación 5.2.1 (Filosofía)**.** La filosofía de los métodos de hiperplanos de corte es la siguiente. Dada la región factible del problema entero relajado, y su solución óptima x^*, si x^* no es solución

entera, entonces, mediante diversos métodos, pegar un tajo a la región factible de forma que eliminemos dicho punto extremo pero no nos carguemos ninguna solución entera.

De esta forma generamos otro problema de programación lineal, mientras la solución óptima de este siga siendo no entera, volveremos a meter otro corte a la región factible. ♢

No obstante resulta que el problema de "pegar un corte" a la región factible de forma que no nos carguemos ninguna solución entera no es trivial. Por fortuna para nosotros, matemáticos como Gomory o Chvátal le dedicaron unas cuantas tardes a este problema, de forma que disponemos de mecanismos explícitos para implementar estos algoritmos.

5.2.1. Corte puro de Gomory

5.2.2. Corte mixto de Gomory

Si no todas las variables de decisión deben ser enteras, tomamos

Parte III

Introducción a la programación no lineal

Capítulo 6

Problemas no lineales

Índice de términos