Programación Lineal

Universidad Complutense de Madrid

Métodos Algorítmicos de Resolución de Problemas

Doble Grado en Matemáticas e Ingeniería Informática

Javier Pellejero
Curso 2016-2017

Índice

1. Introducción

Cuando hablamos de programación lineal, estamos tratando el campo de la optimización que busca la maximización de o minimización de una función lineal de acuerdo a que las variables de la misma estén sujetas a unas restricciones interpretadas en forma de ecuaciones o inecuaciones también lineales.

En definitiva, nos son dadas una variables a las que que queremos asignar valores reales satisfaciendo las restricciones y optimizando (maximizando o minimizando) la función objetivo dada. Antes de continuar veamos algunas definiciones.

Definición 1.1. Sea un problema de programación lineal, distinguimos en este los siguientes elementos:

1) Definimos como **función objetivo** a la función $z = \sum_{j=1}^{n} c_j x_j$, donde cada x_j es una **variable de decisión** que debemos determinar y c_j son los **coeficientes de costo** asignados a cada variable, que son conocidos.

2) Denominamos **restricciones i-ésimas** a las ecuaciones e inecuaciones $\sum_{j=1}^{n} a_{ij} x_j \leq, \geq \text{o} = b_i$, $1 \leq i \leq m$ donde m es el número de restricciones, a_{ij} son los **coeficientes tecnológicos** que son conocidos, al igual que b_i. Además definimos las restricciones $x_j \geq 0$ como **restricciones de no negatividad**.
   Los coeficientes a_{ij} forman la denominada **matriz de restricciones** $A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix}$

3) Si un conjunto de variables $x_1, ..., x_n$ satisface todas y cada una de las restricciones del problema, entonces lo denominamos **punto factible**. Por último, al conjunto de todos estos puntos factibles los denominamos **región factible**.

2. Ejemplos introductorios

2.1. Problema de programación lineal de dos incógnitas (en $\mathbb{R}^2$)

Ejemplo 2.1. Veamos a continuación un ejemplo de problema de programación lineal.

Una empresa fabrica dos tipos de productos: A y B. Los productos A y B generan unos beneficios de 100 y 600 euros respectivamente. La empresa no puede fabricar más de 4 productos diarios y además sólo recibe material suficiente para fabricar a lo sumo 2 productos A y 3 productos B al día. Veamos cuál es el beneficio máximo de la empresa a diario.

Planteemos el problema. Tenemos:
Función objetivo:

$$\text{máx } z = 100x_1 + 600x_2$$

Restricciones:

$$x_1 \leq 2$$

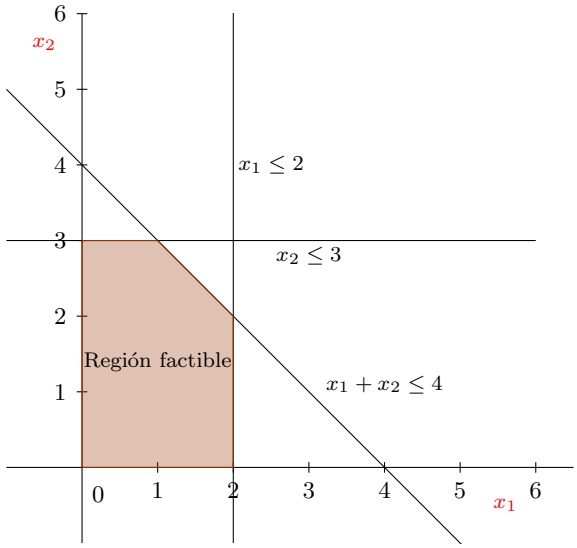
$$x_2 \leq 3$$

$$x_1 + x_2 \leq 4$$

$$x_1, x_2 \geq 0$$

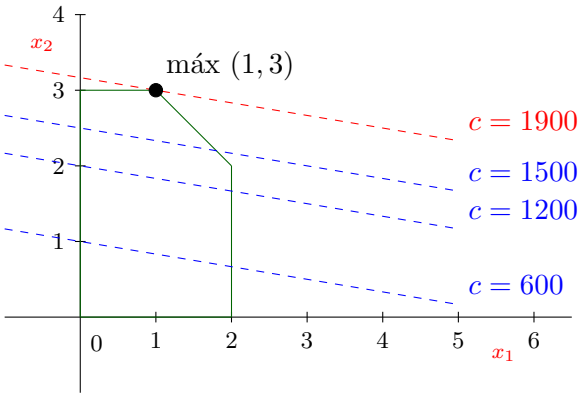
No es difícil darse cuenta que cada restricción define una inecuación lineal en el plano generado por $\mathbb{R}^2$ que a su vez designa una parte del plano que satisface cada inecuación. Luego la región factible de este problema, si la hay, será la intersección de las partes del plano designadas por cada inecuación, como podemos observar en la *figura 2.1.*

Figura 2.1.



Además, supongamos un valor c fijo tal que $z(x_1, x_2) = c \iff 100x_1 + 600x_2 = c \iff \iff x_2 = \dfrac{c - 100x_1}{600}$, luego para cada valor c que puede tomar la función z tenemos otra ecuación lineal que define una recta de nivel. En dicha recta, cada par de puntos x_1, x_2 de la misma genera un mismo valor c para nuestra función objetivo. Por lo tanto podemos trazar rectas paralelas para distintos valores de c tal y como se muestra en la *figura 2.2* para intuir cual es el punto donde encontramos la solución óptima (máximo) de nuestra función.

Figura 2.2.

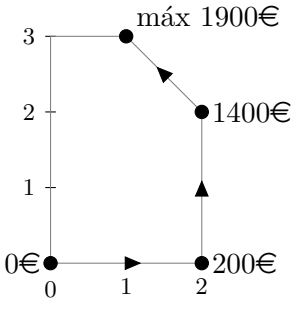


Luego nuestra solución óptima es producir un producto A y tres productos B.

Como hemos podido observar en el ejemplo anterior, la solución óptima ha sido alcanzada en un vértice del polígono que constituye la región factible. Esto se cumplirá, en general, siempre y cuando exista una solución, lo que denominamos como problema factible, y la región factible sea acotada. Esto es extendible a n variables, en $\mathbb{R}^3$ la solución óptima se correspondería con un vértice del poliedro que constituye la región factible y en $\mathbb{R}^n$ con el politopo n-dimensional correspondiente.

Y es en esto en lo que consiste el método *Símplex* que comentaremos más adelante, ideado por George Dantzig en 1947. Adelantaremos brevemente explicando que este algoritmo empieza en un vértice de la región factible, en el ejemplo anterior, por ejemplo el $(0, 0)$ y continua estudiando los vértices vecinos hasta encontrar uno cuyos vecinos no sean una solución mejor que él. Ese vértice se corresponde con la solución óptima. En la *figura 2.3* podemos ver un esquema iterativo de como actuaría el método *Símplex* en el ejemplo anterior.

Figura 2.3.



2.2. Problema de programación lineal de tres incógnitas (en $\mathbb{R}^3$)

Ejemplo 2.2. Veamos un segundo ejemplo. Imaginemos que la empresa del ejemplo anterior fabrica un nuevo producto C que les genera un beneficio de 1300€ por unidad vendida. Las restricciones anteriores se mantienen: el producto A se limita a 2 unidades diarias, el B a 3, de nuevo sólo podemos fabricar 4 productos diarios y además queremos añadir la restricción $x_2 + 3x_3 \leq 6$. Luego el planteamiento del problema sería el siguiente.
Función objetivo:

$$\text{máx } z = 100x_1 + 600x_2 + 1300x_3$$

Restricciones:

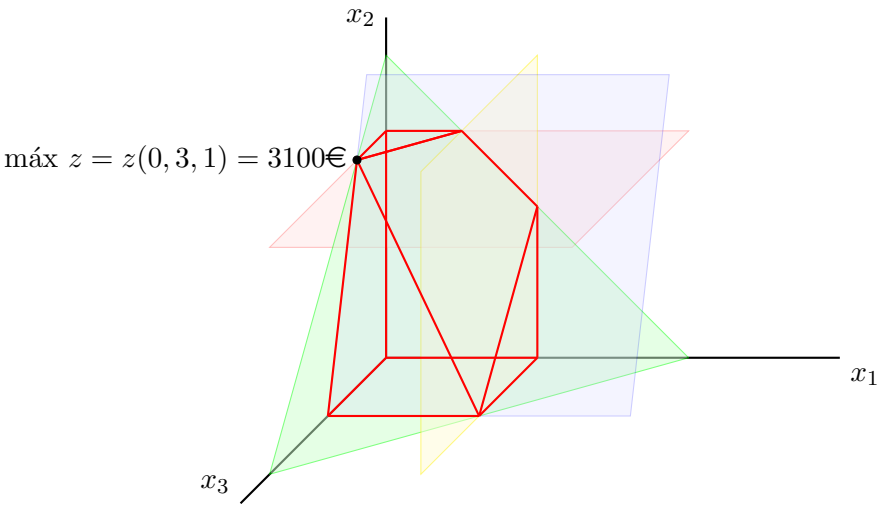
$$x_1 \leq 2$$
$$x_2 \leq 3$$
$$x_1 + x_2 + x_3 \leq 4$$
$$x_2 + 3x_3 \leq 6$$
$$x_1, x_2, x_3 \geq 0$$

Análogamente al ejercicio anterior, donde cada restricción representaba una recta, tenemos que ahora las restricciones representan un plano y la región factible generada es, como hemos comentado antes, un poliedro, como se muestra en la *figura 2.4*. Así, para n variables, cada restricción representa un hiperplano en $\mathbb{R}^n$ que a su vez generan un politopo n-dimensional correspondiente a la región factible.

De igual modo, podemos asignar un valor c de manera que el plano $z(x_1, x_2, x_3) = c$ se corresponde con los puntos (x_1, x_2, x_3) cuyo beneficio es c. De nuevo, podríamos trazar distintos planos paralelos para distintos valores de c para intuir cuál es la solución óptima.

Para finalizar, comentamos que para este problema tiene como solución óptima los valores $x_1 = 0, x_2 = 3, x_3 = 1$.

Figura 2.4.



3. Formulación de problemas como programas lineales

A parte de los ejemplos de la sección anterior que representan los problemas típicos de la programación lineal, podemos adaptar otro tipo de problemas para tratar de resolverlos mediante *método Símplex* o similares.

Algunos de estos problemas son problemas de grafos, como el *problema de flujo de mínimo coste en una red*; el *problema de la mochila* o el *problema de la dieta.*

3.1. Problema de la dieta

Definición 3.1. Definimos el problema de la dieta como uno del siguiente tipo:

Se conocen los contenidos nutritivos de ciertos alimentos, sus precios y la cantidad mínima diaria de nutrientes aconsejada. Se denota por m el número de nutrientes y por n el número de alimentos considerados. La cantidad mínima del nutriente i aconsejada es b_i, el precio de una unidad del alimento j es c_j y la cantidad de nutriente i en una unidad del alimento j es a_{ij}. El problema consiste en determinar la cantidad x_j, de cada alimento j, que debe comprarse de forma que se satisfagan los mínimos aconsejados y se alcance un precio total mínimo.

Ejemplo 3.1. Se considera un problema de la dieta con cinco alimentos y con los mínimos aconsejados para los nutrientes digeribles (DN), proteínas digeribles (DP), calcio (Ca) y fósforo (Ph), dados en la siguiente tabla.

Nutriente	Cantidad requerida	Maíz A	Avena	Maíz B	Salvado	Linaza
DN	74.2	78.6	70.1	80.1	67.2	77.0
DP	14.7	6.50	9.40	8.80	13.7	30.4
Ca	0.14	0.02	0.09	0.03	0.14	0.41
Ph	0.55	0.27	0.34	0.30	1.29	0.56

Los precios unitarios de los alimentos (en determinada unidad monetaria) son los siguientes:

Alimento	Maíz A	Avena	Maíz B	Salvado	Linaza
Coste	1	0.5	2	1.2	3

Procedamos a formular el problema, si nos fijamos, podemos lograr un problema del tipo de la *definición 1.1.*

Identifiquemos nuestros datos: Tenemos x_1 como cantidad de Maíz A, x_2 como avena, x_3 como maíz B, x_4 como salvado y x_5 como linaza. Nuestra matriz de coeficientes de costos es $C = \begin{pmatrix} 1 \\ 0{,}5 \\ 2 \\ 1{,}2 \\ 3 \end{pmatrix}$. Por último, nuestra matriz de restricciones es

$$\left(a_{ij}\right)_{\substack{1\leq j\leq 5\\ 1\leq i\leq 4}} = \begin{pmatrix} 78{,}6 & 70{,}1 & 80{,}1 & 67{,}2 & 77{,}0 \\ 6{,}50 & 9{,}40 & 8{,}80 & 13{,}7 & 30{,}4 \\ 0{,}02 & 0{,}09 & 0{,}03 & 0{,}14 & 0{,}41 \\ 0{,}27 & 0{,}34 & 0{,}30 & 1{,}29 & 0{,}56 \end{pmatrix} \text{ con } B = \begin{pmatrix} 74{,}2 \\ 14{,}7 \\ 0{,}14 \\ 0{,}55 \end{pmatrix}.$$

Es evidente que nuestro fin es reducir los costes de la compra de alimentos, luego nuestra función objetivo queda:

$$\text{mín } z = \sum_{j=1}^{5} c_j x_j = x_1 + 0{,}5x_2 + 2x_3 + 1{,}2x_4 + 3x_5$$

Mientras que las restricciones surgen al tener que cumplir un requisito de nutrientes:

$$AX \geq B \equiv \begin{array}{l} 78{,}6x_1 + 70{,}1x_2 + 80{,}1x_3 + 67{,}2x_4 + 77{,}0x_5 \geq 74{,}2 \\ 6{,}50x_1 + 9{,}40x_2 + 8{,}80x_3 + 13{,}7x_4 + 30{,}4x_5 \geq 14{,}7 \\ 0{,}02x_1 + 0{,}09x_2 + 0{,}03x_3 + 0{,}14x_4 + 0{,}41x_5 \geq 0{,}14 \\ 0{,}27x_1 + 0{,}34x_2 + 0{,}30x_3 + 1{,}29x_4 + 0{,}56x_5 \geq 0{,}55 \end{array}$$

$$x_1, x_2, x_3, x_4, x_5 \geq 0$$

Luego el problema es resoluble por el mismo método que lo eran los ejemplos de la sección anterior, mediante la búsqueda del vértice de menor coste de la región factible originada en $\mathbb{R}^5$, es decir, mediante el *método Símplex*.

La solución lineal de este problema es:

$$x_1 = 0, x_2 = \frac{2857}{1867}, x_3 = 0, x_4 = \frac{43}{1867}, x_5 = 0 \text{ con } z = \frac{12801}{18670}.$$

Cabe la pena destacar que el enunciado podría restringir a cantidades enteras la compra de alimentos, haciendo que nuestra solución fuera incorrecta. Este tipo de restricciones no son objeto de estudio de la programación lineal, sino que lo son de la programación entera que no trataremos en este texto.

3.2. Problema del transporte

Definición 3.2. Definimos el problema del transporte como uno del siguiente tipo:

Un determinado producto debe enviarse en cantidades especificadas $a_1, \ldots, a_m$ desde m centros de suministro y recibirse en cantidades especificadas $b_1, \ldots, b_n$, en n centros de demanda. El coste del envío de una unidad de producto desde el origen i al destino j es c_{ij}. El problema consiste en determinar las cantidades x_{ij}, que deben enviarse desde el origen i al destino j, para conseguir minimizar el coste del envío.

Ejemplo 3.2. En este caso, en lugar de ver un caso concreto, formulemos las condiciones anteriores.

Nuestra función objetivo es la suma de costes de las cantidades enviadas; es decir:

$$\text{mín } z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

Por lo que nuestra solución es un vector de la región factible del problema originada en $\mathbb{R}^{m \cdot n}$.

En cuanto a las restricciones, se limitan a cumplir las solicitudes de los centros de demanda, luego tenemos n restricciones del tipo:

$$\sum_{i=1}^{m} a_i x_{ij} \geq b_j \quad \forall 1 \leq j \leq n$$

Por supuesto, y como siempre, concluimos con que $x_{ij} \geq 0 \quad \forall 1 \leq i \leq m, \quad \forall 1 \leq j \leq n$.

3.3. Problema de flujo de mínimo coste en una red

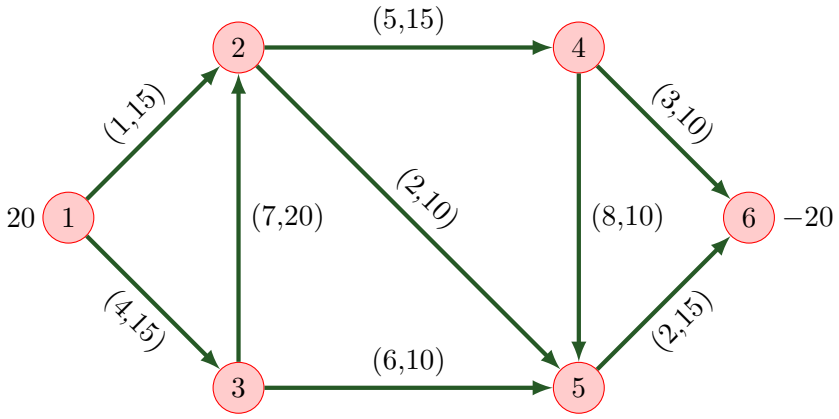
Definición 3.3. Definimos el problema del flujo mínimo en una red como uno del siguiente tipo:

Se considera una red de transporte (por ejemplo de transporte de locomoción, un sistema de comunicaciones, etcétera) a través de la cual se desea enviar un determinado producto desde ciertos puntos de la red, llamados nodos fuente, hasta otros puntos de destino, llamados sumideros. Además de estas dos clases de nodos, la red puede contener nodos intermedios, donde no se genera ni se consume el producto que está fluyendo por la red. Se denota por b_i, el suministro del nodo i (si i es un nodo fuente) y por $-b_i$, la demanda del nodo i (si i es un nodo sumidero). Se denota por x_{ij}, el flujo que se envía desde el nodo i al nodo j; u_{ij}, la capacidad máxima de flujo en la conexión entre el nodo i y el nodo j y por c_{ij}, el precio de enviar una unidad del producto desde el nodo i y al nodo j. El problema consiste en determinar las cantidades x_{ij}, que deben enviarse, para conseguir minimizar el coste total de los envíos.

Ejemplo 3.3. Veamos un ejemplo de problema flujo mínimo en una red:

Se considera el problema de hallar un flujo de mínimo coste en la red con capacidades de la *figura 3.1*, donde cada arco indica su coste y su capacidad (c_{ij}, u_{ij}). Se considera un único vértice fuente (el vértice 1, con suministro de 20 unidades) y un único vértice sumidero (el vértice 6, con demanda de 20 unidades).

Figura 3.1.



Planteemos nuestra función obejtivo. Recordemos que hemos definido x_{ij} como la cantidad llevada del nodo i al j y c_{ij} el coste por unidad transportada de i a j, luego nuestra función objetivo es:

$$\text{mín } z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$$

En este caso podemos expresar de forma más concreta la función anterior, pues en realidad sólo tenemos 9 aristas:

$$\text{mín } z = c_{12}x_{12} + c_{13}x_{13} + c_{24}x_{24} + c_{25}x_{25} + c_{32}x_{32} + c_{35}x_{35} + c_{45}x_{45} + c_{46}x_{46} + c_{56}x_{56} =$$

$$= x_{12} + 4x_{13} + 5x_{24} + 2x_{25} + 7x_{32} + 6x_{35} + 8x_{45} + 3x_{46} + 2x_{56}$$

Hablemos ahora de las restricciones. Si un vértice i es intermedio, la cantidad de producto y recibida es la misma, luego denotamos $b(i) = 0$. Es decir, se debe cumplir que $\sum_{j=1}^{6} x_{ij} - \sum_{j=1}^{6} x_{ji} = 0$. Y para los vértices fuente y sumidero (1 y 6 respectivamente) se tiene que cumplir que $\sum_{j=1}^{6} x_{ij} - \sum_{j=1}^{6} x_{ji} = 20$ y $\sum_{j=1}^{6} x_{ij} - \sum_{i=1}^{6} x_{ji} = -20$ respectivamente. Luego estas restricciones pueden ser representadas de forma matricial de la siguiente forma:

$$\underbrace{\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{pmatrix}}_{A} \begin{pmatrix} x_{12} \\ x_{13} \\ x_{24} \\ x_{25} \\ x_{32} \\ x_{35} \\ x_{45} \\ x_{46} \\ x_{56} \end{pmatrix} = \begin{pmatrix} 20 \\ 0 \\ 0 \\ 0 \\ 0 \\ -20 \end{pmatrix}$$

Donde la matriz A representa con cada una de sus 6 filas un vértice (la fila i-ésima representa el i-ésimo vértice) y cada una de sus 9 columnas representan una arista (en el orden que han sido escritas en la función objetivo). El valor 1 se corresponde con una arista que parte de ese vértice, y el -1 con una arista que llega al mismo (por lo que en cada columna hay un 1, un -1 y el resto ceros. Escribamos las ecuaciones resultantes de la igualdad matricial anterior:

$$\begin{aligned} x_{12} + x_{13} &= 20 \\ -x_{12} + x_{24} + x_{25} - x_{32} &= 0 \\ -x_{13} + x_{32} + x_{35} &= 0 \\ -x_{24} + x_{45} + x_{46} &= 0 \\ -x_{25} - x_{35} - x_{45} + x_{56} &= 0 \\ -x_{46} - x_{56} &= -20 \end{aligned}$$

Por último añadimos las últimas y no menos importantes:

$$0 \leq x_{ij} \leq u_{ij} \quad \text{equivalentemente :}$$

$$\begin{aligned} &0 \leq x_{12} \leq 15, \quad 0 \leq x_{13} \leq 15, \quad 0 \leq x_{24} \leq 15, \\ &0 \leq x_{25} \leq 10, \quad 0 \leq x_{32} \leq 20, \quad 0 \leq x_{10} \leq 15, \\ &0 \leq x_{45} \leq 10, \quad 0 \leq x_{46} \leq 10, \quad 0 \leq x_{56} \leq 15 \end{aligned}$$

Como podemos corroborar, hemos vuelto a conseguir un planteamiento lineal, al igual que en todos los ejemplos anteriores. Eso sí, esta vez hemos obtenido hasta 9 incógnitas y 15 restricciones, que señala que no siempre los problemas podrán ser resueltos cómodamente. Sin ir más lejos, región factible y solución de este problema se alojan en $\mathbb{R}^9$.

Por último señalar, como dato, que la solución del problema es $x_{12} = 15$, $x_{13} = 5$, $x_{24} = 5$, $x_{25} = 10$, $x_{32} = 0$, $x_{35} = 5$, $x_{45} = 0$, $x_{46} = 5$, $x_{56} = 15$ con $z = 155$.

3.4. Problema de la mochila

Definición 3.4. Definimos el problema de la mochila como uno del siguiente tipo:

Se dispone de un presupuesto b para invertir durante un determinado periodo de tiempo, y se consideran n proyectos. El proyecto j requiere una inversión a_j y proporciona un beneficio c_j . El objetivo es elegir un conjunto de proyectos de forma que no se exceda el presupuesto y se obtenga el máximo beneficio.

Ejemplo 3.4. Veamos un ejemplo de problema de la mochila:

Sean 4 proyectos: el primero, de inversión $a_1 = 2$ y beneficio $c_1 = 10$; el segundo, con $a_2 = 1$ y $c_2 = 7$; el tercero, con $a_3 = 6$ y $c_3 = 25$ y el cuarto, $a_4 = 5$ y $c_4 = 24$. Además tenemos $b = 7$ unidades para invertir. Calculemos el beneficio máximo sin exceder el presupuesto.

Podríamos formular el problema de la siguiente manera:

$$\text{máx } z = \sum_{i=1}^{4} c_i x_i = 10x_1 + 7x_2 + 25x_3 + 24x_4$$

como función objetivo y la restricción:

$$\sum_{i=1}^{4} a_i x_i \leq b \equiv 2x_1 + x_2 + 6x_3 + 5x_4 \leq 7$$

Donde x_i toma valor 0 si no escogemos el proyecto y 1 si es escogido; Es decir, $x_i \in \{0, 1\}\ \forall 1 \leq i \leq 4$. Esta restricción nos convierte el problema en uno de programación entera. Veamos cómo resolver este tipo de problemas.

Definamos el problema:

$$f_r(\lambda) = \text{máx } \sum_{i=1}^{r} c_i x_i$$

$$\text{Sujeto a la restriccón } \sum_{i=1}^{r} a_i x_i \leq \lambda$$

$$x_i \in \{0, 1\}\ \forall 1 \leq i \leq r \text{ y } \lambda \in \{1, ..., b\}$$

Es claro que $z = f_n(b)$. Definimos

$$\begin{aligned} f_1(\lambda) &= 0 && \text{si } 0 \leq \lambda < a_1 \\ f_1(\lambda) &= \text{máx}\{c_1, 0\} && \text{si } \lambda \geq a_1 \\ f_r(\lambda) &= \text{máx}\{f_{r-1}(\lambda), c_r + f_{r-1}(\lambda - a_r) && \text{en otro caso} \end{aligned}$$

Calculemos recursivamente f_2, f_3 y f_4 a partir de f_1 para todos los valores $\lambda = 1, ..., b$. Para identificar la solución óptima de $f_r(\lambda)$ utilizamos un indicador

$$p_r(\lambda) = \begin{cases} 0 \text{ si } f_r(\lambda) = f_{r-1}(\lambda) \\ 1 \text{ en caso contrario} \end{cases}$$

En la siguiente tabla se muestran los distintos valores de $f_r(\lambda)$ y $p_r(\lambda)$:

λ	f_1	f_2	f_3	f_4	p_1	p_2	p_3	p_4
0	0	0	0	0	0	0	0	0
1	0	7	7	7	0	1	0	0
2	10	10	10	10	1	0	0	0
3	10	17	17	17	1	1	0	0
4	10	17	17	17	1	1	0	0
5	10	17	17	24	1	1	0	1
6	10	17	25	31	1	1	1	1
7	10	17	32	34	1	1	1	1

La idea del algoritmo es comparar, a la hora de calcular $f_r(\lambda)$, si obtenemos un mejor resultado cogiendo o no x_r. Como x_r tiene una inversión a_r, si es escogido, nos queda $\lambda - a_r$ presupuesto, por lo que miramos en la tabla el valor $f_{r-1}(\lambda - a_r)$. Si cogerlo es conveniente tenemos un nuevo máximo y apuntamos un 1 en $p_r(\lambda)$. Sobra decir que si la inversión de x_r es mayor que λ no podremos cogerlo en ningún caso.

Es interesante recalcar que el uso de programación dinámica para guardar los datos de la anterior tabla es altamente recomendable, puesto que nos ahorraremos numerosos cálculos recursivos ya hechos, quedando de esta manera una complejidad de $\mathcal{O}(bn)$.

Por último, falta que las $p_r(\lambda)$ guardadas jueguen su papel. Las hemos guardado porque sabemos que máx $z = f_4(7) = 34$, pero desconocemos qué variables hemos tomado y cuáles no. Procedemos de la siguiente manera:

Empezamos por $p_4(7)$ que es 1, luego hemos tomado $x_4 = 1$ y esto implica que $f_4(7) = f_3(7 - a_4) + c_4 \overset{p_3(2)=0}{\Longrightarrow}$ no hemos tomado x_3 y $f_3(2) = f_2(2) \overset{p_2(2)=0}{\Longrightarrow}$ no hemos tomado x_2 y $f_2(2) = f_1(2) \overset{p_1(2)=1}{\Longrightarrow}$ hemos tomado x_1. Luego $x_1 = 1,\ x_2 = 0,\ x_3 = 0,\ x_4 = 1$ y $z = 34$.

4. Algoritmo del *Símplex*

No hemos hablado hasta ahora de complejidad salvo en el último ejemplo. Esto es porque, salvo el susodicho ejemplo (*Ejemplo 3.4*), hemos conseguido representar de forma correcta cada uno de los ejemplos vistos como programas lineales correspondientes a la *Definición 1.1*. El último ejemplo se escapaba de la linealidad y hemos tenido que ingeniárnoslas mediante la recursión para poder hallar la solución. Además la complejidad $\mathcal{O}(nb)$ es engañosa a causa del uso de programación dinámica para su resolución, puesto que b puede alcanzar valores cuantiosos, y realmente esconde un orden pseudopolinómico. En cambio, el resto van a poder ser resueltos mediante el *método Símplex*, del que ya hemos comentado brevemente y estudiaremos en esta sección y, aunque en caso peor es NP-completo, en la práctica, con n de tamaño prudente resulta un algoritmo eficiente.

4.1. ¿Cómo funciona el *Símplex*?

Definición 4.1. Decimos que un problema de programación lineal está en su **forma estándar** cuando todas las restricciones son del tipo:

$$\sum_{j=1}^{n} a_{ij}x_j = b_i,\ 1 \leq i \leq m \text{ donde } m \text{ es el número de restricciones.}$$

Esta definición se aleja de la *Definición 1.1*, pero podemos convertir restricciones conformadas por inecuaciones en igualdades del tipo anterior de la siguiente forma:

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i \implies \sum_{j=1}^{n} a_{ij}x_j + x_{n+i} = b_i$$

$$\sum_{j=1}^{n} a_{ij}x_j \geq b_i \implies \sum_{j=1}^{n} a_{ij}x_j - x_{n+i} = b_i$$

A las variables x_{n+i} las denominamos **variables de holgura o excedente, respectivamente, asociada a la restricción i-ésima**.

¿Por qué esta definición? Pues sencillamente porque es un prerrequisito del *método Símplex* que nuestro problema esté formulado en forma estándar.

Aún no estamos listos, sin embargo, para aplicar el método. Como ya comentamos, *Símplex* visita los vértices de la región factible buscando el óptimo, pero, ¿por cuál empieza? Pues este método empieza por el origen, es decir por $x = \overline{0} \in \mathbb{R}^n$. Y aquí surge el problema, qué pasa si dicho punto no pertenece a la región factible.

En los ejemplos vistos en la *Sección 2*, todas las restricciones eran del tipo "$\leq$". Esto, sumado a que las restricciones de no negatividad $x_j \geq 0$ nos aseguraba la existencia del origen en la región factible. Sin embargo, si existen restricciones del tipo "$\geq$" o "$=$" no nos asegura tal existencia. La solución a este problema es la introducción de una **variable artificial**, que denotaremos a_i, en cada i-ésima restricción de este tipo **después de ser convertida a forma estándar**. En la siguiente tabla podemos ver los pasos a seguir en función del tipo de restricción:

Formulación inicial	Forma estándar	Forma ampliada
$\sum_{j=1}^{n} a_{ij}x_j \leq b_i$	$\sum_{j=1}^{n} a_{ij}x_j + x_{n+i} = b_i$	$\sum_{j=1}^{n} a_{ij}x_j + x_{n+i} = b_i$
$\sum_{j=1}^{n} a_{ij}x_j \geq b_i$	$\sum_{j=1}^{n} a_{ij}x_j - x_{n+i} = b_i$	$\sum_{j=1}^{n} a_{ij}x_j - x_{n+i} + a_i = b_i$
$\sum_{j=1}^{n} a_{ij}x_j = b_i$	$\sum_{j=1}^{n} a_{ij}x_j = b_i$	$\sum_{j=1}^{n} a_{ij}x_j + a_i = b_i$

El objetivo de la inclusión de variables artificiales es la de llevar el problema a una dimensión de $\mathbb{R}$ superior donde si exista el origen en la región factible.

Denominamos comúnmente **forma ampliada** a las restricciones donde ya han sido incluidas variables de holgura (para "$\leq$") y variables artificiales (para "$\geq$" y "$=$"). Dichas variables de holgura y artificiales conformarán las **variables básicas**, o lo que es lo mismo, una solución básica factible (no óptima en general) que se centra en el origen.

4.2. Explicación e implementación del algoritmo

Si las variables básicas están compuestas sólo de variables de holgura, podemos proceder con el *Símplex*, en otro caso, debemos recurrir a otros métodos como el *método de las dos fases* o el *método Big M* (o *M grande*).

El primero consta de dos fases, la primera, que no detallaremos, consiste en aplicar el *Símplex* en busca de que las variables artificiales sean nulas donde la función objetivo es mín $z = \sum_{i=0}^{k} a_i$. Si esto ocurre podemos eliminar dichas variables y continuar con el *Símplex* habitual. *Big M* es un único proceso que combina ambas fases.

Procedamos con el *Símplex*. Si todas las restricciones eran del tipo "$\leq$" entonces no ha hecho falta el *método de las dos fases* y el origen es el vértice inicial. En caso contrario, hemos ejecutado la primera fase y nuestro algoritmo y tenemos un vértice P que es solución factible básica. La idea del algoritmo es desplazarse desde ese punto extremo o vértice a otro vecino a lo largo de una arista que los una, mejorando con ello la función objetivo. Continuaremos realizando este proceso hasta llegar a un vértice óptimo o que una arista nos lleve a ∞.

Sea B la matriz definida por las variables básicas de la forma aumentada, es decir, la base. Cada vez que nos desplacemos a un vértice vecino, una variable básica saldrá de B y por tanto dejará de formar parte de la solución (valdrá 0) y dejará entrar a otra. Las variables que no pertenecen a la base forman la matriz N. Estas matrices irán variando en cada iteración del algoritmo.

La variable que entra puede ser cualquiera de coste $c_i > 0$, pero nosotros cogeremos $c_e = \text{máx}\{c_i\}$. Si $c_e \leq 0$, hemos acabado. Para escoger la variable que sale tomamos $l \in B$ tal que $\Delta_l = \text{mín}\{\Delta_i : i \in B\}$ donde $\Delta_i = \begin{cases} \infty \text{ si } a_{ei} \geq 0 \\ \dfrac{b_i}{a_{ie}} \text{ si } a_{ie} > 0 \end{cases}$. Si el mínimo $\Delta_l = \infty$ el problema es no acotado. En caso contrario pivotamos, consistente en dividir toda la fila l entre a_{le} de forma que el nuevo a_{le} (denotamos $\hat{a}_{le}$) valga 1. Además queremos que el resto de valores de la columna e pasen a valer 0.

Conseguimos esto sumándole a la fila $i(\neq l)$ la fila $l \cdot (-a_{ie})$, luego $\hat{a}_{ij} = a_{ij} + \hat{a}_{lj}(-a_{ie})$

para todo j y para todo $i \neq l$. En particular, $\hat{a}_{ie} = a_{ie} + \underbrace{\hat{a}_{le}}_{1}(-a_{ie}) = 0$.

Además $\hat{b}_i = \begin{cases} b_l/a_l e & \text{si } i = l \\ b_i - a_{ie}\hat{b}_l & \text{si } i \neq l \end{cases}$, $\hat{c}_j = c_j - c_e\hat{a}_{lj}$ (En particular $\hat{c}_e = 0$) y $\hat{z} = z + c_e\hat{b}_e$. Por último, y no menos importante, la fila l pasa a ser la e intercambiando sus componentes a B y N respectivamente. Recordemos que los a_{ij} son los coeficientes tecnológicos, que conforman la matriz A, los c_j son los coeficientes de costo que conforman la matriz C, los b_i, a los que no hemos puesto nombre, conforman la matriz B y por último, z representa el valor de la función z en el vértice en el que nos encontramos.

A continuación presentamos el algoritmo en pseudocódigo.

```
/**********************
* simplex (A, b, c)
**********************/
(N, B, A, b, c, z) = inicializarSimplex(A, b, c)
Creamos Δ[m]
while ∃j ∈ N tq c_j > 0 do
    Elegimos e ∈ N tq c_e = máx{c_j : j ∈ N}
    for cada i ∈ B do
        if a_ie > 0 then
            Δ_i = b_i/a_ie
        else
            Δ_i = ∞
        end if
        Elegimos l ∈ B tq Δ_l = mín{Δ_j : j ∈ B}
        if Δ_l == ∞ then
            return "no acotado"
        else
            (N, B, A, b, c, v) = pivotar(N, B, A, b, c, z, l, e)
        end if
    end for
end while
for i = 0 to n do
    if i ∈ B then
        x̄_i = b_i
    else
        x̄_i = 0
    end if
end for
return (x̄_1, x̄_2, ..., x̄_n)
```

```
/*********************
* pivotar(N, B, A, b, c, z, l, e)
*********************/
// Calcula los coeficientes de la ecuación para la nueva variable básica x_e.
Creamos Â[m][n]
b̂_e = b_l / a_le
for cada j ∈ N − {e} do
    â_ej = a_l j / a_le
end for
â_el = 1 / a_le
//Calculamos los coeficientes de las demás constantes
for cada i ∈ B − {l} do
    b̂_i = b_i − a_ie b̂_e
    for cada j ∈ N − {e} do
        â_ij = a_ij − a_ie â_ej
    end for
    â_il = −a_ie â_el
end for
// Calculemos el nuevo valor de la función objetivo
ẑ = z + c_e b̂_e
for cada j ∈ N − {e} do
    ĉ_j = c_j − c_e â_ej
end for
ĉ_l = −c_e â_el
// Calculamos los nuevos conjuntos de variables básicas y no básicas.
N̂ = N − {e} ∪ {l}
B̂ = B − {l} ∪ {e}
return (N̂, B̂, Â, b̂, ĉ, ẑ)
```

Ejemplo 4.1. Veamos un ejemplo de problema de programación lineal resuelto mediante el *algoritmo Símplex*:

$$\text{máx } z = 3x_1 + x_2 + 3x_3$$

$$\text{sujeto a : } \begin{cases} 2x_1 + x_2 + x_3 \leq 2 \\ x_1 + 2x_2 + 3x_3 \leq 5 \\ 2x_1 + 2x_2 + x_3 \leq 6 \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \end{cases}$$

Consideramos el problema en la forma ampliada:

$$\text{máx } z = 3x_1 + x_2 + 3x_3$$

$$\text{sujeto a : } \begin{cases} 2x_1 + x_2 + x_3 \quad +x_4 \qquad\quad = 2 \\ x_1 + 2x_2 + 3x_3 \qquad +x_5 \quad = 5 \\ 2x_1 + 2x_2 + x_3 \qquad\quad +x_6 = 6 \\ x_1 \geq 0, x_2 \geq 0, x_3, x_4 \geq 0, x_5 \geq 0, x_6 \geq 0 \end{cases}$$

	x_1	x_2	x_3	x_4	x_5	x_6	
x_4	2	1	1	1	0	0	2
x_5	1	2	3	0	1	0	5
x_6	2	2	1	0	0	1	6
	3	1	3	0	0	0	$z = 0$

Elegimos $a_{41} = 2$ para pivotar. Pivotamos:

	x_1	x_2	x_3	x_4	x_5	x_6	
x_1	1	1/2	1/2	1/2	0	0	1
x_5	0	3/2	5/2	-1/2	1	0	4
x_6	0	1	0	-1	0	1	4
	0	-1/2	3/2	-3/2	0	0	$Z = 3$

Elegimos $a_{53} = 5/2$ para pivotar. Pivotamos:

	x_1	x_2	x_3	x_4	x_5	x_6	
x_1	1	1/5	0	3/5	-1/5	0	1/5
x_3	0	3/5	1	-1/5	2/5	0	8/5
x_6	0	1	0	-1	0	1	4
	0	-7/5	3/2	-3/2	0	0	$Z = 27/5$

Luego $x_1 = 1/5, x_2 = 0,\ x_3 = 8/5$ y $z = 27/5$

4.3. Complejidad

La complejidad del *algoritmo Símplex* es difícil de determinar, así que vamos a dar pinceladas sobre el coste esperado de ejecución.

Sea un problema en forma aumentada de n incógnitas (incluyendo variables de holgura y excedente) y m restricciones. Cabe esperar que $n \geq 2m$, debido a que hemos incluido entre una y 2 incógnitas por restricción. Si es así, podemos aproximar el número de puntos extremos (vértices) en $\dfrac{n!}{m!(n-m)!} \geq \left(\dfrac{n}{m}\right)^m \geq 2^m$. De acuerdo con esto podemos decir que en el caso peor *Símplex* es exponencial, sin olvidar que cada iteración tiene un alto coste de multiplicaciones y sumas también.

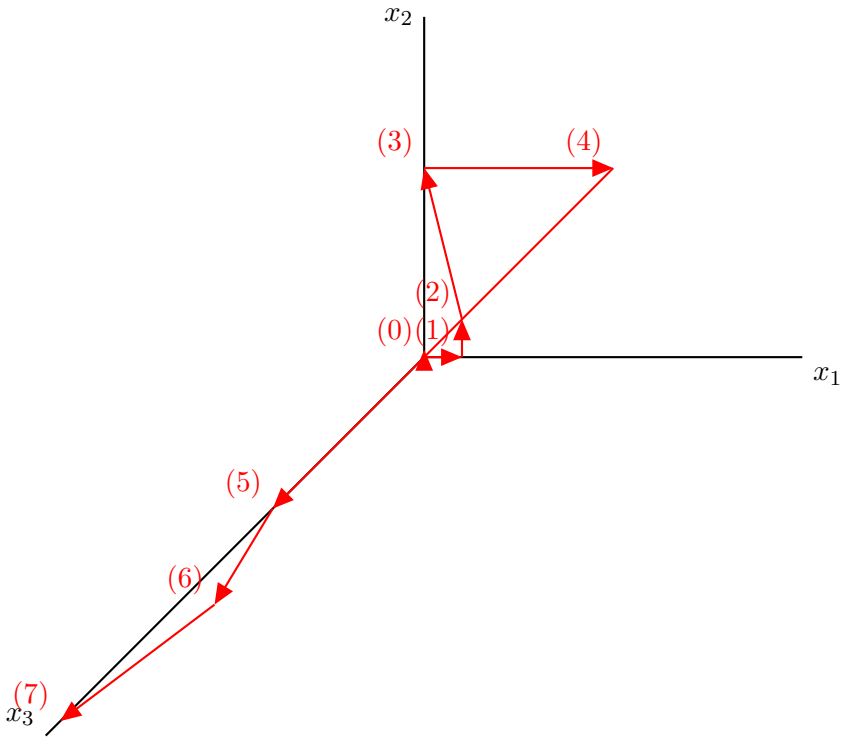
Ejemplo 4.2. Un problema clásico es el de **Klee** y **Minty**:

$$\text{máx } z = \sum_{i=n}^{n} 2^{n-i} x_i$$

Sujeto a n restricciones:
$x_1 \leq 5$
$4x_1 + x_2 \leq 25$
$8x_1 + 4x_2 + x_3 \leq 125$
...
$2^n x_1 + 2^{n-1} x_{n-1} + ... + 4x_{n-1} + x_n \leq 5^n$

Este problema tiene 2^n puntos extremos y, empezando por el origen los recorre todos los vértices. En la *figura 4.1* se muestra el caso $n = 3$.

Figura 4.1.



Observación.

- Sin embargo, el estudio del *Símplex* desde su invención en 1947 permite certificar que los peores casos, como el ejemplo de Klee y Minty, rara vez ocurren en la realidad.
- El número de iteraciones esperado para problemas de tamaño moderado es polinómico (*Spielman & Teng* (2001), *Kelner & Spielman* (2006) y *Dasgupta, Leiserson, Rivest & Stein* (2011)).
- En resumen, se estima que el coste promedio esperado es $\mathcal{O}(km^2n)$ donde k es una constante y mn es el coste de cada iteración (correspondiente al tamaño de A).

5. Bibliografía

1. Cormen, T., Leiserson, C., Rivest, R. and Stein, C. (2009). Introduction to algorithms. 3rd ed. Cambridge, Massachusetts: The MIT Press.

2. Dasgupta, S., Papadimitriou, C. and Vazirani, U. (2011). Algorithms. 1st ed. Boston: McGraw-Hill Higher Education.

3. Notas de la asignatura *investigación operativa* impartida por *doña María Inés Sobrón Fernández* en la *Facultad de Matemáticas* de la *Universidad Complutense de Madrid* durante el curso 2016-2017.