Grado en Matemáticas

TRABAJO FIN DE ESTUDIOS

Introducción a la Programación por Metas

Ana García Gómez

Sevilla, Junio de 2021

Índice general

Prólogo

Me gustaría agradecer a mi familia su apoyo incondicional durante estos cuatro años, lo cual ha sido primordial para que hoy pueda entregar este trabajo.

En especial, dar las gracias a mis padres por haberme dado la oportunidad con mucho esfuerzo y trabajo de estudiar en Sevilla y por haberme enseñado que con sacrificio y sobre todo con muchas ganas puedo conseguir todo lo que me proponga.

También dar las gracias a mis hermanos por la paciencia infinita que tienen conmigo, por calmar mis agobios y no dejar que me rinda.

No puedo olvidarme del Colegio Mayor Hernando Colón, ya que aquí he pasado estos cuatro años y me he cruzado con muchos compañeros que siempre han estado dispuestos a ayudar.

Por último, agradecer a mi tutor, Pedro Luis Luque Calvo, su dedicación, entrega y apoyo para realizar este trabajo.

Resumen

Comenzaré el trabajo haciendo un resumen de los resultados, definiciones y conceptos más importantes de la **Programación Lineal**; así como las circunstancias históricas que llevaron a los inicios de esta.

Dentro de la Programación Lineal, haremos un recorrido por los aspectos históricos más importantes de la **Programación Multiobjetivo**, exponiendo la forma de un problema de programación de este tipo, así como distintas técnicas para la resolución de estos problemas, destacando entre ellas el título de este trabajo; la **Programación por Metas**.

Adentrándonos en la Programación por Metas veremos la forma general de un problema de este tipo, así como las modificaciones más importantes (Programación por Metas Ponderadas, Programación por Metas minimax y Programación por Metas Lexicográficas); las cuales nos permitirán resolver problemas desde diferentes puntos de vista.

Dedicaremos una sección a resolver los problemas propuestos en el trabajo con la ayuda del software R y AMPL y presentaremos distintas conclusiones y reflexiones de los resultados obtenidos.

Por último; veremos tres aplicaciones reales de la Programación por Metas en la actualidad, para así conocer el alcance del tema tratado.

Abstract

Firstly, I will start with a summary of the key results, concepts and definitions of the **Linear Programming** as well as the historical circumstances that made it happen.

Within the Linear Programming, we will go through the most important historical aspects of the **Multi-objective Programming**, by presenting the pattern of this type of problems as well as different resolution techniques, highlighting the topic of this document: **Goal Programming**.

Diving into the Goal Programming, we will see the generic pattern of this type of problems, as well as the most important variants (Weighted Goal Programming, Minimum Goal Programming and Lexicographic Goal Programming) that will allow us to solve problems in different ways.

Also, we will find a section focused on how to solve problems with the help of the R and AMPL software, and we will present a few conclusions and reflections on the results obtained.

Finally, we will see in detail the applications of the Goal Programming nowadays, so that we check the impact of this subject.

Índice de figuras

Índice de tablas

Capítulo 1

Introducción: Programación lineal

La información con la que se ha escrito este capítulo se ha recopilado principalmente de los libros: [16] y [17]. También he usado páginas webs como: [4], [3], además de los apuntes de la asignatura de Programación Matemática del profesor Justo Puerto.

1.1. Investigación

Cualquier proceso de investigación consiste en un ciclo continuo en el cual el científico, a partir de unas observaciones, formula unas hipótesis que decidirá rechazar o aceptar en función de los resultados de los experimentos que haya llevado a cabo.

En la investigación existen dos aspectos fundamentales los cuales podríamos designar como investigación pura e investigación aplicada.

Caracterizar los dos tipos de investigación no es una tarea fácil; para diferenciarlas diremos que la investigación pura no se suele preocupar de la posterior utilidad de los trabajos, sin embargo, la investigación aplicada se preocupa por la utilidad humana, social y económica de sus posibles resultados.

La metodología que a partir de los problemas planteados por los métodos científicos-técnicos da lugar a resultados cuantitativos, que sirven como base para tomar decisiones militares, sociales, económicas,. . . , es lo que se conoce como Investigación Operativa (IO). En definitiva, la IO se ocupa de aplicar los métodos de los investigadores científicos para mejorar muchos aspectos de nuestra vida. La IO utiliza herramientas y resultados de muchas otras áreas científicas.

1.2. Inicios de la Investigación operativa (IO)

Las ideas y metodologías de la IO se han ido plasmando a través de la historia de la ciencia, siendo sus pilares fundamentales la Matemática, la Economía y la Estadística.

- Desde el punto de vista matemático destacan los trabajos sobre modelos lineales de Jordan, Minkowski o Farkas (entre muchos otros), todos ellos de finales del siglo XIX.
- Si nos adentramos en la Economía destacan autores como: Quesnay (siglo XVIII) o Walras (siglo XIX), quienes plantearon los primeros modelos sobre programación lineal. Más tarde también destacaron autores como Neumann, Kantorovich o Dantzid.

- En cuanto a la Estadística los orígenes sobre la IO se encuentran en las investigaciones sobre fenómenos de espera de Erlang (siglo XX).

El término de IO empezó a utilizarse en la Segunda Guerra Mundial, cuando en las tres alas del ejército británico se formaron grupos de trabajos interdisciplinares dirigidos por científicos con el fin de resolver importantes problemas tácticos y estratégicos. Tal fue el éxito que al finalizar la guerra muchos de estos científicos pasaron a las áreas industriales, financieras, administrativas,...; donde continuaron aplicando y extendiendo los modelos que habían llevado a cabo en la guerra.

La IO abarca un área muy amplia lo cual dificulta dar una definición exacta de ella, centrándonos en sus objetivos diremos que la IO es la utilización de métodos cuantitativos para ayudar a analistas y decisores en el diseño, análisis y mejora de la ejecución de problemas reales dentro de cualquiera de las áreas financiera, científica, industrial,...; siendo todos ellos examinados dentro del marco sistemático del método científico.

Actualmente la Investigación Operativa incluye gran cantidad de ramas[1] como la Programación Lineal, Programación No Lineal, Programación Dinámica, Simulación, Teoría de Colas, Teoría de Inventarios, Teoría de Grafos, etc. Nos centraremos en la primera de estas.

Figura 1.1: Ramas IO

1.3. Programación lineal (PL): Inicios, aplicaciones y motivación

La programación lineal (PL) es una clase especial de modelos de programación matemática que se desarrolló a partir de la Segunda Guerra Mundial para resolver ciertos problemas de asignación de recursos entre distintas actividades. Su nombre no procede de la creación de programas de ordenador, sino de un término militar, programar, que

[1] Imagen obtenida de https://sites.google.com/site/mscingindustrialmape/investigacion-de-operaciones-sistemas

significa 'realizar planes o propuestas de tiempo para el entrenamiento, la logística o el despliegue de las unidades de combate'. Las aplicaciones posteriores a una amplia variedad de problemas han sido numerosas y esto ha llevado a que los modelos de optimización lineal constituyan una de las herramientas básicas más utilizadas de la IO.

Aunque parece ser que la programación lineal fue utilizada por G. Monge en 1776, se considera a L. V. Kantoróvich uno de sus creadores. La presentó en su libro Métodos matemáticos para la organización y la producción (1939) y la desarrolló en su trabajo sobre la transferencia de masas (1942). Kantoróvich recibió el premio Nobel de economía en 1975 por sus aportaciones al problema de la asignación óptima de recursos humanos.

El objetivo fundamental de la programación lineal es la búsqueda de un óptimo para un programa definido por una función lineal en varias variables y restringido por un conjunto de ecuaciones y/o inecuaciones lineales en las citadas variables.

En la siguiente imagen[2] mostraremos algunas de sus aplicaciones:

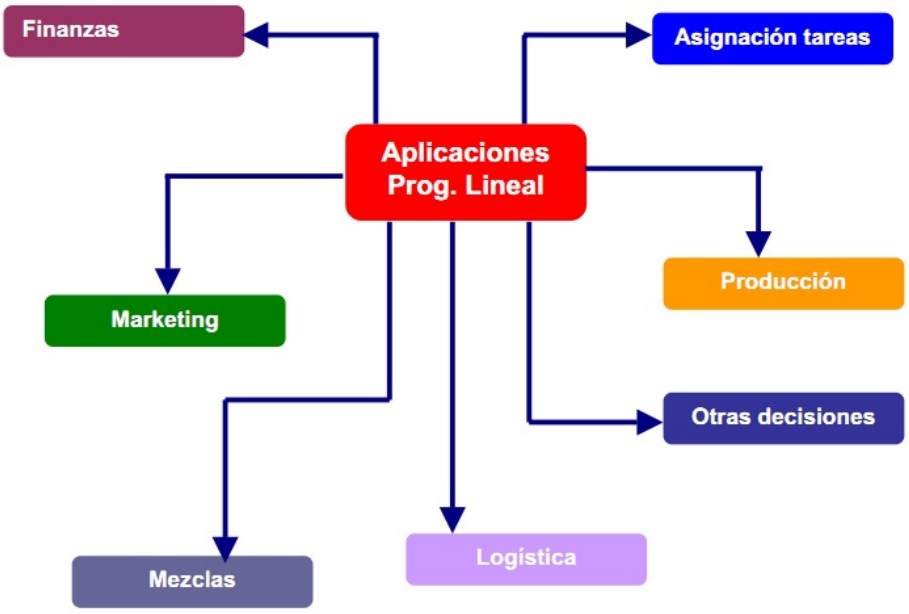

Figura 1.2: Aplicaciones PL

La investigación de operaciones en general y la programación lineal en particular recibieron un gran impulso gracias a los ordenadores. Uno de los momentos más importantes fue la aparición del método del simplex, desarrollado por G. B. Dantzig en 1947 y del cual hablaremos más adelante.

Nos centraremos en la PL porque un enfoque más realista y reciente de esta consiste en considerar varias funciones objetivos que en muchas situaciones podrían ser conflictivas, no siendo posible reducirlas a una sola función objetivo y, por tanto, teniendo que tratarlas de forma conjunta. Este enfoque más flexible es lo que se conoce como *programación multiobjetivo* dentro de la cual encontraremos el tema de este trabajo: **La Programación por Metas**.

[2]Imagen obtenida de https://inveoperaciones.wordpress.com/aplicaciones-de-la-programacion-lineal/

1.4. Conceptos fundamentales PL

En esta sección introduciremos conceptos básicos y necesarios para avanzar en este trabajo.

- **Problema de PL** : consiste en determinar $x = (x_1, ..., x_n) \in \mathbb{R}^n$ tal que:

$$\begin{gathered} max \text{ ó } min\ z = c_1x_1 + c_2x_2 + ... + c_nx_n \\ s.a.\ a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n(\leq, \geq \text{ ó } =)\ b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n(\leq\ , \geq \text{ ó } =)\ b_2 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n(\leq, \geq \text{ ó } =)\ b_n \\ x_j \geq 0, j = 1, .., n \end{gathered} \tag{1.1}$$

donde c_j, a_{ij}, b_j son constantes conocidas para todo i, j.

Diremos que un problema de programación lineal está en **forma estándar** si todas las restricciones son de igualdad y todas las constantes $b_j \geq 0$. Lo expresaremos en forma matricial como:

$$\begin{gathered} max \text{ ó } min\ z = c^T x \\ s.a.\ Ax = b \\ x \geq 0 \end{gathered}$$

donde A es la **matriz de coeficientes** de dimensiones $m \times n$, x es un vector de dimensión $n \times 1$, cuyas componentes se llaman **variables de decisión**, b es un vector de dimensión $m \times 1$ y c es un vector de dimensión $n \times 1$ denominado **vector de costes o beneficios**.

A los valores de $x_1, \ldots, x_n$ que verifican las restricciones los llamaremos **puntos factibles** y a la región que determinan **región factible**.

Veamos, que todo problema de programación lineal puede expresarse en forma estándar.

Proposición 1 Todo problema de programación lineal puede escribirse en forma estándar.

Demostración:

- Veamos como transformar la función objetivo:

$$max\ z = \sum_{j=1}^{n} c_j x_j \text{ es equivalente a } min\ z' = -\sum_{j=1}^{n} c_j x_j$$

- Veamos como transformar las restricciones:
  - $\sum_{j=1}^{n} a_{ij}x_j \leq b_i$ *se puede escribir como* $\sum_{j=1}^{n} a_{ij}x_j + s_i = b_i$

- $\sum_{j=1}^{n} a_{ij}x_j \geq b_i$ *se puede escribir como* $\sum_{j=1}^{n} a_{ij}x_j - t_i = b_i$
- Además si algún b_i es negativo; multiplicando por -1 la igualdad o desigualdad conseguiremos cambiar el signo.

A las variables $s_i, t_i \geq 0$ les llamaremos **variables de holgura y exceso** respectivamente.

- **Solución Básica**:

  Se dice que una solución básica es aquella que tiene al menos n-m componentes nulos o variables no básicas a las que llamamos x_N. Las m variables restantes se denominan variables básicas y las denominamos x_B (donde n es el número de variables y m numero de restriciones del problema de PL). Existen varios tipos de solución básica:

  - Solución básica factible (SBF). Todas las variables básicas son mayores o iguales que cero (condición de no negatividad).
  - Solución básica factible no degenerada. Todas las variables básicas son estrictamente positivas.
  - Solución básica factible degenerada. Alguna variable básica toma un valor nulo.

Observación: Puede demostrarse que cada SBF representa un vértice del conjunto factible. Sin embargo, un vértice puede ser representado por más de una SBF si la solución es degenerada.

1.5. Existencia de solución de los problemas de programación lineal

Antes de proponer métodos para la resolución de estos problemas veremos un resultado que nos garantize la existencia de solución. Primero vamos a introducir algunos conceptos que necesitaremos:

- **Definición conjunto convexo**: $S \subset \mathbb{R}^n$ es un conjunto convexo si verifica lo siguiente: $\forall x_1, x_2 \in S \; y \; \forall \lambda \in [0,1], \lambda x_1 + (1-\lambda)x_2 \in S$.
- **Definición punto extremo**: Un punto extremo de un conjunto convexo S, $x \in S$, es aquel que verifica que si lo podemos escribir de la forma $x = \lambda x_1 + (1-\lambda)x_2$ con $x_1, x_2 \in S$ y $\lambda \in (0,1)$ entonces necesariamente $x = x_1 = x_2$.
- **Definición dirección**: d es una dirección de S en x si $x + \lambda d \in S \; \forall \lambda \geq 0$.
- **Definición dirección extrema**: d es una dirección extrema si no puede expresarse como combinación lineal positiva de dos direcciones distintas de S; es decir, siempre que $d = \alpha_1 d_1 + \alpha_2 d_2, \; con \; \alpha_1, \alpha_2 > 0$ y d_1, d_2 direcciones de S, entonces d_1 es proporcional a d_2.
- **Definición separación conjuntos convexos**: Sean $S_1, S_2 \in \mathbb{R}^n$ convexos. Decimos que el hiperplano $p^t x = \alpha$ con $P \in \mathbb{R}^n$, $\alpha \in \mathbb{R}$ separa los conjuntos anteriores si $p^t x \leq \alpha \; \forall x \in S_1$ y $p^t x \geq \alpha \; \forall x \in S_2$.
- **Definición poliedro en forma estándar**: Sean $A \in \mathbb{R}^{mxn}$ (con $m \leq n$ y rg(A)=m) y $b \in \mathbb{R}^m$. El conjunto P=$\{x \in \mathbb{R}^n \; tq \; Ax = b, x \geq 0\}$ es un poliedro en forma estándar.

Una vez que conocemos las definiciones anteriores veamos algunos teoremas previos que necesitaremos para la demostración del Teorema Fundamental de la Programación Lineal.

Teorema 1.5.1 (Teorema de representación de puntos extremos). Sea P un poliedro en forma estándar. Entonces x es un punto extremo de P si y solo si existe una submatriz de A, $B \subset A$, $B \in \mathbb{R}^{mxm}$ de rango máximo tal que $B^{-1}b \geq 0$ y x=$\begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$.

Demostración:

($\Leftarrow$) Supongamos x=$\begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$ con B en las condiciones del teorema y veamos que entonces es un punto extremo.

Lo haremos por reducción al absurdo. Supongamos que existen $x^1, x^2 \in P$ distintos de x tales que $x = \lambda x^1 + (1-\lambda)x^2$ con $\lambda \in (0,1)$. Si escribimos esta igualdad por bloques obtenemos:

$$\begin{pmatrix} x_B \\ x_N \end{pmatrix} = \lambda \begin{pmatrix} x_B^1 \\ x_N^1 \end{pmatrix} + (1-\lambda)\begin{pmatrix} x_B^2 \\ x_N^2 \end{pmatrix}$$

Como x=$\begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$ gracias a lo anterior obtenemos que $0 = x_N = \lambda x_N^1 + (1-\lambda)x_N^2$ y como están en P entonces; $x_N^1, x_N^2 \geq 0$ y por tanto obtenemos que $x_N^1 = x_N^2 = 0$.

Por otro lado como $x^1, x^2 \in P$, entonces $Ax^1 = Ax^2 = b$. Escrito de forma matricial tendremos:

$$\begin{pmatrix} B & N \end{pmatrix}\begin{pmatrix} x_B^i \\ 0 \end{pmatrix} = b$$

con i=1,2; esto implica que $Bx_B^i + N0 = b$. Gracias a que B tiene rango máximo podremos despejar y obtener $x_B^i = B^{-1}b$; es decir; $x = x^1 = x^2 = \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$ lo cual nos lleva a una contradicción usando la definición de punto extremo.

Veamos ahora la otra implicación ($\Rightarrow$) :

Supongamos que x es punto extremo y veamos que x=$\begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$. Entonces, sin pérdida de generalidad, podemos suponer que $x = (x_1, ..., x_k, 0, ..., 0)$ con $k \leq m$ (por ser m = rg(A)), con $x_i > 0 \ \forall i = 1, ..., k$ (no tienen por qué ser las primeras coordenadas las no nulas, pero las hemos reordenado para que estén las primeras). Consideremos que las columnas de A asociadas a los valores positivos de x son $[a_1, ..., a_k]$. Probaremos por reducción al absurdo que estas columnas son linealmente independientes.

Supongamos que fuesen linealmente dependientes, entonces existirán $\lambda_1, \ldots, \lambda_k$ no todos nulos tales que $\sum_{i=1}^{k} \lambda_i a_i = 0$.

A continuación definimos $\lambda^t = [\lambda_1, \ldots, \lambda_k, 0, \ldots, 0] \in \mathbb{R}^n$ y construimos dos puntos distintos:

$$x^1 = x + \alpha\lambda$$
$$x^2 = x - \alpha\lambda$$

con α positivo y lo suficientemente pequeño como para que los dos puntos anteriores sean mayores o iguales que 0.

Por construcción tenemos que estos puntos tienen las primeras k componentes positivas y el resto nulas. Veamos que $x^i \in P$; pues tiene componentes no negativas y $Ax^i = Ax \pm \alpha A\lambda$

con $Ax = b$ porque $x \in P$ y con $A\lambda = \sum_{i=1}^{k} \lambda_i a_i = 0$; es decir; $Ax^i = b$. Pero entonces obtenemos

$$\frac{1}{2}x^1 + \frac{1}{2}x^2 = \frac{1}{2}x + \frac{\alpha}{2}\lambda + \frac{1}{2}x - \frac{\alpha}{2}\lambda = \frac{1}{2}x + \frac{1}{2}x = x$$

y con esto llegamos a una contradicción pues x era punto extremo. Por tanto; dichas columnas de A deben ser linealmente independientes.

Ampliamos las columnas hasta rango m $B = [a_1, \ldots, a_k, a_{k+1}, \ldots, a_m]$. Escribiendo $Ax = b$ por bloques:

$$(B\ N)\begin{pmatrix} x_1 \\ \vdots \\ x_k \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = b \Rightarrow B\begin{pmatrix} x_1 \\ \vdots \\ x_k \\ 0 \\ \vdots \\ 0 \end{pmatrix} + N\begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} = b$$

podemos concluir que hemos construido B con rango máximo por lo que tendrá inversa y podremos obtener:

$$\begin{pmatrix} x_1 \\ \vdots \\ x_k \\ 0 \\ \vdots \\ 0 \end{pmatrix} = B^{-1}b$$

Gracias a esto tenemos que x=$\begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$ como queríamos probar.

Teorema 1.5.2 (Teorema de representación de direcciones extremas). Sea P un poliedro en forma estándar. Entonces d es una dirección extrema de P si y solo si existe una submatriz de A, $B \subset A$, $B \in \mathbb{R}^{mxm}$ de rango máximo y una columna de A a_j que no esté en B tal que $B^{-1}a_j \leq 0$ y d=$\begin{pmatrix} -B^{-1}a_j \\ e_j \end{pmatrix}$ donde $e_j = (0, \ldots, 0, 1, 0, \ldots, 0)$ con el 1 en la posición j-ésima.

Demostración: Análoga a la del teorema anterior.

Lema 1.5. 1 Sea P un poliedro en forma estándar. Entonces d es una dirección de P si y solo si $d \geq 0$ y Ad = 0.

Demostración:

d dirección$\Leftrightarrow \forall x \in P, \forall \lambda \geq 0, x + \lambda d \in P \Leftrightarrow x + \lambda d \geq 0, \forall \lambda \geq 0 \Leftrightarrow d \geq 0$; pues si d tuviese una coordenada negativa existiría un λ adecuado como para que $x_i + \lambda d_i \leq 0$.

$$A(x + \lambda d) = b\ \forall \lambda \geq 0 \Leftrightarrow \underbrace{Ax}_{=b} + \lambda Ad = b\ \forall \lambda \geq 0 \Leftrightarrow \lambda Ad = 0\ \forall \lambda \geq 0 \Leftrightarrow Ad = 0$$

Teorema 1.5.3 (Minkowski-Weyl). Sea P en forma estándar y sean $x_1, \ldots, x_k$ sus puntos extremos y $d_1, \ldots, d_l$ sus direcciones extremas entonces el conjunto

$$S = \left\{ x \in \mathbb{R}^n : x = \sum_{i=1}^{k} \lambda_i x_i + \sum_{j=1}^{l} \mu_j d_j \; con \; \lambda_i \geq 0 \; \forall i, \; \sum \lambda_i = 1, \mu_j \geq 0 \; \forall j \right\}$$

verifica que P = S.

Demostración:

Lo probaremos por doble inclusión:

$(S \subset P)$ Sea $x \in S$, entonces $x = \sum_{i=1}^{k} \lambda_i x_i + \sum_{j=1}^{l} \mu_j d_j$ con λ_i, μ_j verificando las condiciones del teorema. Sabemos que $x \geq 0$ pues $x_i \in P$ implica que $x_i \geq 0$. Además, las direcciones d_j hemos probado en el lema anterior (Lema 1) que son no negativas. Como estamos sumando productos de elementos no negativos, la suma será no negativa. Teniendo en cuenta que $Ax_i = b$ pues $x_i \in P$ tendremos:

$$Ax = \sum_{i=1}^{k} \lambda_i A x_i + \sum_{j=1}^{l} \mu_j A d_j = \sum_{i=1}^{k} \lambda_i b + \sum_{j=1}^{l} \mu_j 0 = b \underbrace{\sum_{i=1}^{k} \lambda_i}_{=1} = b$$

Luego hemos probado que $x \in P$ pues Ax=b y $x \geq 0$

Veamos ahora la otra inclusión:

$(P \subset S)$ Lo probaremos por reducción al absurdo. Supongamos que existe un $z \in P$ y $z \notin S$. S es convexo y cerrado por definición, luego existe un hiperplano que separa a z de S; es decir; $\exists \, p \in \mathbb{R}^n, \; \alpha \in \mathbb{R} : \; p^t z > \alpha, \; p^t x \leq \alpha \; \forall x \in S$.

Por tanto tendremos:

- (1) $p^t z > \alpha \geq p^t x_i \; \forall i = 1, \ldots, k$.
- (2) $p^t d_j \leq 0 \; \forall j = 1, ..., l$. Esto lo probaremos por reducción al absurdo. Si no fuese cierto existiría un $\hat{j}$ tal que $p^t d_{\hat{j}} > 0$. Si cogemos $x(\mu) = x_1 + \mu d_{\hat{j}}, \mu > 0$; entonces $x(\mu) \in S, \; \forall \mu \geq 0$ y $\lim_{\mu \to \infty} p^t x(\mu) = \lim_{\mu \to \infty} (p^t x_1 + \mu \underbrace{p^t d_{\hat{j}}}_{\geq 0}) = +\infty > \alpha$!!!. Por tanto; tendremos que (2) es cierto.

Cogemos $\bar{x}$ el punto extremo tal que $p^t \bar{x} = max \; p^t x_i \; con \; i = 1, ..., k$ $(\star)$. Como $\bar{x}$ es un extremo; usando el teorema de representación de puntos extremos obtendremos que para alguna B cumpliendo las condiciones del enunciado del Teorema 1.5.1 $\bar{x} = \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$ con $B^{-1}b \geq 0$.

Consideramos que z tiene la forma $z = \begin{pmatrix} z_B \\ z_N \end{pmatrix}$. Como $z \in P$, Az=b; luego se tiene $Az = \begin{pmatrix} B & N \end{pmatrix} \begin{pmatrix} z_B \\ z_N \end{pmatrix} = b$; es decir; $Bz_B + Nz_N = b$ y como B tiene inversa pues hemos supuesto que es de rango máximo obtenemos: $z_B = B^{-1}b - B^{-1}Nz_N$.

Usando (1) tenemos que $p^t z > \alpha \geq p^t x_i \; \forall i$; particularizando para $\bar{x}$: $0 < p^t(z - \bar{x})$. Escribiendolo por bloques:

$$0 < p^t(z - \bar{x}) = \begin{pmatrix} p_B^t & p_N^t \end{pmatrix} \begin{pmatrix} z_B - \bar{x}_B \\ z_N - \bar{x}_N \end{pmatrix} = \begin{pmatrix} p_B^t & p_N^t \end{pmatrix} \begin{pmatrix} B^{-1}b - B^{-1}Nz_N - B^{-1}B \\ z_N - 0 \end{pmatrix} = -p_B^t B^{-1} N z_N + p_N^t z_N.$$

Luego hemos probado que $0 < (-p_B^t B^{-1}N + p_N^t)z_N$ y como $z > 0$ obtenemos que $p_N^t - p_B^t B^{-1}N > 0$.

- (3) $\exists\, j\ tq\ p_j - p_B^t B^{-1} a_j > 0$; donde a_j es la columna j-ésima de N.

Consideramos el vector $d_j = \begin{pmatrix} -B^{-1}a_j \\ e_j \end{pmatrix} := \begin{pmatrix} -y_j \\ e_j \end{pmatrix}$. Veamos que el vector y_j tiene alguna coordenada positiva. Lo probaremos por reducción al absurdo:

Si $y_j \leq 0\ \forall j$, entonces d_j sería una dirección extrema de P por el Teorema 1.5.2. Sea $p^t d_j = \begin{pmatrix} p_B^t & p_N^t \end{pmatrix}\begin{pmatrix} -B^{-1}a_j \\ e_j \end{pmatrix} = -p_B^t B^{-1} a_j + p_j > 0$ por (3). Entonces $x_1 + \mu d_j \in S\ \forall \mu > 0$; pero $p^t(x_1 + \mu d_j) = p^t x_1 + \mu \underbrace{p^t d_j}_{>0} \xrightarrow{\mu \to \infty} \infty$!!! pues $p^t x \leq \alpha\ \forall x \in S$.

Consideramos $\hat{x} = \bar{x} + \frac{\bar{b}_r}{y_{rj}} d_j$ con $\bar{b} = B^{-1}b$ y $\frac{\bar{b}_r}{y_{rj}} = min_i \left\{ \frac{\bar{b}_i}{y_{ij}}, y_{ij} > 0 \right\}$. Esto está bien definido porque acabamos de probar que para cada y_j tenemos una componente positiva. Queremos probar que al movernos de $\bar{x}$ a $\hat{x}$ en la dirección d_j no nos salimos de P. Vamos a demostrar que con la elección de $\frac{\bar{b}_r}{y_{rj}} d_j$ nos vamos a otro punto extremo (uno contiguo). Para ello tenemos que comprobar que $\hat{x} \in P$.

- **Condición 1**: $\hat{x} \geq 0$
  - $i \in B, i \neq r$:

    $$\hat{x}_i = \bar{x_i} + \frac{\bar{b}_r}{y_{rj}}(d_j)_i = \bar{x_i} + \frac{\bar{b}_r}{y_{rj}}(-y_{ij}) = \bar{b_i} + \frac{\bar{b}_r}{y_{rj}}(-y_{ij})$$

    Por tanto; tendremos que ver que $\bar{b_i} \geq \frac{\bar{b}_r}{y_{rj}}(y_{ij})$ como $\bar{b_i} = (B^{-1}b)_i \geq 0$ solo nos preocupa el caso en el que $y_{ij} > 0$ pues y_{rj} lo hemos tomado positivo y $\bar{b_i}$ lo es. Consideremos $y_{ij} > 0$; lo podremos pasar dividiendo y tendremos: $\frac{\bar{b_i}}{y_{ij}} \geq \frac{\bar{b}_r}{y_{rj}}$ lo cual es cierto $\forall\ y_{ij}$ pues hemos tomado $\frac{\bar{b}_r}{y_{rj}}$ como el mínimo de $\frac{\bar{b_i}}{y_{ij}}$.
  - $r \in B;\ \hat{x}_r = \bar{b_r} + \frac{\bar{b}_r}{y_{rj}}(-y_{rj}) = 0$
  - $l \in N, l \neq j : \hat{x}_l = 0 + \frac{\bar{b}_r}{y_{rj}} 0 = 0$
  - $j \in N, : \hat{x}_j = 0 + \frac{\bar{b}_r}{y_{rj}} 1 > 0$
- **Condición 2**: $A\hat{x} = B$:

$$A\hat{x} = A(\bar{x} + \frac{\bar{b}_r}{y_{rj}} d_j) = A\bar{x} + \frac{\bar{b}_r}{y_{rj}} A d_j = A\bar{x} + \frac{\bar{b}_r}{y_{rj}} \begin{pmatrix} B & N \end{pmatrix}\begin{pmatrix} -B^{-1}a_j \\ e_j \end{pmatrix} = b + \frac{\bar{b}_r}{y_{rj}}(-a_j + \underbrace{Ne_j}_{a_j}) = b$$

$\hat{x}$ está asociado como solución a $\hat{B} = B \setminus \{a_r\} \cup \{a_j\}$. Luego $\hat{x}$ es punto extremo de P. Ya hemos construido el punto extremo que va a contradecir $(\star)$.

$$p^t \hat{x} = p^t(\bar{x} + \frac{\bar{b}_r}{y_{rj}} d_j) = p^t \bar{x} + \frac{\bar{b}_r}{y_{rj}} p^t d_j = p^t \bar{x} + \underbrace{\frac{\bar{b}_r}{y_{rj}}}_{>0} \underbrace{(p_j - p_B^t B^{-1} a_j)}_{>0 por (3)} > p^t \bar{x}!!!$$

Teorema 1.5.4 (Teorema Fundamental de la P.L.). Sean P en forma estándar y $c \in \mathbb{R}^n$. La condición necesaria y suficiente para que exista mínimo es que $c^t d^j \geq 0 \ \forall \ d_j$ dirección extrema de P. En este caso, el mínimo $\bar{x}$ se alcanza en un punto extremo de P.

Demostración:

($\Leftarrow$) Supongamos que $c^t d^j \geq 0 \ \forall \ d_j$ y consideremos $x \in P$. Usaremos el teorema de **Minkowski-Weyl** el cual hemos visto antes (Teorema 1.5.3).

Obtenemos entonces que $x \in S$ y por tanto x será de la forma:

$$x = \sum_{i=1}^{k} \lambda_i x_i + \sum_{j=1}^{l} \mu_j d_j \ con \ \lambda_i \geq 0 \ \forall i, \ \sum \lambda_i = 1, \mu_j \geq 0 \ \forall j$$

usando lo anterior obtenemos:

$$c^{'} x = c^{'} \left(\sum_{i=1}^{k} \lambda_i x_i + \sum_{j=1}^{l} \mu_j d_j \right) = \sum_{i=1}^{k} \lambda_i c^t x^i + \underbrace{\sum_{k=1}^{l} \mu_j \underbrace{c_t d^j}_{\geq 0}}_{\geq 0}$$

gracias a la hipótesis y a que $\mu_j \geq 0 \ \forall j$.

Si seguimos trabajando con la igualdad anterior obtenemos:

$$c^{'} x = \sum_{i=1}^{k} \lambda_i c^t x^i + \underbrace{\sum_{k=1}^{l} \mu_j \underbrace{c_t d^j}_{\geq 0}}_{\geq 0} \geq \sum_{i=1}^{k} \lambda_i c^t x^i \geq c' \bar{x} \sum \lambda_i = c' \bar{x}$$

usando que $\bar{x}$ es el mínimo de los x y que $\sum \lambda_i = 1$.

Luego hemos partido de $c^t d^j \geq 0 \ \forall d_j$ y hemos llegado a que existe el mínimo y que se alcanza en un punto extremo.

Veamos ahora la otra implicación ($\Rightarrow$) :

Por negación del recíproco. Si existe j tal que $c^t d^j < 0$ entonces consideramos:

$$x(\mu) = x + \mu d^j, \mu \geq 0$$

.

Por tanto, tendremos :

$$c^{'} x(\mu) = c^{'} x + \mu \underbrace{c^{'} d^j}_{<0} \xrightarrow{\mu \to \infty} -\infty$$

por lo cual no existirá mínimo.

1.6. Construcción del modelo matemático de un problema de PL

En esta sección estudiaremos como "traducir" problemas de otras áreas científicas como la economía o la medicina, a problemas matemáticos con los que estamos acostumbrados a trabajar, para así obtener soluciones que ayuden en muchas otras áreas científicas.

La contrucción de un modelo matemático se puede iniciar respondiendo a las tres preguntas siguientes:

- ¿Qué busca determinar el modelo? Dicho de otra forma, ¿cuáles son las variables (incógnitas) del problema?
- ¿Qué restricciones deben imponerse a las variables a fin de satisfacer las limitaciones del sistema representado por el modelo?
- ¿Cúal es el objetivo qué necesita alcanzarse para determinar la solución óptima (mejor) de entre todos los valores *factibles* de las variables?

Veamos esto con un ejemplo.

1.6.1. Ejemplo de Reddy Mikks Company

Ejemplo de Reddy Mikks Company: Reddy Mikks Company posee una pequeña fábrica de pinturas que produce colorantes para interiores y exteriores de casas para su distribución de mayoreo. Se utilizan dos materiales básicos, A y B, para producir las pinturas. La disponibilidad máxima de A es de seis toneladas diarias; la de B es de ocho toneladas por día. Los requisitos diarios de materia prima por tonelada de pintura para interiores y exteriores se recogen en la siguiente tabla que expresa las toneladas de materia prima por toneladas de pintura; donde MPA es materia prima A y MPB materia prima B.

	EXTERIOR	INTERIOR	DISPONIBILIDAD MÁXIMA (toneladas)
MPA	1	2	6
MPB	2	1	8

En este caso la compañía busca determinar las cantidades de pintura para exteriores e interiores que se producirán para maximizar el ingreso bruto total, a la vez que se satisfacen las restricciones de la demanda y el uso de la materia prima. Por tanto tenemos:

- **Variables**:

$$x_E = \text{toneladas de pinturas para exteriores producidas diariamente}$$

$$x_I = \text{toneladas de pintura para interiores producidas diariamente}$$

- **Función objetivo**: Suponemos que cada tonelada de pintura para exteriores se vende en 3000 unidades monetarias, por tanto el ingreso bruto obtenido de la venta es de x_E toneladas es de $3x_E$ miles de unidades monetarias, de forma análoga supongamos que cada tonelada de pintura para interiores se vende en 2000 unidades

monetarias, por tanto el ingreso bruto obtenido de la venta de x_I toneladas es de $2x_I$ miles de unidades monetarias. Por tanto la funcion objetivo será:

$$z = 3x_E + 2x_I$$

- **Restricciones**: Tendremos restricciones sobre el uso de materias primas y sobre la demanda.
  - Sobre la materia prima: (uso de materias primas en ambas pinturas) $\leq$ (disponibilidad máxima de materias primas). Esto nos lleva a:

$$x_E + 2x_I \leq 6$$

$$2x_E + x_I \leq 8$$

  - En cuanto a la demanda supongamos que tenemos:

$$\text{(cantidad en exceso de pinturas para int sobre ext)} \leq 1\text{(tonelada por día)}$$

$$\text{(demanda de pintura para interiores)} \leq 2\text{(toneladas por día)}$$

    Esto se traduce en:

$$x_I - x_E \leq 1$$

$$x_I \leq 2$$

  - Restricciones de no negatividad:

$$x_E \geq 0$$

$$x_I \geq 0$$

Diremos que una solución es factible si satisface todas las restricciones del modelo. Resumiendo; tendremos el siguiente problema:

$$\begin{array}{c} Max\ z = 3x_E + 2x_I \\ s.a.\ x_E + 2x_I \leq 6 \\ 2x_E + x_I \leq 8 \\ x_I - x_E \leq 1 \\ x_I \leq 2 \\ x_E, x_I \geq 0 \end{array} \tag{1.2}$$

1.7. Solución gráfica de modelos de PL

El ejemplo (1.2) podremos resolverlo de forma gráfica porque solo tiene dos variables. Para modelos de tres o más variables la resolución gráfica resulta inviable y tendremos que recurrir a otras técnicas como el método del simplex; del cual hablaremos posteriormente.

- El primer paso es dibujar un sistema de coordenadas cartesianas en el que cada variable de decisión este representada por un eje.
- El segundo paso de este método consiste en graficar el espacio de soluciones factibles, es decir, el conjunto infinito de puntos que cumplen todas las restricciones; la cual en nuestro ejemplo será la región coloreada en rosa de la figura 1.3.

- Por último lugar determinamos los puntos extremos (vértices de la región factible), que sabemos por (Teorema 1) que son los candidatos a óptimos y evaluamos la función objetivo en estos puntos. Aquel o aquellos que maximicen o minimicen (según el problema) la función objetivo serán la solución que buscamos. En nuestro ejemplo se corresponde al punto C con:

$$x_E = \frac{10}{3}$$

$$x_I = \frac{4}{3}$$

Por tanto el valor de la función objetivo y la solución de nuestro problema será:

$$z = 3\frac{10}{3} + 2\frac{4}{3} = \frac{38}{3}$$

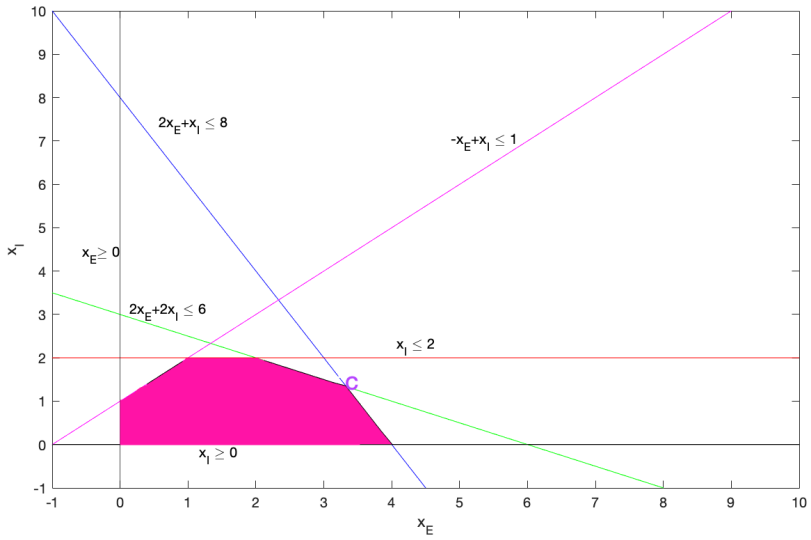

Figura 1.3: Ejemplo región factible

1.8. Método del simplex

Este método fue desarrollado por G. B. Dantzig en 1947 y consiste en la utilización de un algoritmo para optimizar el valor de la función objetivo teniendo en cuenta las restricciones del problema.

El método del simplex parte de uno de los vértices de la región factible, por ejemplo el vértice marcado con la estrella en la figura 1.4, y se basa en la propiedad: si la función objetivo no toma su valor máximo en dicho vértice, entonces existe una arista que parte del vértice y a lo largo de la cual la función objetivo aumenta y se llega a otro vértice.

El procedimiento es iterativo, pues mejora los resultados de la función objetivo en cada etapa hasta alcanzar la solución buscada. Como hemos visto anteriormente la solución se encuentra en un punto extremo, es decir, en un vértice[3] del que no parta ninguna arista a lo largo de la cual la función objetivo pueda aumentar.

[3]Imagen obtenida de https://www.plandemejora.com/metodo-simplex-paso-a-paso-ejemplos-maximizar-minimizar/

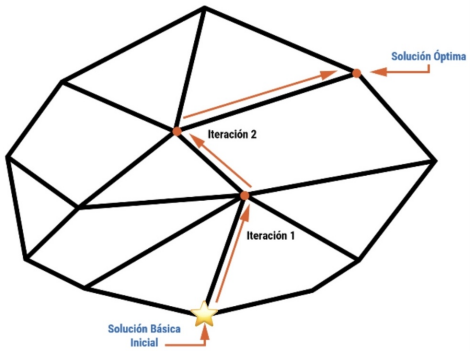

Figura 1.4: Idea gráfica método simplex

1.8.1. Tabla simplex

El método del simplex parte de un problema de P.L. en forma estándar o canónica que como ya sebemos puede expresarse como:

$$max\ z = c^T x$$

$$s.a.\ Ax = b$$

$$x \geq 0$$

Estudiaremos el método del simplex en forma tabular, el cual tiene la siguiente forma:

	$c_j \rightarrow$	c_i	$\dots$	c_n	
c_B	VB	x_1	$\dots$	x_n	x_B
c_{B1}	x_{B1}	y_{11}	$\dots$	y_{1n}	x_{B1}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
c_{Bm}	x_{Bm}	y_{mm}	$\dots$	y_{mn}	x_{Bm}
		$z_1 - c_1$	$\dots$	$z_n - c_n$	z

Tabla 1.2: Forma tabla simplex

donde:

- La columna VB contiene las variables básicas $x_B^T = (x_{B1}, ..., x_{Bm})$.
- La columna c_B está formada por los coeficientes de las variables básicas en la función objetivo.
- En la parte superior de la tabla encontramos todas las variables x_j del problema incluidas las variables de holgura y/o exceso. Sobre ellas están sus respectivos coeficientes c_j en la función objetivo.
- Los elementos interiores se corresponden con los vectores columnas y_j asociados a las variables x_j.
- En la columna de la derecha encontramos los valores de las variables básicas.

- En la parte inferior encontramos $z_j - c_j$; que los denominaremos **costes reducidos** también llamada fila indicador. Finalmente, a la derecha de esta última fila encontramos el valor z que toma la función objetivo en cada iteración. Esta tabla contiene toda la información necesaria para aplicar el método del Simplex. Además, forma un sistema de ecuaciones y, por tanto, se le puede aplicar una serie de transformaciones elementales, que no cambian su solución. Esto es:
  - Intercambiar dos filas.
  - Multiplicar una fila por un valor distinto de cero.
  - Sustituir la fila i por la fila i más r veces la fila j, siendo r un escalar arbitrario.

El método Simplex aplicado a problemas en forma de tabla se reduce a aplicarle a la tabla estas operaciones de forma que:

- La nueva tabla representa una nueva solución básica factible.
- Salvo en el caso de soluciones degeneradas, el valor de la función objetivo mejora en cada iteración. Cuando se aplica este mecanismo de resolución, a cada iteración se la denomina pivoteo. En cada pivoteo se lleva a cabo el paso de una SBF a otra.

El algoritmo consiste en lo siguiente :

- **Paso 0**. Construir la tabla.
- **Paso 1**. Si hay algún coste reducido negativo (posibilidad de mejora) ir a 3 y en caso contrario ir a 2.
- **Paso 2**. Si todos los costes reducidos son mayores o iguales a cero y no hay variables artificiales en la base con valor positivo estamos ante una solución óptima. Sin embargo, si los costes reducidos tienen valor mayor o igual que cero pero queda alguna variable artificial en la base con valor positivo nos encontramos ante un problema infactible.
- **Paso 3**. Si algún coste reducido tiene un valor negativo tendremos dos casos:
  - $y_j \leq 0$ entonces el problema es no acotado
  - En caso contrario existe mejora y debemos ir al paso 4.
- **Paso 4**. En este paso seleccionamos las variables de entrada y salida, la variable de entrada será aquella cuyo coste reducido sea el más negativo (si es x_k entonces k es la columna pivote). La variable de salida será aquella que haga mínima la razón, $\frac{x_{Bi}}{y_{ik}}$; con $y_{ik} \geq 0$, para cada fila, dicha fila se denominará fila pivote y al elemento y_{ik} lo llamaremos pivote.
- **Paso 5**. Calcular la nueva tabla:
  - Primero construimos la nueva tabla vacía sustituyendo la variable básica x_{Br} de salida por x_r y c_{Br} por c_r.
  - Después la fila r de la nueva tabla la obtenemos dividiendo la fila r de la anterior por el pivote y la columna k de la nueva tabla estará formada por 0 excepto en la posición (r,k) que encontraremos un uno.
  - A continuación, representaremos con las mismas letras pero con ˆ los elementos de la nueva tabla, que a excepción de los de la fila k y la columna r los obtendremos: $\hat{y_{ij}} = y_{ij} - p$, $\hat{x_{Bi}} = x_{Bi} - p$, $\hat{z_j} - \hat{c_j} = z_j - c_j - p$ y $\hat{z} = z - p$ con $p = \frac{(e_{rj} \times e_{ik})}{y_{rk}}$ $(i \neq r)$ donde e_{rj} es el elemento en la fila pivote de la columna j y e_{ij} es el elemento de la fila i en la columna pivote.

- Por último tenemos que volver a 1.

1.8.2. Ejemplo tabla simplex

Veremos un ejemplo: Dado el problema de programación lineal:

$$\begin{gathered} max\ z = 3x_1 + 2x_2 \\ s.a.\ 2x_1 + 3x_2 \leq\ 12 \\ 2x_1 + x_2 \leq\ 8 \\ x_1, x_2 \geq 0 \end{gathered} \tag{1.3}$$

Si lo pasamos a forma estándar tendremos:

$$\begin{gathered} max\ z = 3x_1 + 3x_2 + 0x_3 + 0x_4 \\ s.a.\ 2x_1 + 3x_2 + x_3 =\ 12 \\ 2x_1 + x_2 + x_4 =\ 8 \\ x_1, x_2, x_3, x_4 \geq 0 \end{gathered} \tag{1.4}$$

A continuación lo escribimos en forma tabular:

	$c_j \rightarrow$	3	2	0	0	
c_B	VB	x_1	x_2	x_3	x_4	x_B
0	x_3	2	3	1	0	12
0	x_4	2^*	1	0	1	8
		-3	-2	0	0	0

Tabla 1.3: Ejemplo simplex tabla 1

- **Paso 1**: Como hay valores $z_j - c_j \leq 0$ ($z_1 - c_1 = -3$ y $z_2 - c_2 = -2$), vamos al **Paso 3**.
- **Paso 3**: No hay vectores y_j asociados con los valores indicadores negativos ($y_1^t = (2, 2)$ e $y_2^t = (3, 1)$) con todos sus elementos menores o iguales que cero, por lo que es posible la mejora de la solución actual.
- **Paso 4**: El $z_j - c_j$ más negativo es $z_1 - c_1 = -3$, así que x_1 es la variable de entrada en la base con k=1 columna pivote. Determinamos las razones $\frac{x_{Bi}}{y_{i1}}$(para $y_{i1} > 0$), que son: $\frac{x_{B1}}{y_{11}} = \frac{12}{2} = 6$ y $\frac{x_{B2}}{y_{21}} = \frac{8}{2} = 4$ y la mínima corresponde a la segunda fila así que r=2 es la fila pivote con x_4 variables que sale de la base; por tanto el elemento $y_{21} = 2$ es el pivote, señalado en la tabla 1.3 con un asterisco.
- **Paso 5**:
  - a) Construimos una nueva tabla en la que la variable x_1 sustituye a la x_4 en la columna VB y lo mismo para sus respectivos coeficientes pasando de ser $c_{B2} = c_4 = 0$ a ser $c_{B2} = c_1 = 3$, tal y como veremos en la siguiente tabla.

- b) La fila r=2 de la nueva tabla se obtiene dividiendo la fila 2 de la tabla 1.3 por el pivote $y_{21} = 2$ y la columna 1 (k=1) de la nueva tabla está formada toda por ceros, excepto $\hat{y_{21}} = 1$, gracias a que dividiremos el pivote por si mismo.
- c) El resto de los elementos de la tabla los calculamos aplicando las reglas indicadas en el algoritmo, que conducen a los siguientes valores: $\hat{y_{12}} = 2 - \frac{1*2}{2} = 2$, $\hat{y_{13}} = 1 - \frac{0*2}{2} = 1$, $\hat{y_{14}} = 0 - \frac{1*2}{2} = -1$, $\hat{x_{B1}} = 12 - \frac{8*2}{2} = 4$, $\hat{z_2} - \hat{c_2} = -2 - \frac{1*(-3)}{2} = -\frac{1}{2}$, $\hat{z_3} - \hat{c_3} = -0 - \frac{0*(-3)}{2} = 0$, $\hat{z_4} - \hat{c_4} = -0 - \frac{1*(-3)}{2} = -\frac{3}{2}$, $\hat{z} = 0 - \frac{8*(-3)}{2} = 12$. Por tanto, tendremos la tabla 1.4.
- d) Volver al **Paso 1**.

	$c_j \rightarrow$	3	2	0	0	
c_B	VB	x_1	x_2	x_3	x_4	x_B
0	x_3	0	2^*	1	-1	4
3	x_1	1	$\frac{1}{2}$	0	$\frac{1}{2}$	4
		0	$-\frac{1}{2}$	0	$\frac{3}{2}$	12

Tabla 1.4: Ejemplo simplex tabla 2

- **Paso 1**: Como el valor indicador de la variable no básica x_2, $z_2 - c_2 = -\frac{1}{2}$, es negativo, vamos al **Paso 3**.
- **Paso 3**: Hay elementos positivos en el vector asociado $y_2^t = (2, \frac{1}{2})$, por lo que es posible la mejora de la solución actual e iremos al **Paso 4**.
- **Paso 4**: El $z_j - c_j$ más negativo (y único) es $z_2 - c_2$, así que x_2 es la variable de entrada en la base con k= 2 columna pivote. Determinamos las razones $\frac{x_{Bi}}{y_{i2}}$(para $y_{i2} > 0$), que son: $\frac{x_{B1}}{y_{12}} = \frac{4}{2} = 2$ y $\frac{x_{B2}}{y_{22}} = \frac{4}{\frac{1}{2}} = 8$ y la mínima corresponde a la primera fila así que r=1 es la fila pivote con x_3 variable que sale de la base. El elemento $y_{12} = 2$ es el pivote, señalado en la tabla 1.4 con un asterisco.
- **Paso 5**:
  - a) Construimos una nueva tabla en la que la variable x_2 sustituye a la x_3 en la columna VB y lo mismo para sus respectivos coeficientes siendo ahora $c_{B1} = 2$.
  - b) La fila r=1 de la nueva tabla se obtiene dividiendo la fila 1 de la tabla 1.4 por el pivote $y_{12} = 2$ y la columna 2 k=2 de la nueva tabla está formada toda por ceros, excepto $\hat{y_{12}} = 1$, gracias a que dividiremos el pivote por si mismo.
  - c) El resto de los elementos de la tabla los calculamos aplicando las reglas indicadas en el algoritmo igual que se hizo en el paso anterior, obteniendo así la tabla 1.5.
  - d) Volver al **Paso 1**.
- **Paso 1**: Como todos los valores indicadores son no negativos vamos al **Paso 2**.
- **Paso 2**: No hay variables artificiales en la base con valor positivo por lo que estamos ante una solución óptima. Como $x_B^T = (x_1, x_2) = (3, 2)$ la solución óptima es $x^{T*} = (x_1, x_2, x_3, x_4) = (3, 2, 0, 0)$ y el valor óptimo es z*=13.

	$c_j \rightarrow$	3	2	0	0	
c_B	VB	x_1	x_2	x_3	x_4	x_B
2	x_2	0	1	$\frac{1}{2}$	$-\frac{1}{2}$	2
3	x_1	1	0	$-\frac{1}{4}$	$\frac{3}{2}$	3
		0	0	$\frac{1}{4}$	$\frac{5}{2}$	13

Tabla 1.5: Ejemplo simplex tabla 3

1.8.3. Casos especiales del método y como localizarlos en la tabla del simplex

A lo largo del proceso iterativo es posible reconocer todos los casos que pueden darse en la resolución de un PL:

- **Solución única**: En la última tabla, los costes reducidos de las variables no básicas son estrictamente negativos (maximización) o estrictamente positivos (minimización).
- **Soluciones alternativas**: En la última tabla, alguno de los costes reducidos de las variables no básicas es igual a cero. Esto quiere decir que la variable no básica cuyo coste reducido es cero podría introducirse en la base sin perjudicar el valor de la función objetivo.
- **Solución no acotada**: Si al efectuar el test de salida de la base, todos los coeficientes de la columna correspondiente a la variable entrante son no positivos. Esto querría decir que la variable entrante puede aumentarse hasta el infinito sin que su crecimiento resulte bloqueado.
- **Problema infactible**: Se reconoce porque alguna variable de holgura/artificial queda en la base en la tabla final.
- **Solución degenerada**: Alguna variable básica vale cero.

1.9. Dualidad

Sea un problema de PL con la forma:

$$
\begin{aligned}
&min\ z = c_1x_1 + c_2x_2 + c_3x_3 \\
&s.a.\ a_{11}x_1 + a_{12}x_2 + a_{13}x_3\ \geq b_1 \\
&a_{21}x_1 + a_{22}x_2 + a_{23}x_3\ \leq b_2 \\
&a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \\
&x_1 \geq 0 \\
&x_2 \leq 0 \\
&x_3\ libre
\end{aligned}
\tag{1.5}
$$

a dicho problema lo denominaremos primal (P) y le asociaremos otro problema de programación lineal al que llamaremos dual (D) con la forma:

$$\begin{gathered} max\ z = u'b_1 + v'b_2 + w'b_3 \\ s.a.\ u'a_{11} + v'a_{21} + w'a_{31} \ \leq c_1 \\ u'a_{12} + v'a_{22}x_2 + w'a_{32} \ \geq c_2 \\ u'a_{13} + v'a_{23} + w'a_{33} = c_3 \\ u \geq 0 \\ v \leq 0 \\ w\ libre. \end{gathered} \tag{1.6}$$

La transformación que lleva (P) en (D) crea por cada restricción de (P) una variable de decisión en (D) y por cada variable de decisión en (P) una restricción en (D) de la forma que se indica en la siguiente tabla:

mín	máx
restricción $\geq$	*variable* ≥ 0
restricción $\leq$	*variable* ≤ 0
restricción $=$	*variable libre*
variable ≥ 0	*restricción* $\leq$
variable ≤ 0	*restricción* $\geq$
variable libre	*restricción* $=$

Observación: La transformación que se lleva a cabo depende de si estamos calculando el dual de un problema de minimización o de un problema de maximización. A cada problema de minimización le corresponde por tanto un problema de maximización y a cada problema de maximización un problema de minimización. Más aún, como puede verse con los dos problemas genéricos anteriores, siguiendo la tabla de la transformación, el problema dual del dual es el primal; es decir; existe una biyección entre ambos.

Capítulo 2

PROGRAMACIÓN POR METAS

Este capítulo lo he llevado acabo gracias a la información obtenida de los libros: [16] y [18]. Además he obtenido información de páginas web como: [5], [19], [20] y [11]

2.1. Introducción a la Programación multiobjetivo

Existen muchas situaciones en las que la modelización de sistemas reales nos lleva a construir modelos multiobjetivos (es decir, con varias funciones objetivos), los cuales pueden ser total o parcialmente conflictivos de forma que la mejora de algunos objetivos suele llevar al empeoramiento de otros objetivos.

Un problema multiobjetivo con n variables de decisión x_j, m restricciones y p objetivos consiste en:

Determinar los valores $x_1, \ldots, x_n$ tal que:

$$max\ z = (z_1(x), \ldots, z_p(x))$$
$$s.a\ x \in F$$

donde F$\subset \mathbb{R}^n$ región factible del espacio de soluciones $\mathbb{R}^n$ y $z(F) = Z \subset \mathbb{R}^p$ región factible del espacio de objetivos en $\mathbb{R}^p$. En muchos casos F puede expresarse como:

$$F = \{x \in \mathbb{R}^n : g_i(x) \leq 0, x_j \geq 0, \forall i, j.\}$$

donde las funciones g_i son las **restricciones** y a las funciones z_k las denominaremos **objetivos**.

Observación: Recordemos que por la Proposición 1 podremos convertir problemas de maximización en problemas de minimización y viceversa.

A diferencia de la programación lineal con un solo objetivo aquí no exisitirá el concepto de óptimo, ya que una solución que maximice un objetivo por lo general minimizará los otros. Por ello resulta imprescindible introducir el concepto de **solución eficiente**.

- **Definición**: De manera informal diremos que un punto x que sea factible será **eficiente**, si no existe otro punto factible x' que mejore el valor de un objetivo sin

causar un empeoramiento de los otros objetivos. De manera más rigurosa vamos a definir el conjunto de los puntos eficientes como:

$$\epsilon = \{x \in F : \nexists x' \in F \; tal \; que \; z_k(x') \geq z_k(x) \; \forall k$$

$$y \; z_t(x') > z_t(x) \; para \; al \; menos \; un \; t \in \{1, ..., p\}\}.$$

Este conjunto está formado por muchas soluciones y finalmente tendremos que elegir una como solución final, la cual recibirá el nombre de **solución de mejor compromiso**. Son muchos los métodos que nos ayudan a encontrarla, entre ellos destacan: método de las ponderaciones, método de ϵ-restricciones (los cuales son válidos tanto para problemas lineales como no lineales), el método del simplex multiobjetivo y la programación por metas, que consiste en una técnica multiobjetivo basada en el método del simplex que incorpora preferencias del decisor. Esta técnica será muy útil para encontrar directamente la solución de mejor compromiso.

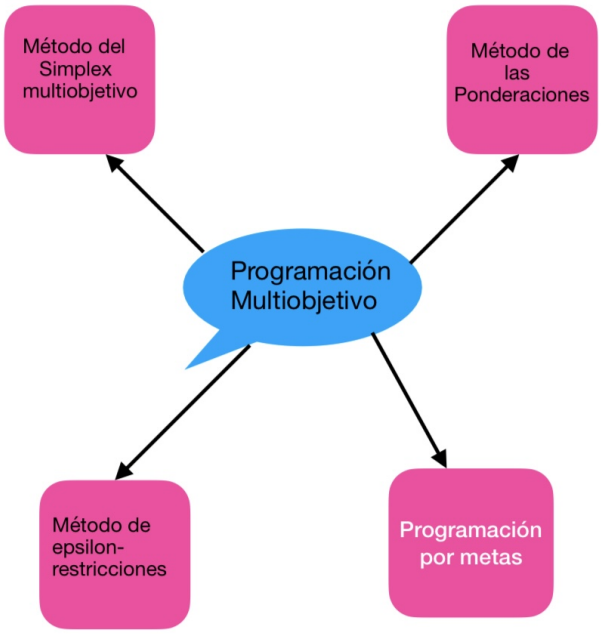

Figura 2.1: Métodos multiobjetivo

2.2. Programación por metas: inicios y desarrollo del modelo

La programación por metas tiene sus inicios en los trabajos de Charnes y Cooper a principios de los años sesenta como una aplicación de la programación lineal con múltiples objetivos, siendo esta una de las áreas con mayor número de aplicaciones en la programación multiobjetivo.

La idea del método consiste en proponer metas o niveles de aspiración para los distintos objetivos que se desean alcanzar. Una solución óptima se define como aquella que minimiza la desviación de las metas. El tomador de decisiones debe ser capaz de establecer al menos una importancia ordinal, para clasificar estas metas. Una ventaja importante de la programación por metas es su flexibilidad en el sentido de que permite al tomador de decisiones, experimentar con una multitud de variaciones de las restricciones y de prioridades de las metas cuando se involucra con un problema de decisión de objetivos múltiples. Lo formularemos como: Buscar $x \in \mathbb{R}^n$ tal que :

$$\begin{aligned} &min\ z = \textstyle\sum_{i=1}^{p} |z_i(x) - \hat{z}_i| \\ &s.a.\ x \in F \end{aligned} \tag{2.1}$$

donde $\hat{z}_i$ es la **meta** establecida por el decisor para el i-ésimo objetivo z_i y F es la región factible formada por restricciones lineales.

Por tanto, el objetivo es hacer mínima la suma de los valores absolutos de las desviaciones o las diferencias entre los objetivos y sus metas. Si observamos (2.1) vemos que la función objetivo no es lineal y por tanto no podremos aplicar el método del simplex. Sin embargo, seremos capaces de redefinir el problema y así conseguir una función objetivo lineal. Para ello, introduciremos dos nuevas variables:

- **Definición**: d_i^+ (variable de desviación por **exceso** de su meta)

$$d_i^+ = \frac{1}{2}(|z_i(x) - \hat{z}_i| + (z_i(x) - \hat{z}_i))$$

- **Definición**: d_i^- (variable de desviación por **defecto** de su meta)

$$d_i^- = \frac{1}{2}(|z_i(x) - \hat{z}_i| - (z_i(x) - \hat{z}_i))$$

Si intentamos interpretar lo anterior observamos que d_i^+ es igual a $(z_i(x) - \hat{z}_i))$ si $(z_i(x) \geq \hat{z}_i))$ y cero en otro caso; por el contrario d_i^- será igual a $(z_i(x) - \hat{z}_i))$ si $(z_i(x) \leq \hat{z}_i))$ y cero en otro caso. Es por esto que d_i^+ y d_i^- se interpretan como el **exceso** y el **defecto**, respectivamente, del nivel de aspiración o de la meta del i-ésimo objetivo. Las dos variables de desviación tomarán el valor cero cuando la meta alcance exactamente su nivel de aspiración.

- **Definición**: Una variable de desviación se dice que es **no deseada** cuando al decisor le conviene que la variable en cuestión alcance su valor más pequeño, es decir, cero.

Además para cada i tendremos:

$$\begin{aligned} d_i^+ + d_i^- &= |z_i(x) - \hat{z}_i|\ , d_i^+ \times d_i^- = 0 \\ d_i^+ - d_i^- &= z_i(x) - \hat{z}_i\ ,\ d_i^+, d_i^- \geq 0 \end{aligned}$$

Gracias a esta observación podemos transformar (2.1) en:

$$\begin{gathered} min\ z = \sum_{i=1}^{p}(d_i^+ + d_i^-) \\ s.a.\ x \in F \\ z_i(x) - d_i^+ + d_i^- = \hat{z}_i \\ d_i^+, d_i^- \geq 0,\ i = 1, \ldots p \end{gathered} \tag{2.2}$$

En el problema anterior podemos observar que la forma de un problema de programación por metas sigue siendo la misma que la de un modelo de programación lineal, es decir, también se tiene una función objetivo que se busca optimizar sujeta a una o más restricciones. Sin embargo, dentro de este marco de referencia se agregarán dos conceptos nuevos.

- **Definición**: denominaremos **restricciones de meta** a $((z_i(x) - d_i^+ + d_i^- = \hat{z}_i))$

Con la formulación (2.2) ya podremos aplicar el método del simplex para buscar una solución x de la región factible que haga mínima la suma de las variables de desviación.

Hemos omitido la restricción $d_i^+ \times d_i^- = 0$ porque convertiría (2.2) en un problema no lineal, pero realmente el método del simplex trabajará con ellas ya que no tiene sentido que haya simultáneamente un exceso y un defecto, por tanto, si $d_i^+ \geq 0$ entonces $d_i^- = 0$ y viceversa.

Resumiendo, se puede decir que los pasos para la formulación de problemas de Programación de Metas son:

- **Paso 1**. Identificación de las variables de decisión; en el cual se definen además 2 nuevas variables para cada objetivo; una para representar la cantidad en el cual el objetivo se pasa del objetivo especificado y la otra para representar la cantidad que está por debajo de la meta.
- **Paso 2**. Identificación de las restricciones.
- **Paso 3**. Identificación de la Función Objetivo: en la programación de metas el objetivo es minimizar la penalización total por no haber logrado las dos metas. Aplicando la descomposición se tiene el siguiente resultado:

Penalización Total= (Penalización por no alcanzar la meta)+ (Penalización por exceder la meta)

2.2.1. Ejemplo

Veremos un ejemplo (basado en Taha, 2012):

- **Descripción del ejemplo**: En cierto país de 20.000 habitantes se tienen las siguientes bases tributarias: 550 millones por predial, 35 millones por alimentos y medicinas y 55 millones por ventas. El consumo anual de gasolina es de 7.5 millones de galones. Se quieren satisfacer las siguientes metas:
  - Meta 1: Tener un ingreso por impuestos de 16 millones.
  - Meta 2: Que el impuesto para alimentos y medicinas no exceda el 10 % del total de impuestos.

- Meta 3: Que el impuesto sobre ventas no exceda el 20 % del total de impuestos.
- Meta 4: Que el impuesto para gasolina no exceda de 2 centavos por galón.

Veamos como definir el problema:

- **Variables** (Paso 1): Definimos las siguientes variables:
  - x_1 = tasa tributaria predial
  - x_2 = tasa tributaria por alimentos y medicinas
  - x_3 = tasa tributaria por ventas
  - x_4 = impuesto para gasolina en centavos por galón.
  - d_i^+= variable de desviación por exceso de la meta i; i=1,. . . ,4
  - d_i^-= variable de desviación por defecto de la meta i; i=1,. . . ,4
- **Metas**: Las metas quedarían expresadas de la siguiente forma:
  - Meta 1: $550x_1 + 35x_2 + 55x_3 + 0.075x_4 \geq 16$
  - Meta 2: $35x_2 \leq 0.1(550x_1 + 35x_2 + 55x_3 + 0.075x_4)$

  Haciendo las operaciones correspondientes, y simplificando, la meta anterior quedaría:

  $55x_1 - 31.5x_2 + 5.5x_3 + 0.0075x_4 \geq 0$
  - Meta 3: $55x_3 \leq 0.2(550x_1 + 35x_2 + 55x_3 + 0.075x_4)$

  Haciendo las operaciones correspondientes, y simplificando, la meta anterior quedaría:

  $110x_1 + 7x_2 - 44x_3 + 0.015x_4 \geq 0$
  - Meta 4: $x_4 \leq 2$

La planificación por metas incluyendo las variables de desviación (Paso 2) sería:

$550x_1 + 35x_2 + 55x_3 + 0.075x_4 + d_1^- - d_1^+ = 16$

$55x_1 - 31.5x_2 + 5.5x_3 + 0.0075x_4 + d_2^- - d_2^+ = 0$

$110x_1 + 7x_2 - 44x_3 + 0.015x_4 + d_3^- - d_3^+ = 0$

$x_4 + d_4^- - d_4^+ = 0$

- **Función objetivo** (**Paso 3**) :

La función objetivo sería:

$min\ D = d_1^- + d_1^+ + d_2^- + d_2^+ + d_{3+}^- + d_3^+ + d_4^- + d_4^+$

2.3. Solución gráfica

Expondremos un método para resolver un problema de programación por metas en el caso de dos variables de decisión.

La diferencia fundamental con el método de resolución gráfica expuesta en el capítulo anterior es que este en vez de buscar un punto extremo de la región factible, buscará una región que sea un compromiso para el conjunto de objetivos del problema.

Para ello tendremos que elegir un conjunto de prioridades $P_1, ..., P_n$ para agrupar las variables de la función objetivo, estas prioridades serán elegidas por el decisor; siendo las variables acompañadas por P_1 las de mayor relevancia y las acompañadas por P_n las de menor relevancia. Consistirá en:

- **Paso 1**: Dibujar las restricciones de meta correspondientes a las restricciones rígidas en términos de las variables de decisión, es decir, dada $z_i(x) - d_i^+ + d_i^- = \hat{z}_i$ dibujaremos $z_i(x) = \hat{z}_i$ para toda meta en P_1.
- **Paso 2**: Determinar el conjunto de soluciones para P_1 (si existe), que será la región factible para el problema (F).
- **Paso 3**: Pasamos a la siguiente prioridad y determinamos el (mejor) conjunto de soluciones de F para este nuevo conjunto de metas, sin que dejen de satisfacerse los valores obtenidos para las metas en las prioridades de orden superior.
- **Paso 4**: Repetir el paso 3 con la sucesión de prioridades, hasta que se converja a un punto o se hayan evaluado todas.

2.3.1. Ejemplo

Veamos un ejemplo. Suponemos que tenemos la siguiente información

$$
\begin{aligned}
&min\ P = P_1(d_1^+ + d_2^+ + d_3^+) + P_2(d_4^-) + P_3(d_5^+) \\
&s.a.\ x_1 + 3x_2 - d_1^+ + d_1^- = 24 \\
&2x_1 + x_2 - d_2^+ + d_2^- = 18 \\
&x_1 + x_2 - d_3^+ + d_3^- = 10 \\
&2x_1 + 3x_2 - d_4^+ + d_4^- = 21 \\
&4x_1 + 2x_2 - d_5^+ + d_5^- = 12 \\
&x_j, d_i^+, d_i^- \geq 0,\ j = 1, 2,\ i = 1, \dots 5
\end{aligned}
\tag{2.3}
$$

En primer lugar dibujamos sobre un sistema de coordenadas (x_1, x_2) las metas correspondientes a la primera prioridad como funciones en las variables de decisión. Para cada meta consideramos el efecto de incrementar d_i^- o d_i^+ que se corresponderá con las rectas paralelas a las que dibujaremos en la siguiente imagen (rectas paralelas del semiplano inferior para incrementar d_i^- y las del semiplano superior para aumentar d_i^+), todo esto lo representaremos con flechas.

Si consideramos las tres metas de la primera prioridad (P_1) como nuestro objetivo es minimizar hacemos 0 las siguientes variables: d_1^+, d_2^+, d_3^+ .

La región que queda formará la región factible del problema que se corresponderá a la zona sombreada en celeste de la siguiente figura (recordemos $d_1^-, d_2^-, d_3^-, x_1, x_2$ no negativas)

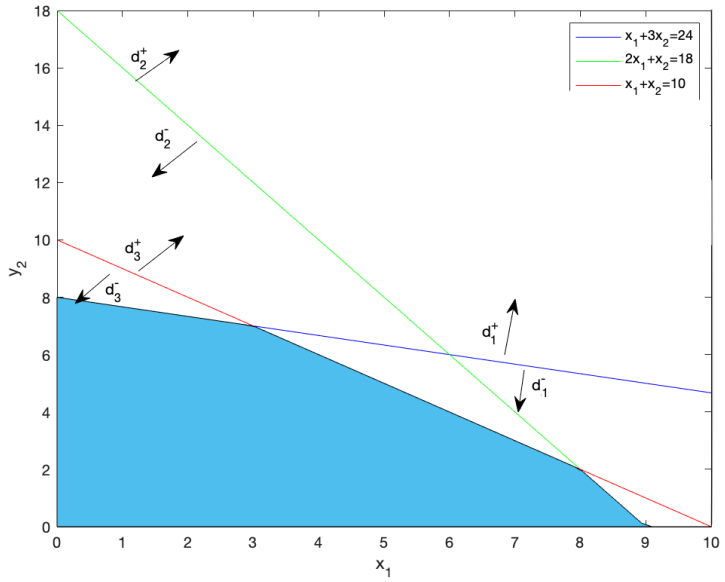

Figura 2.2: Ejemplo solución gráfica 1

Ahora trabajaremos con la prioridad 2 (P_2), en este caso para minimizar la función objetivo observamos que debemos minimizar d_4^- que observamos que podemos hacerla cero sin incrementar d_1^+, d_2^+, d_3^+ . Por lo tanto la nueva región reducida que represente al conjunto de soluciones que satisfacen las restricciones rigidas y la meta ($d_4^- = 0$) será la región pintada en verde:

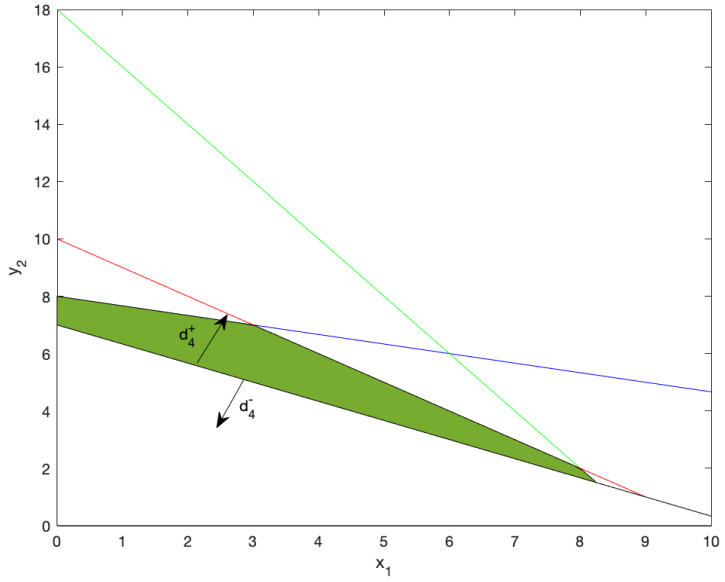

Figura 2.3: Ejemplo solución gráfica 2

Finalmente pasaremos a la última propiedad (P_3), queremos minimizar d_5^+ y si la dibujamos en la gráfica anterior se corresponde con:

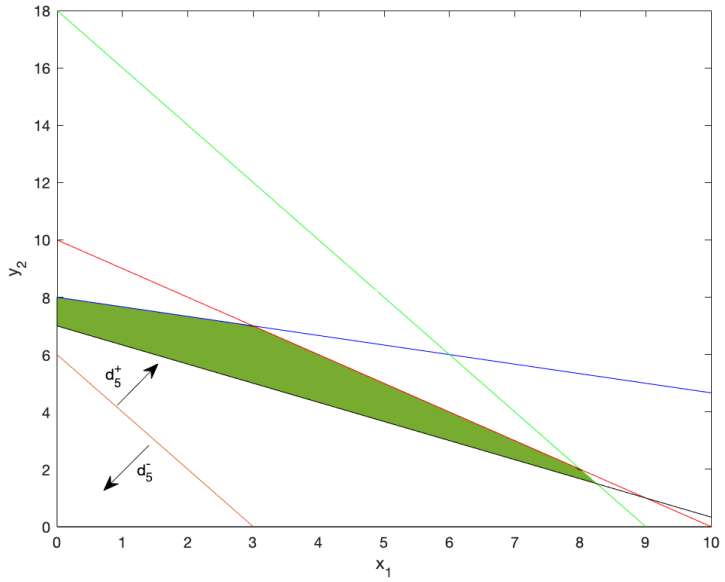

Figura 2.4: Ejemplo solución gráfica 3

Conclusión: no se puede hacer igual a 0 sin que se deje de satisfacer el nivel previamente establecido para P_2. Además el valor más pequeño de d_5^+ para el que se cumple la prioridad anterior es 2 en el punto (0,7); por tanto hemos reducido el espacio de soluciones a un punto terminando así este método. Con la solución obtenida, la última meta quedará superada por un exceso de dos unidades.

2.4. Modelos de programación por metas

En esta sección veremos varios modificaciones de los problemas de programación por metas; las cuales nos permitirán resolver este tipo de problemas usando técnicas de programación lineal. Estos modelos serán: programación por metas poneradas, programación por metas minimax, algoritmo lineal secuencial y programación por metas lexicográficas. También haremos una breve descripción de la programación por metas generalizada.

Figura 2.5: Fases Programación por metas

2.4.1. Programación por metas ponderadas

Tendremos un problema con la siguiente forma:

$$\begin{gathered} min\ z = \sum_{i=1}^{p}(\alpha_i d_i^+ + \beta_i d_i^-) \\ s.a.\ x \in F \\ z_i(x) - d_i^+ + d_i^- = \hat{z}_i \\ d_i^+, d_i^- \geq 0,\ i = 1, \dots p \end{gathered} \tag{2.4}$$

La idea básica es ponderar las variables de desviación. Normalmente los distintos criterios vienen dados en distintas unidades, así que la función objetivo estaría agregando valores de distintas unidades. Una primera opción es ponderar las desviaciones dividiendo por el nivel de aspiración, con lo que serían desviaciones porcentuales que no tienen unidades y así corregimos el efecto de las distintas magnitudes de estas. Por otro lado, si sólo se ponderan dividiendo por el nivel de aspiración, implícitamente lo que se está haciendo es dar una misma importancia a todos los criterios. Si no es ese el caso, se deben multiplicar por pesos que muestren la importancia dada por el decisor a cada meta.

2.4.1.1. Ejemplo programación por metas ponderadas 1

Un empresario ha de producir un compuesto basado en un determinado componente básico. Este componente puede ir en una cantidad variable entre el 50 y el 100 % del total de la composición. Por otra parte del componente básico hay dos calidades, una calidad superior que cuesta 200€/unidad y otra que cuesta 100€. El mercado le obliga a que al menos un 20 % ha de ser del componente de calidad superior. El empresario se plantea maximizar la calidad y minimizar el coste.

Para completar el ejemplo hemos de dar un nivel de aspiración para la calidad y uno para el coste. Por ejemplo, supongamos que queremos una calidad del 75 % y un coste de 10000€. Y supongamos que no damos preferencias subjetivas más allá de las marcadas por los propios niveles de aspiración. En ese caso el modelo sería:

Definimos las variables:

- x_1: cantidad del componente de calidad alta
- x_2: cantidad del componente de cantidad baja

Tendremos por tanto el siguiente problema:

$$
\begin{aligned}
&min\ d_1^- + d_2^+ \\
&s.a.\ x_1 \geq 20 \\
&x_1 + x_2 \leq 100 \\
&x_1 + x_2 \geq 50 \\
&x_1 - d_1^+ + d_1^- = 75 \\
&200x_1 + 100x_2 - d_2^+ + d_2^- = 10000 \\
&x_j, d_i^+, d_i^- \geq 0,\ j = 1, 2,\ i = 1, \ldots 2
\end{aligned}
\tag{2.5}
$$

Si aplicamos la programación por metas ponderadas ponderando por las desviaciones; es decir; dividiendo por el nivel de aspiración, obtendremos el siguiente problema:

$$
\begin{aligned}
&min\ \tfrac{d_1^-}{75} + \tfrac{d_2^+}{10000} \\
&s.a.\ x_1 \geq 20 \\
&x_1 + x_2 \leq 100 \\
&x_1 + x_2 \geq 50 \\
&x_1 - d_1^+ + d_1^- = 75 \\
&200x_1 + 100x_2 - d_2^+ + d_2^- = 10000 \\
&x_j, d_i^+, d_i^- \geq 0,\ j = 1, 2,\ i = 1, \ldots 2
\end{aligned}
\tag{2.6}
$$

La solución es 50 de Calidad y 10000 de Coste, es decir, se cumple la meta del coste a cambio de incumplir la de la calidad. Esto muestra que no es posible cumplir las dos metas a la vez, pero además, este método en muchas ocasiones da una solución de este tipo, en la que unas metas se cumplen por completo y otras se incumplen totalmente (normalmente por un tema de unidades).

2.4.1.2. Ejemplo programación por metas ponderadas 2

TopAd, una nueva agencia de publicidad con 10 empleados, firmó un contrato para promover un nuevo producto. La agencia puede hacer publicidad por radio y televisión. La siguiente tabla proporciona la cantidad de personas alcanzadas diariamente por cada tipo de anuncio publicitario, así como los requerimientos de costos y mano de obra.

	Radio	Televisión
Exposición(en millones de personas)/min	4	8
Costo en miles de dólares por minuto	8	24
Empleados asignados/min	1	2

El contrato prohíbe a TopAd utilizar más de 6 minutos de publicidad por radio. Además, los anuncios de radio y televisión tienen que llegar al menos a 45 millones de personas.

TopAd tiene una meta presupuestaria de 100,000 dólares para el proyecto. ¿Cuántos minutos de anuncios de radio y televisión debe utilizar TopAd?

Sean x_1 y x_2 los minutos asignados a los anuncios de radio y televisión. La formulación del problema como un problema de programación por metas será:

$$\begin{gathered}
Minimizar\ G_1^- = d_1^-\ (Satisfacer\ la\ meta\ de\ exposición) \\
Minimizar\ G_2^+ = d_2^+\ (Satisfacer\ la\ meta\ de\ presupuesto) \\
s.a.\ 4x_1 + 8x_2 + d_1^- - d_1^+ = 45\ (Meta\ de\ exposición) \\
8x_1 + 24x_2 + d_2^- - d_2^+ = 100\ (Meta\ de\ presupuesto) \\
x_1 + 2x_2 \leq 10\ (Límite\ de\ presonal) \\
x_1 \leq 6\ (Límite\ de\ radio) \\
x_j, d_i^+, d_i^- \geq 0,\ j = 1, 2,\ i = 1, \ldots 2
\end{gathered} \tag{2.7}$$

La gerencia de TopAd estima que la meta de exposición es dos veces más importante que la meta de presupuesto; usando esto podremos resolver el problema utilizando metas ponderadas. Por lo tanto, la función objetivo combinada se convierte en

$$Minimizar\ z\ = 2G_1 + G_2 = 2d_1^- + d_2^+$$

Teniendo por tanto el problema:

$$\begin{gathered}
Minimizar\ z\ = 2G_1 + G_2 = 2d_1^- + d_2^+ \\
s.a.\ 4x_1 + 8x_2 + d_1^- - d_1^+ = 45\ (Meta\ de\ exposición) \\
8x_1 + 24x_2 + d_2^- - d_2^+ = 100\ (Meta\ de\ presupuesto) \\
x_1 + 2x_2 \leq 10\ (Límite\ de\ presonal) \\
x_1 \leq 6\ (Límite\ de\ radio) \\
x_j, d_i^+, d_i^- \geq 0,\ j = 1, 2,\ i = 1, \ldots 2
\end{gathered} \tag{2.8}$$

Resolviendo el problema de PL obtenemos que la solución óptima es z =10, $x_1 = 5$ minutos, $x_2 = 2.5$ minutos, $d_1^- = 5$ millones de personas, $d_2^+ = 0$.

El hecho de que el valor óptimo de z no sea cero indica que al menos una de las metas no se cumple. Específicamente, $d_1^- = 5$ significa que la meta de exposición (de al menos 45 millones de personas) falla por 5 millones de personas. Por otra parte, la meta de presupuesto (de no exceder 100,000 dólares) no se viola porque $d_2^+ = 0$.

Comentarios: La programación de metas busca sólo una solución eficiente, más que óptima. Por ejemplo, la solución $x_1 = 6$ y $x_2 = 2$ produce la misma exposición (4 x 6 x 8 x 2) = 40 millones de personas) pero cuesta menos (8 x 6 x 24 x 2) =\$96,000). En esencia, lo que la programación de metas hace es hallar una solución que satisfaga las metas sin darle tanta importancia a la optimización. Algunos autores piensan que el hecho de no obtener la solución óptima pone en evidencia la viabilidad de la programación por metas como una técnica de optimización.

2.4.2. Programación por metas Minimax o Tchebychev

En este caso se busca una solución "equilibrada", de modo que ninguna de las metas se desvíe en exceso de su nivel de aspiración. Para ello se minimiza la máxima distancia a este nivel, es decir, el problema se plantearía en los siguientes términos:

$$\begin{gathered} min\ D \\ s.a.\ x \in F \\ \alpha_i d_i^+ + \beta_i d_i^- \leq D\ i = 1, \ldots, p \\ z_i(x) - d_i^+ + d_i^- = \hat{z}_i\ i = 1, \ldots, p \\ d_i^+, d_i^- \geq 0\ i = 1, \ldots p \end{gathered} \tag{2.9}$$

2.4.2.1. Ejemplo programación por metas Minimax o Tchebychev

Aplicamos esta modificación al problema (2.5) y obtenemos el problema:

$$\begin{gathered} min\ D \\ s.a.\ x_1 \geq 20 \\ x_1 + x_2 \leq 100 \\ x_1 + x_2 \geq 50 \\ x_1 - d_1^+ + d_1^- = 75 \\ 200x_1 + 100x_2 - d_2^+ + d_2^- = 10000 \\ \frac{d_1^-}{75} \leq D \\ \frac{d_2^+}{1000} \leq D \\ x_j, d_i^+, d_i^- \geq 0,\ j = 1, 2,\ i = 1, \ldots 2 \end{gathered} \tag{2.10}$$

cuya solución obtenida resolviendo como un problema de PL es 60 de Calidad y 12000 de Coste, una solución equilibrada (a diferencia del método anterior), que se desvía un

20 % de los niveles de aspiración de ambos criterios.

2.4.3. Algoritmo lineal secuencial

Este método consiste en resolver sucesivamente los problemas lineales con un solo objetivo que se obtienen al establecer una partición del problema de acuerdo con los niveles de prioridad, comenzaremos resolviendo la prioridad más alta (P_1) y así sucesivamente hasta alcanzar el nivel más bajo. Fue inspirado en el método lexicográfico de Ignizio (1976); del cual hablaremos más adelante; propuesto por Debreu en los años cincuenta y en el que se han introducido varias modificaciones para adaptarlos a la programación por metas.

Introduciremos una nueva forma de representar la función objetivo, teniendo en cuenta que el significado de los factores de prioridad es como una ordenación lexicográfica para K factores de prioridad, donde $d^+ = (d_1^+ \dots d_m^+)$ y $d^- = (d_1^- \dots d_m^-)$. Por tanto la función objetivo tendrá la forma:

$$min\ lex\ \lambda = z_1(d^+, d^-), \dots, z_k(d^+, d^-)$$

EL algoritmo será:

- **Paso 0**: Hacer k=1
- **Paso 1**: Considerar el problema lineal (uniobjetivo) para P_1

$$\begin{gathered} min\ \lambda_1 = z_1(d^+, d^-) \\ z_i(x) - d_i^+ + d_i^- = \hat{z}_i, i \in P_1 \\ x \geq 0 \\ d_i^+, d_i^- \geq 0,\ i \in P_1 \end{gathered} \tag{2.11}$$

- **Paso 2**: Resolver el problema construido. Sea λ_k^* el valor óptimo de λ_k
- **Paso 3**: Hacer k=k+1. Si k>K, ir a 5. En caso contrario ir a 4.
- **Paso 4**: Considerar el problema lineal uniobjetivo

$$\begin{gathered} min\ \lambda_k = z_k(d^+, d^-) \\ z_q(x) - d_q^+ + d_q^- = \hat{z}_q, q \in P_1 \cup \dots P_k \\ z_t(d^+, d^-) = \lambda_t^*, t = 1, \dots, k-1 \\ x \geq 0 \\ d^+, d^- \geq 0 \end{gathered} \tag{2.12}$$

e ir a 2.

- **Paso 5**: El vector x^* solución del último problema resuelto, es la solución del problema de programación por metas original.

Observación: En el ejemplo (2.5) no tiene mucho sentido presentarlo. Por ejemplo, si damos prioridad a la calidad, y un nivel de aspiración de 75 %, el primer paso será aproximarse a este nivel. Dado que es posible, la solución del primer paso fijará la calidad en 75 % o más, y el segundo nivel, el del coste, nos dará la solución de menos coste para esa calidad, es decir, 75 % de Calidad y 1500 de Coste. Veremos por tanto otro ejemplo:

2.4.3.1. Ejemplo algoritmo lineal-secuencial 1

Consideramos un problema con la siguiente forma:

$$\begin{gathered} min\ P = P_1(d_1^+ + d_2^+ + d_3^+) + P_2(d_4^-) + P_3(d_5^+) \\ s.a.\ x_1 + 3x_2 - d_1^+ + d_1^- = 24 \\ 2x_1 + x_2 - d_2^+ + d_2^- = 18 \\ x_1 + x_2 - d_3^+ + d_3^- = 10 \\ 2x_1 + 3x_2 - d_4^+ + d_4^- = 21 \\ 4x_1 + 2x_2 - d_5^+ + d_5^- = 12 \\ x_j, d_i^+, d_i^- \geq 0,\ j = 1, 2,\ i = 1, \ldots 5 \end{gathered} \tag{2.13}$$

- **Paso 0**: Hacer k=1.
- **Paso 1**: Consideramos el problema:

$$\begin{gathered} min\ \lambda_1 = d_1^+ + d_2^+ + d_3^+ \\ s.a.\ x_1 + 3x_2 - d_1^+ + d_1^- = 24 \\ 2x_1 + x_2 - d_2^+ + d_2^- = 18 \\ x_1 + x_2 - d_3^+ + d_3^- = 10 \\ x_1, x_2, d_1^+, d_2^+, d_1^-, d_2^-, d_3^+, d_3^- \geq 0 \end{gathered} \tag{2.14}$$

- **Paso 2**: Aplicando el método del simplex obtenemos el valor óptimo $\lambda_1^* = 0$.
- **Paso 3**: Hacemos k=k+1=2 y como $k = 2 \leq 3 = K$ vamos a 4.

- **Paso 4**: Consideramos el siguiente problema:

$$\begin{gathered} min\ \lambda_2 = d_4^- \\ s.a.\ x_1 + 3x_2 - d_1^+ + d_1^- = 24 \\ 2x_1 + x_2 - d_2^+ + d_2^- = 18 \\ x_1 + x_2 - d_3^+ + d_3^- = 10 \\ 2x_1 + 3x_2 - d_4^+ + d_4^- = 21 \\ d_1^+ + d_2^+ + d_3^+ = 0 \\ x_j, d_i^+, d_i^- \geq 0,\ j = 1, 2,\ i = 1, \ldots 4 \end{gathered} \tag{2.15}$$

  y vamos al paso 2.
- **Paso 2**: Aplicamos el método del simplex al problema anterior y obtenemos $\lambda_2^* = 0$.
- **Paso 3**: Hacemos k=k+1=3 y como $k = 3 \leq 3 = K$ vamos a 4.
- **Paso 4**: Consideramos el siguiente problema:

$$\begin{gathered} min\ \lambda_3 = d_5^+ \\ s.a.\ x_1 + 3x_2 - d_1^+ + d_1^- = 24 \\ 2x_1 + x_2 - d_2^+ + d_2^- = 18 \\ x_1 + x_2 - d_3^+ + d_3^- = 10 \\ 2x_1 + 3x_2 - d_4^+ + d_4^- = 21 \\ 4x_1 + 2x_2 - d_5^+ + d_5^- = 12 \\ d_1^+ + d_2^+ + d_3^+ = 0, d_4^- = 0 \\ x_j, d_i^+, d_i^- \geq 0,\ j = 1, 2,\ i = 1, \ldots 5 \end{gathered} \tag{2.16}$$

  y vamos al paso 2.
- **Paso 2**: Aplicamos el método del simplex al problema anterior y obtenemos $\lambda_3^* = 2$.
- **Paso 3**: Hacemos k=k+1=4 y como $k = 4 > 3 = K$ vamos a 5.
- **Paso 5**: La solución del problema original será la solución óptima del último problema, que es $x_1^* = 0$ y $x_2^* = 7$.

2.4.4. Programación por metas lexicográficas (método preventivo)

En este tipo de método, el tomador de decisiones clasifica las metas del problema en orden de importancia. Dada una situación de n metas, los objetivos del problema se

escriben como

$$Minimizar\ G_1 = \rho_1(Máxima\ prioridad)$$
$$\vdots$$
$$Minimizar\ G_n = \rho_n(Mínima\ proridad)$$

La variable ρ_i es el componente de las variables de desviación d_i^- o d_i^+, que representan la meta i. Por ejemplo, en el modelo de TopAd (ejemplo 2.7), $\rho_1 = d_1^-$ y $\rho_2 = d_2^+$.

El procedimiento de solución se inicia con la optimización de la prioridad máxima G_1, y termina con la optimización de la prioridad mínima G_n. El método preventivo está diseñado de modo que una solución de menor prioridad nunca degrade a una solución de alta prioridad.

El método consiste en dar los siguientes pasos:

- **Paso 0**. Identifique las metas del modelo y clasifíquelas en orden de prioridad: $G_1 = \rho_1 \succ G_2 = \rho_2 \succ, \ldots, \succ G_n = \rho_n$. Establezca i=1.
- **Paso general** Resuelva el problema de programación lineal para cada i (LPi) que minimice G_i, y que $\rho_i = \rho_i^*$ defina el valor óptimo correspondiente de la variable de desviación ρ_i. Si i=n, deténgase; el problema de programación lineal correspondiente a n resuelve el problema de n metas. En caso contrario, agregue la restricción $\rho_i = \rho_i^*$ a las restricciones del problema G_i para garantizar que el valor no se degrade en problemas futuros. Establezca i=i+1, y repita el paso i.

2.4.4.1. Ejemplo programación por metas lexicográficas 1

Consideramos el ejemplo 2.7 y aplicamos este método:

- **Paso 0**. Establecemos las prioridades: $G_1 > G_2$

$$Minimizar\ G_1^- = d_1^-\ (Satisfacer\ la\ meta\ de\ exposición)$$
$$Minimizar\ G_2^+ = d_2^+\ (Satisfacer\ la\ meta\ de\ presupuesto)$$

- **Paso 1**.

Resolver:

$$\begin{gathered} Minimizar\ G_1^- = d_1^- \\ s.a.\ 4x_1 + 8x_2 + d_1^- - d_1^+ = 45\ (Meta\ de\ exposición) \\ 8x_1 + 24x_2 + d_2^- - d_2^+ = 100\ (Meta\ de\ presupuesto) \\ x_1 + 2x_2 \leq 10\ (Límite\ de\ presonal) \\ x_1 \leq 6\ (Límite\ de\ radio) \\ x_j, d_i^+, d_i^- \geq 0,\ j = 1, 2,\ i = 1, \ldots 2 \end{gathered} \tag{2.17}$$

La solución óptima es $x_1 = 5$ minutos, $x_2 = 2.5$ minutos, $d_1^- = 5$ millones de personas, con las variables restantes iguales a cero. La solución muestra que 5 millones de personas violan la meta de exposición, G_1. La restricción adicional que añade al problema G_2 es $d_1^- = 5$.

- **Paso 2**: Aquí tendremos que resolver:

La función objetivo de la PL2 es

$$Minimizar\ G_2^+ = d_2^+$$

Las restricciones son las mismas que en el paso 1 más la restricción adicional $d_1^- = 5$. Por lo general, la restricción adicional $d_1^- = 5$ también puede explicarse al sustituir en la primera restricción. El resultado es que el lado derecho de la restricción de la meta de exposición cambiará de 45 a 40, lo que reduce la LP2 a

$$\begin{gathered} Minimizar\ G_2^+ = d_2^+ \\ s.a.\ 4x_1 + 8x_2 - d_1^+ = 40\ (Meta\ de\ exposición) \\ 8x_1 + 24x_2 + d_2^- - d_2^+ = 100\ (Meta\ de\ presupuesto) \\ x_1 + 2x_2 \leq 10\ (Límite\ de\ presonal) \\ x_1 \leq 6\ (Límite\ de\ radio) \\ x_j, d_i^+, d_i^- \geq 0,\ j = 1, 2,\ i = 1, \ldots 2 \end{gathered} \tag{2.18}$$

En realidad, la optimización de la PL2 no es necesaria en este problema porque la solución óptima al problema G1 ya da por resultado $d_2^+ = 0$; es decir, ya es óptima para la PL2. Tales oportunidades de ahorro de cálculos deben aprovecharse siempre que se presenten durante el curso de implementación del método preventivo.

2.4.4.2. Ejemplo programación por metas lexicográficas 2

En este ejemplo demostraremos que puede obtenerse una mejor solución para el problema (2.7) si se utiliza el método preventivo para optimizar los objetivos en lugar de satisfacer las metas. Las metas del ejemplo anterior se puede formular como:

Prioridad 1: Minimizar la exposición (P1)

Prioridad 2: Minimizar el costo (P2) matemáticamente, los dos objetivos se dan como:

$$Minimizar\ P1 = 4x_1 + 8x_2(Exposición)$$

$$Minimizar\ P2 = 8x_1 + 24x_2(Costo)$$

Los límites específicos para las metas de exposición y de costo (= 45 y 100) en los ejemplos anteriores se eliminan, porque dejaremos que el método simplex determine estos límites óptimamente. Por lo tanto el nuevo problema se formula como

$$
\begin{gathered}
Minimizar\ P1 = 4x_1 + 8x_2 \\
Minimizar\ P2 = 8x_1 + 24x_2 \\
s.a.\ x_1 + 2x_2 \leq 10\ (\textit{Límite de presonal}) \\
x_1 \leq 6\ (\textit{Límite de radio}) \\
x_j,\ j = 1, 2
\end{gathered}
\tag{2.19}
$$

Primero resolvemos el problema siguiendo el procedimiento presentado en el ejemplo de arriba.

- **Paso 1.** Resuelva la PL1.

$$
\begin{gathered}
Minimizar\ P1 = 4x_1 + 8x_2 \\
s.a.\ x_1 + 2x_2 \leq 10\ (\textit{Límite de presonal}) \\
x_1 \leq 6\ (\textit{Límite de radio}) \\
x_j,\ j = 1, 2
\end{gathered}
\tag{2.20}
$$

La solución óptima es $x_1 = 0$, $x_2 = 5$ con P1 =40, lo que demuestra que la exposición máxima que podemos obtener es de 40 millones de personas.

- **Paso 2**. Agregue la restricción $4x_1 + 8x_2 \geq 40$ para asegurarnos de que la meta G1 no se degrade. Por lo tanto, resolvemos la PL2 como:

$$
\begin{gathered}
Minimizar\ P2 = 8x_1 + 24x_2 \\
s.a.\ x_1 + 2x_2 \leq 10\ (\textit{Límite de presonal}) \\
x_1 \leq 6\ (\textit{Límite de radio}) \\
4x_1 + 8x_2 \geq 40 (\textit{restricción adicional}) \\
x_j,\ j = 1, 2
\end{gathered}
\tag{2.21}
$$

  La solución óptima del PL2 es P2=96,000 dólares, $x_1 = 6$ minutos, y $x_2 = 2$ minutos. Esto da por resultado la misma exposición (P1 =40 millones de personas) pero a un costo menor que el del ejemplo de arriba, donde buscamos satisfacer en lugar de optimizar las metas.

2.4.5. Programación por metas generalizada

Otra modificación importante de la programación por metas consiste en considerar la función objetivo como una suma ponderada de pesos $\lambda_i \geq 0$ y potencias de orden q de las diferencias $|z_i(x) - \hat{z}_i|$. En este caso el problema es:

$$\begin{aligned} min\ z &= \sum_{i=1}^{p} \lambda_i |z_i(x) - \hat{z}_i|^q \\ s.a.\ x &\in F \end{aligned} \tag{2.22}$$

Esta formulación recibe el nombre de **programación por metas generalizada** y normalmente nos lleva a un problema no lineal y por ello no profundizaremos en este tema.

2.5. Temas críticos en la programación por metas

Aunque el éxito de la programación por metas sea indudable, diferentes autores han apuntado algunas debilidades de los modelos de programación por metas. La mayoría se presentan en las metas lexicográficas. Algunas de estas debilidades las expondremos a continuación.

- **1. Posible equivalencia de soluciones entre modelos de programación por metas y modelos de optimización de un solo criterio.**

  En las metas **lexicográficas** si se fijan unos valores muy pesimistas en los primeros niveles (fáciles de alcanzar) y muy optimistas en los últimos (difíciles de alcanzar), al resolverlo para los primeros niveles se obtendrá un valor 0 de las desviaciones, y llegará un momento en que para una meta no haya óptimos alternativos, por lo tanto, estaremos haciendo redundantes las metas de prioridades más bajas.

  Esto también podemos verlo en metas **ponderadas**, por ejemplo, si se plantean unos niveles de aspiración muy pesimistas para las metas, excepto para una en que resulte muy optimista. En dicho caso, el problema puede ser equivalente a optimizar ese atributo incluyendo las demás metas como restricciones.

  En cualquier caso, estos problemas se derivan de la formulación del modelo y no a debilidades de la metodología de la programación por metas.

- **2. Conclusiones equivocadas por agregar metas lexicográficas en una función objetivo.**

  Al plantear un modelo de programación por metas lexicográficas, suele haber una tendencia a intentar plantear un modelo equivalente que se pueda resolver en una única iteración. Así es usual ver que el problema se modela con una función objetivo como en el procedimiento de metas ponderadas donde se multiplica cada variable de desviación por un peso que dependerá del nivel de prioridad (así para el primer nivel el peso es mucho mayor que para el segundo, etc.). En general, este incorrecto planteamiento, puede llevar a obtener soluciones que difieran mucho de las que se obtendrían con el modelo lexicográfico, y por lo tanto, darán lugar a conclusiones claramente incorrectas.

- **3. Problemas derivados de la omisión de una variable de desviación.**

  Otros problemas provienen de no incluir algunas de las variables de desviación. En muchas ocasiones, se ha prescindido de la variable de desviación que no es considerada, por ejemplo, si queremos que un atributo sea superior a un nivel de aspiración, la variable no deseada es la de holgura al nivel de aspiración, d_i^-, sin

embargo podemos pensar en prescindir de la variable de exceso, d_i^+. Hacer esto implica que el atributo no pueda tomar valores superiores al nivel de aspiración, sólo inferiores o iguales, lo que restringe el espacio de soluciones, eliminando muchas de las posibles soluciones. Por esto, la variable de desviación que no es la no deseada, también debe aparecer en el problema. Además de que, en general, no incluirlas no produce una mejora computacional.

- **4. Problemas derivados de la inclusión de metas con dos lados innecesariamente.**

  Las metas con dos lados no deseados son menos habituales que las que tienen un único lado deseado, esto se debe a que es raro que un decisor desee el logro exacto de una meta más que superar, o no hacerlo, un nivel determinado. Incluir entre las no deseadas una desviación que no lo es, puede llevarnos a soluciones subóptimas. Es evidente, que hay que tener un especial cuidado en identificar cuáles son las variables de desviación no deseadas.

- **5. Incompatibilidad entre ordenaciones lexicográficas y la existencia de una función de utilidad.**

  Una de las críticas más fuertes al modelo **lexicográfico** ha sido su incompatibilidad con una función de utilidad del decisor.

  Los axiomas que deben ser cumplidos por una función de utilidad son comparabilidad, reflexividad, transitividad y continuidad (este último es el que falla en este caso). Según este axioma, los conjuntos de nivel de la función de utilidad son superficies continuas, es decir, dada una combinación de dos elecciones posibles de un decisor, siempre se podrá reducir la cantidad de una elección encontrando un incremento de la otra elección que compense exactamente la reducción; lo cual asegura la existencia de un equilibrio competitivo.

  El que esto no se cumpla no es realmente un problema, pues hay situaciones en las que suponer válido este axioma puede ser un disparate; por ejemplo, supóngase un problema de planificación forestal donde se tienen en cuenta varios criterios, por ejemplo, uno económico que fuera la producción de madera y otro que fuera un índice que mida el riesgo de colapso ecológico del bosque. En este caso el supuesto de continuidad diría que siempre existe un incremento en el volumen de madera producida que compense un incremento del riesgo de colapso del bosque, por alto que este sea, lo que puede ser un disparate.

- **6. Posible ineficiencia paretiana de la solución obtenida por metas, y forma de resolverlo.**

  La programación por metas en general no da soluciones eficientes sino soluciones satisfacientes. Es decir podemos obtener una solución de un modelo de programación por metas no eficiente. Para ello, lo que tiene que pasar es que haya óptimos alternativos del problema de metas **ponderadas** o del último nivel de metas **lexicográficas** (esta condición es necesaria aunque no es suficiente).

  Veamos como comprobar la eficiencia: Para comprobar si una solución obtenida mediante programación por metas es eficiente, se fijan los valores de las variables de desviación no deseadas obtenidos, y se maximizan las variables opuestas; si la solución no cambia, es que la solución es eficiente, mientras que si varía, la

nueva solución será una solución eficiente con los mismos valores de las variables no deseadas.

- **7. El concepto de meta redundante y repercusión en las metas lexicográficas:**

  El concepto de metas redundante puede surgir tanto en metas **ponderadas** como en **lexicográficas**, aunque con un significado distinto.

  En metas ponderadas puede surgir porque se plantee una meta inalcanzable y las demás de modo pesimista, lo que implicará que el problema puede desviarse hacia la meta inalcanzable olvidando las demás.

  Sin embargo, este concepto es más relevante en las metas lexicográficas, ya que si en un determinado nivel de prioridad se llega a un único óptimo, todas las metas que se encuentran en niveles inferiores no podrán variar la solución.

  Definiremos como meta redundante aquella cuya omisión no influye en el resultado. La existencia de metas redundantes puede darse principalmente por tres causas:

  1. Una excesiva priorización de las metas, es decir, una agrupación en un número excesivamente elevado de niveles de prioridad.
  2. Ser excesivamente optimista, fijando los niveles de aspiración muy próximos al ideal del atributo.
  3. Incluir muchas metas con dos lados, obligando a la minimización de ambas variables de desviación.

  Cuando se da la redundancia es interesante saber en qué nivel de prioridad se ha dado para así poder replantear el problema. El replanteamiento puede hacerse:

  1. Redefiniendo los niveles de aspiración de las metas anteriores.
  2. Viendo si se han incluido metas de dos lados innecesariamente.
  3. Agrupando en distintos niveles, por ejemplo, se puede plantear un último nivel con la última meta no redundante ponderada para darle más importancia y a las metas restantes redundantes darle una ponderación menor.

Capítulo 3

Resolución de distintos problemas con R y AMPL

Los problemas a los que día a día se enfrentan matemáticos y estadísticos suelen tener grandes dimensiones, y esto unido a la complejidad, hace que sea necesario el uso de los ordenadores y con ello el uso de programas informáticos.

En este trabajo, nos ayudaremos de programas específicos como R y AMPL para la resolución de los problemas del capítulo 2 y comprobaremos que los resultados obtenidos coinciden con la información que hemos recopilado de los libros.

3.1. Introducción a R

R es un software de uso libre que presenta versiones tanto para Windows, Linux como para Mac. R fue desarrollado por Bell Laboratories y es capaz de resolver varios tipos de problemas; como ya he visto en las asignaturas de Inferencia Estadística, Modelos Lineales y Diseño de Experimentos y Análisis de datos multivariantes.

Cuenta con diversas librerías para resolver distintos problemas de Programación y optimización. Para resolver problemas de programación por metas como realmente trabajaremos con problemas de Programación Lineal, usaremos cualquier paquete de R que resuelva este tipo de problemas; entre ellos destacaremos los paquetes "linprog", "lpSolve" y "lpSolveAPI".

He trabajado con RStudio. RStudio es un entorno de desarrollo integrado (IDE) para el lenguaje de programación R, dedicado a la computación estadística y gráficos. Incluye una consola, editor de sintaxis que facilita la ejecución de código, así como herramientas para el trazado, la depuración y la gestión del espacio de trabajo.

Figura 3.1: Logo de la aplicación RStudio

Su página web oficial es https://www.rstudio.com

3.1.1. Paquete lpSolveAPI

A continuación, se recoge información sobre el uso de la librería lpSolveAPI (véase [10]).

Comenzaremos con una breve explicación de como usaremos este paquete para resolver problemas de minimización en programación lineal.

- Primero, suponemos que tenemos n restricciones y m variables de decisión. Usaremos el comando **lprec <- make.lp(n, m)** para fijar las dimensiones.
- Después iremos fijando cada columna de la matriz de restricciones de la siguiente forma: **set.column(lprec, 1, vector1), set.column(lprec, 2, vector2),. . . , set.column(lprec, m, vectorm)**; donde vectori con i=1,. . .,m es el vector cuya entrada son los coeficientes de la columna i-ésima de la matriz de restricciones.
- A continuación creamos la función objetivo con el comando: **set.objfn(lprec, vector con los coeficientes de la función objetivo)**.
- Crearemos otro vector con cada una de las desigualdades correspondientes a las distintas restricciones: **set.constr.type(lprec, c(“>=”, “<=”, “>=”. . .))**.
- Por último, creamos un vector con la información del vector b, es decir, sus coeficientes serán los del lado derecho de las desigualdades de cada restricción: **set.rhs(lprec,vector)**.

Finalmente, usaremos el comando **“get.objetive(lprec)”** para obtener el valor de la función objetivo y con el comando **“get.variables(lprec)”** obtendremos la solución óptima. Si el comando **“solve(lprec)”** nos devuelve un 0 significará que el problema se ha resuelto con éxito.

3.1.2. Paquete linprog

A continuación, se recoge información sobre el uso de la librería linprog (véase [6]).

La solución de problemas de programación lineal en lenguaje R generalmente utiliza el paquete linprog y su función principal es solveLP que tendrá la siguiente forma:

```
solveLP(objetivo, brestricciones, mrestricciones,
        maximum = FALSE, des, lpSolve = FALSE)
```

- Cuyos argumentos son:
  - **objetivo**: vector con n elementos, en el cual introduciremos los coeficientes de la función objetivo.
  - **brestricciones**: vector con m elementos, en el que introduciremos los coeficientes de las restricciones.
  - **mrestricciones**: matriz A (dimensiones m × n), cuyas entradas serán los coeficientes de las restricciones.
  - **maximum**: introduciremos un valor lógico, según se trate de un problema de maximización o minimización.
  - **des**: vector de cadenas de caracteres que dan las direcciones de las restricciones: cada valor debe ser uno de "<," "<=," "=," "==," ">," o "> =".
  - **lpSolve**: lógico. ¿Debería utilizarse el paquete "lpSolve" para resolver el problema de LP?. Siempre que haya restricciones de igualdad el valor lógico de "lpSolve" debe ser TRUE.
- Datos de salida:
  - **opt**: valor óptimo (mínimo o máximo) de la función objetivo.
  - **solution**: vector de valores óptimos de las variables.
  - **iter1**: iteraciones del algoritmo Simplex en la fase 1. Funciona siempre que indiquemos que "lpSolve" toma el valor lógico FALSE.
  - **iter2**: iteraciones del algoritmo Simplex en la fase 2. Funciona siempre que indiquemos que "lpSolve" toma el valor lógico FALSE.
  - **basvar**: vector de variables básicas (= distintas de cero) (en el óptimo).
  - **maximum**: lógico. Indica si la función objetivo se maximizó o minimizó.
  - **Tab**: 'Tableau' final del algoritmo Simplex.
  - **lpSolve**: lógico. ¿Se ha utilizado el paquete "lpSolve" para resolver el problema de LP?.

Observación: si hacemos un summary obtendremos la solución y el valor de la función objetivo que a priori será lo que más nos interese.

3.1.3. Paquete lpSolve

Hablaremos del comando lp (véase [2]). Tendrá la siguiente forma:

```
sol=lp("min", objetivo, mrestricciones, des, brestricciones)
```

- Cuyos argumentos son:
  - **"min" o "max"**: según se trate de un problema de minimización o maximización.
  - **objetivo**: vector con n elementos, en el cual introduciremos los coeficientes de la función objetivo.

- **mrestricciones**: matriz A (dimensiones m × n), cuyas entradas serán los coeficientes de las restricciones.
- **des**: vector de cadenas de caracteres que dan las direcciones de las restricciones: cada valor debe ser uno de "<," "<=," "=," "==," ">," o "> =".
- **brestricciones**: vector con m elementos, en el que introduciremos los coeficientes de las restricciones.

Observación: Si ponemos "sol$solution" obtendremos la solución del problema.

3.1.4. Ejemplos R

En esta sección resolveremos con los paquetes anteriores de R los problemas 2.5 y 2.7.

3.1.4.1. Problema 1 (Ejemplo 2.5)

Resolveremos el ejemplo 2.5 empleando las mismas modificaciones de programación por metas que en el capítulo anterior:

- Metas ponderadas
- Metas minimax

3.1.4.1.1. Metas poneradas

Como ya vimos en el capítulo anterior si al ejemplo 2.5 le imponemos una calidad del 75 % y un coste de 10000 euros; obtendremos:

$$\begin{gathered} min\ \tfrac{d_1^-}{75} + \tfrac{d_2^+}{10000} \\ s.a.\ x_1 \geq 20 \\ x_1 + x_2 \leq 100 \\ x_1 + x_2 \geq 50 \\ x_1 - d_1^+ + d_1^- = 75 \\ 200x_1 + 100x_2 - d_2^+ + d_2^- = 10000 \\ x_j, d_i^+, d_i^- \geq 0,\ j = 1, 2,\ i = 1, \ldots 2 \end{gathered} \tag{3.1}$$

Reescribiéndolo las restricciones en forma matricial tendremos:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & -1 & 0 & 0 \\ 200 & 100 & 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ d_1^- \\ d_1^+ \\ d_2^- \\ d_2^+ \end{pmatrix} \begin{pmatrix} \geq \\ \leq \\ \geq \\ = \\ = \end{pmatrix} \begin{pmatrix} 20 \\ 100 \\ 50 \\ 75 \\ 1000 \end{pmatrix}$$

Podemos observar que el problema anterior está escrito como un problema de programación lineal y por tanto podremos usar los paquetes de R de esta para resolverlo, lo haremos de tres formas distintas; usando: "solveLP", "lp" y el paquete lpSolveAPI .

Ahora ya podremos pasar a resolver el problema desde RStudio.

En primer lugar con la función solveLP:

```
library(linprog)
library(lpSolve)
#vector función objetivo
objetivo=c(0,0,1/75,0,0,1/10000)
#matriz restricciones
mrestricciones=matrix(c(1,1,1,1,200,
                        0,1,1,0,100,
                        0,0,0,1,0,
                        0,0,0,-1,0,
                        0,0,0,0,1,
                        0,0,0,0,-1),ncol=6)
#vector b , valores restricciones
brestricciones=c(20,100,50,75,10000)
#desigualdades
des=c(">=","<=",">=","=","=")
solucion1=solveLP(objetivo,brestricciones,mrestricciones,
                  maximum=FALSE,des,lpSolve=TRUE,verbose=4)
summary(solucion1)
```

```
##
##
## Results of Linear Programming / Linear Optimization
##
## Objective function (Minimum): 0.3333333
##
## Solution
##   opt
## 1  50
## 2   0
## 3  25
## 4   0
## 5   0
## 6   0
```

```
solucion1$solution
```

```
##  1  2  3  4  5  6
## 50  0 25  0  0  0
```

```
solucion1$opt
```

```
## [1] 0.3333333
```

```
solucion1
```

```
##
##
## Results of Linear Programming / Linear Optimization
## (using lpSolve)
##
## Objective function (Minimum): 0.333333
##
## Solution
##   opt
## 1  50
## 2   0
## 3  25
## 4   0
## 5   0
## 6   0
##
## Constraints
##   actual dir  bvec free
## 1     50  >=    20   30
## 2     50  <=   100   50
## 3     50  >=    50    0
## 4     75   =    75    0
## 5  10000   = 10000    0
```

El comando "solución1" nos da la solución obtenida empleando la función "solveLp" de la librería linprog.

En segundo lugar con la función "lp":

```
solucion2=lp("min",objetivo,mrestricciones,des,brestricciones)
solucion2$solution
```

```
## [1] 50  0 25  0  0  0
```

La "solución2" nos da la solución obtenida empleando la función "lp".

En tercer lugar emplearemos "lpSolveAPI":

```
library(lpSolveAPI)
#Dimensionalizamos
lprec <- make.lp(5,6)
#Fijamos las columnas
set.column(lprec, 1, c(1,1,1,1,200))
set.column(lprec, 2, c(0,1,1,0,100))
set.column(lprec, 3, c( 0,0,0,1,0))
set.column(lprec, 4, c(  0,0,0,-1,0))
set.column(lprec, 5, c( 0,0,0,0,1))
set.column(lprec, 6, c(  0,0,0,0,-1))
#Función objetivo
set.objfn(lprec, c(0,0,1/75,0,0,1/10000) )
```

```
#Desigualdades
set.constr.type(lprec,c(">=","<=",">=","=","=") )
#Vector b
set.rhs(lprec, c(20,100,50,75,10000))
solve(lprec)
```

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

```
get.objective(lprec)
```

```
## [1] 0.3333333
```

```
get.variables(lprec)
```

```
## [1] 50  0 25  0  0  0
```

Finalmente; el objeto "lprec" nos dará la solución obtenida usando la librería lpSolveAPI.

Conclusiones:

Gracias a los comandos: "solucion1$solution", "solucion2$solution" y get.variables(lprec) obtenemos que $d_2^+ = 0$, $d_1^- = 25$ (variables sexta y tercera respectivamente según he definido el problema), lo que significa que se cumple la meta del coste a cambio de incumplir la de la calidad. Las soluciones obtenidas por los tres métodos son idénticas entres sí e iguales a las obtenidas en el capítulo 2. Además podemos observar que el valor de la función objetivo será 0.3333333.

3.1.4.1.2. Metas minimax

En este caso como ya vimos en el capítulo dos el problema se formula como:

$$\begin{gathered} min\ D \\ s.a.\ x_1 \geq 20 \\ x_1 + x_2 \leq 100 \\ x_1 + x_2 \geq 50 \\ x_1 - d_1^+ + d_1^- = 75 \\ 200x_1 + 100x_2 - d_2^+ + d_2^- = 10000 \\ \frac{d_1^-}{75} \leq D \\ \frac{d_2^+}{1000} \leq D \\ x_j, d_i^+, d_i^- \geq 0,\ j = 1, 2,\ i = 1, \ldots 2 \end{gathered} \tag{3.2}$$

Que lo reescribiremos como:

$$
\begin{gathered}
min\ D \\
s.a.\ x_1 \geq 20 \\
x_1 + x_2 \leq 100 \\
x_1 + x_2 \geq 50 \\
x_1 - d_1^+ + d_1^- = 75 \\
200x_1 + 100x_2 - d_2^+ + d_2^- = 10000 \\
\tfrac{d_1^-}{75} - D \leq 0 \\
\tfrac{d_2^+}{1000} - D \leq 0 \\
x_j, d_i^+, d_i^- \geq 0,\ j = 1, 2,\ i = 1, \ldots 2
\end{gathered}
\tag{3.3}
$$

Si reescribimos las restricciones en forma matricial obtendremos:

$$
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & -1 & 0 & 0 & 0 \\
200 & 100 & 0 & 0 & 1 & -1 & 0 \\
0 & 0 & \frac{1}{75} & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{10000} & -1
\end{pmatrix}
\begin{pmatrix} x_1 \\ x_2 \\ d_1^- \\ d_1^+ \\ d_2^- \\ d_2^+ \\ D \end{pmatrix}
\begin{pmatrix} \geq \\ \leq \\ \geq \\ = \\ = \\ \leq \\ \leq \end{pmatrix}
\begin{pmatrix} 20 \\ 100 \\ 50 \\ 75 \\ 1000 \\ 0 \\ 0 \end{pmatrix}
$$

De nuevo podremos observar que el problema anterior está escrito como un problema de programación lineal y por tanto podremos usar los paquetes de R de programación lineal para resolverlo, lo haremos de tres formas distintas; usando: "solveLP", "lp" y el paquete lpSolveAPI .

En primer lugar con la función solveLP:

```
library(linprog)
library(lpSolve)
#vector función objetivo
objetivo=c(0,0,0,0,0,0,1)

#matriz restricciones
mrestricciones=matrix(c(1,1,1,1,200,0,0,
                        0,1,1,0,100,0,0,
                        0,0,0,1,0,1/75,0,
                        0,0,0,-1,0,0,0,
                        0,0,0,0,1,0,0,
                        0,0,0,0,-1,0,1/10000,
                        0,0,0,0,0,-1,-1),ncol=7)
#vector b , valores restricciones
```

```
brestricciones=c(20,100,50,75,10000,0,0)
#desigualdades
des=c(">=","<=",">=","=","=","<=","<=")
solucion1=solveLP(objetivo,brestricciones,mrestricciones,
                  maximum=FALSE,des,lpSolve=TRUE)
summary(solucion1)
```

```
##
##
## Results of Linear Programming / Linear Optimization
##
## Objective function (Minimum): 0.2
##
## Solution
##      opt
## 1   60.0
## 2    0.0
## 3   15.0
## 4    0.0
## 5    0.0
## 6 2000.0
## 7    0.2
```

```
solucion1$solution
```

```
##      1      2      3      4      5      6      7
##   60.0    0.0   15.0    0.0    0.0 2000.0    0.2
```

```
solucion1$opt
```

```
## [1] 0.2
```

El comando "solución1" nos da la solución obtenida empleando la función "solveLp" de la librería linprog.

En segundo lugar con la función "lp":

```
solucion2=lp("min",objetivo,mrestricciones,des,brestricciones)
solucion2$solution
```

```
## [1]   60.0    0.0   15.0    0.0    0.0 2000.0    0.2
```

Por otro lado la "solución2" nos da la solución obtenida empleando la función "lp".

En tercer lugar emplearemos "lpSolveAPI":

```
library(lpSolveAPI)
#Dimensionalizamos
lprec <- make.lp(7,7)
#Fijamos las columnas
set.column(lprec, 1, c(1,1,1,1,200,0,0))
set.column(lprec, 2, c(0,1,1,0,100,0,0))
set.column(lprec, 3, c( 0,0,0,1,0,1/75,0))
```

```
set.column(lprec, 4, c(  0,0,0,-1,0,0,0))
set.column(lprec, 5, c( 0,0,0,0,1,0,0))
set.column(lprec, 6, c(  0,0,0,0,-1,0,1/10000))
set.column(lprec, 7, c( 0, 0,0,0,0,-1,-1))
#Función objetivo
set.objfn(lprec, c(0,0,0,0,0,0,1) )
#Desigualdades
set.constr.type(lprec,c(">=","<=",">=","=","=","<=","<=") )
#Vector b
set.rhs(lprec, c(20,100,50,75,10000,0,0))
solve(lprec)
```

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

```
get.objective(lprec)
```

```
## [1] 0.2
```

```
get.variables(lprec)
```

```
## [1]    60.0     0.0    15.0     0.0     0.0 2000.0     0.2
```

Finalmente; el objeto "lprec" nos dará la solución obtenida usando la librería lpSolveAPI.

Conclusiones:

Gracias a los comandos: "solucion1$solution", "solucion2$solution" y get.variables(lprec) obtenemos que D=0.2 (la última variable), lo cual implica que tendremos una solución equilibrada, que se desvía un 20 % de los niveles de aspiración de ambos criterios. Las soluciones obtenidas por los tres metodos son idénticas entre sí e iguales a las obtenidas en el capítulo 2.

3.1.4.2. Problema 2 (Ejemplo 2.14)

Para terminar los ejemplos de R veremos la resolución del ejemplo 2.14, el cual resolveremos usando el algoritmo lineal secuencial (igual que estaba resuelto en el capítulo 2). Para ello tenemos que resolver los tres subproblemas siguientes:

Subproblema 1:

$$\begin{gathered} min\ \lambda_1 = d_1^+ + d_2^+ + d_3^+ \\ s.a.\ x_1 + 3x_2 - d_1^+ + d_1^- = 24 \\ 2x_1 + x_2 - d_2^+ + d_2^- = 18 \\ x_1 + x_2 - d_3^+ + d_3^- = 10 \\ x_1, x_2, d_1^+, d_2^+, d_1^-, d_2^-, d_3^+, d_3^- \geq 0 \end{gathered} \tag{3.4}$$

Si reescribimos las restricciones en forma matricial tendremos:

$$\begin{pmatrix} 1 & 3 & -1 & 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & -1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ d_1^+ \\ d_1^- \\ d_2^+ \\ d_2^- \\ d_3^+ \\ d_3^- \end{pmatrix} \begin{pmatrix} = \\ = \\ = \end{pmatrix} \begin{pmatrix} 24 \\ 18 \\ 10 \end{pmatrix}$$

En primer lugar resolveremos con la función solveLP:

```
library(linprog)
library(lpSolve)
#vector función objetivo
objetivo=c(0,0,1,0,1,0,1,0)
#matriz restricciones
mrestricciones=matrix(c(1,2,1,
                        3,1,1,
                        -1,0,0,
                        1,0,0,
                        0,-1,0,
                        0,1,0,
                        0,0,-1,
                        0,0,1),ncol=8)
#vector b , valores restricciones
brestricciones=c(24,18,10)
#desigualdades
des=c("=","=","=")
solucion1=solveLP(objetivo,brestricciones,mrestricciones,maximum=FALSE,
                  des,lpSolve=TRUE)
summary(solucion1)
```

```
##
##
## Results of Linear Programming / Linear Optimization
##
## Objective function (Minimum): 0
##
## Solution
##   opt
## 1   0
## 2   8
## 3   0
## 4   0
## 5   0
## 6  10
## 7   0
## 8   2
```

```
solucion1$solution
```

```
##  1  2  3  4  5  6  7  8
##  0  8  0  0  0 10  0  2
```

```
solucion1$opt
```

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

Ahora lo resolveremos con la función lp del paquete lpSolve:

```
solucion2=lp("min",objetivo,mrestricciones,des,brestricciones)
solucion2$solution
```

```
## [1]  0  8  0  0  0 10  0  2
```

Y por último con el paquete lpSolveAPI:

```
library(lpSolveAPI)
#Dimensionalizamos
lprec <- make.lp(3, 8)
#Fijamos las columnas
set.column(lprec, 1, c(1,2,1))
set.column(lprec, 2, c( 3,1,1))
set.column(lprec, 3, c(-1,0,0))
set.column(lprec, 4, c( 1,0,0))
set.column(lprec, 5, c(0,-1,0))
set.column(lprec, 6, c( 0,1,0))
set.column(lprec, 7, c(0,0,-1))
set.column(lprec, 8, c( 0,0,1))
#Función objetivo
set.objfn(lprec, c(0,0,1,0,1,0,1,0)  )
#Desigualdades
 set.constr.type(lprec, c("=","=","="))
#Vector b
 set.rhs(lprec,c(24,18,10))
solve(lprec)
```

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

```
get.objective(lprec)
```

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

```
get.variables(lprec)
```

```
## [1]  0  8  0  0  0 10  0  2
```

Gracias a los comandos "solucion1$opt" y "get.objective(lprec)" podemos observar que $\lambda_1^* = 0$ como habíamos visto en el capítulo 2.

Subproblema 2: Tendremos que resolver el siguiente problema:

$$
\begin{aligned}
& min\ \lambda_2 = d_4^- \\
& s.a.\ x_1 + 3x_2 - d_1^+ + d_1^- = 24 \\
& 2x_1 + x_2 - d_2^+ + d_2^- = 18 \\
& x_1 + x_2 - d_3^+ + d_3^- = 10 \\
& 2x_1 + 3x_2 - d_4^+ + d_4^- = 21 \\
& d_1^+ + d_2^+ + d_3^+ = 0 \\
& x_j, d_i^+, d_i^- \geq 0,\ j = 1, 2,\ i = 1, \ldots 4
\end{aligned}
\tag{3.5}
$$

Si reescribimos las restricciones en forma matricial tendremos:

$$
\begin{pmatrix}
1 & 3 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\
2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
x_1 \\ x_2 \\ d_1^+ \\ d_1^- \\ d_2^+ \\ d_2^- \\ d_3^+ \\ d_3^- \\ d_4^+ \\ d_4^-
\end{pmatrix}
\begin{pmatrix} = \\ = \\ = \\ = \\ = \end{pmatrix}
\begin{pmatrix} 24 \\ 18 \\ 10 \\ 21 \\ 0 \end{pmatrix}
$$

En primer lugar resolveremos con la función solveLP:

```
library(linprog)
library(lpSolve)
#vector función objetivo
objetivo=c(0,0,0,0,0,0,0,0,0,1)

#matriz restricciones
mrestricciones=matrix(c(1,2,1,2,0,
                        3,1,1,3,0,
                        -1,0,0,0,1,
                        1,0,0,0,0,
                        0,-1,0,0,1,
                        0,1,0,0,0,
                        0,0,-1,0,1,
                        0,0,1,0,0,
                        0,0,0,-1,0,
                        0,0,0,1,0
                        ),ncol=10)
#vector b , valores restricciones
brestricciones=c(24,18,10,21,0)
```

```
#desigualdades
des=c("=","=","=","=","=")
solucion1=solveLP(objetivo,brestricciones,mrestricciones,maximum=FALSE,
                  des,lpSolve=TRUE)
summary(solucion1)
```

```
##
##
## Results of Linear Programming / Linear Optimization
##
## Objective function (Minimum): 0
##
## Solution
##    opt
## 1    0
## 2    7
## 3    0
## 4    3
## 5    0
## 6   11
## 7    0
## 8    3
## 9    0
## 10   0
```

```
solucion1$solution
```

```
##  1  2  3  4  5  6  7  8  9 10
##  0  7  0  3  0 11  0  3  0  0
```

```
solucion1$opt
```

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

Ahora lo resolveremos con la función lp del paquete lpSolve:

```
solucion2=lp("min",objetivo,mrestricciones,des,brestricciones)
solucion2$solution
```

```
##  [1]  0  7  0  3  0 11  0  3  0  0
```

Y por último con el paquete lpSolveAPI:

```
library(lpSolveAPI)
#Dimensionalizamos
lprec <- make.lp(5, 10)
#Fijamos las columnas
set.column(lprec, 1, c(1,2,1,2,0))
set.column(lprec, 2, c( 3,1,1,3,0))
set.column(lprec, 3, c(-1,0,0,0,1))
set.column(lprec, 4, c( 1,0,0,0,0))
set.column(lprec, 5, c(0,-1,0,0,1))
```

```
set.column(lprec, 6, c( 0,1,0,0,0))
set.column(lprec, 7, c(0,0,-1,0,1))
set.column(lprec, 8, c( 0,0,1,0,0))
set.column(lprec, 9, c(0,0,0,-1,0))
set.column(lprec, 10, c(0, 0,0,1,0))
#Función objetivo
set.objfn(lprec, c(0,0,0,0,0,0,0,0,0,1)   )
#Desigualdades
 set.constr.type(lprec, c("=","=","=","=","="))
#Vector b
 set.rhs(lprec,c(24,18,10,21,0))
solve(lprec)
```

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

```
get.objective(lprec)
```

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

```
get.variables(lprec)
```

```
##  [1]  0  7  0  3  0 11  0  3  0  0
```

De nuevo gracias a los comandos "solucion1$opt" y "get.objective(lprec)" observamos que $\lambda_2^* = 0$ como vimos en el capítulo 2 gracias a la información de los libros.

Finalmente, tendremos que resolver un tercer problema, que nos dará la solución del problema 2.12

Subproblema 3

$$
\begin{aligned}
&min\ \lambda_3 = d_5^+ \\
&s.a.\ x_1 + 3x_2 - d_1^+ + d_1^- = 24 \\
&2x_1 + x_2 - d_2^+ + d_2^- = 18 \\
&x_1 + x_2 - d_3^+ + d_3^- = 10 \\
&2x_1 + 3x_2 - d_4^+ + d_4^- = 21 \\
&4x_1 + 2x_2 - d_5^+ + d_5^- = 12 \\
&d_1^+ + d_2^+ + d_3^+ = 0, d_4^- = 0 \\
&x_j, d_i^+, d_i^- \geq 0,\ j = 1, 2,\ i = 1, \ldots 5
\end{aligned}
\tag{3.6}
$$

Si reescribimos las restricciones en forma matricial obtendremos:

$$\begin{pmatrix} 1 & 3 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 4 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ d_1^+ \\ d_1^- \\ d_2^+ \\ d_2^- \\ d_3^+ \\ d_3^- \\ d_4^+ \\ d_4^- \\ d_5^+ \\ d_5^- \end{pmatrix} \begin{pmatrix} = \\ = \\ = \\ = \\ = \\ = \\ = \end{pmatrix} \begin{pmatrix} 24 \\ 18 \\ 10 \\ 21 \\ 12 \\ 0 \\ 0 \end{pmatrix}$$

En primer lugar resolveremos con la función solveLP:

```
library(linprog)
library(lpSolve)
#vector función objetivo
objetivo=c(0,0,0,0,0,0,0,0,0,0,1,0)

#matriz restricciones
mrestricciones=matrix(c(1,2,1,2,4,0,0,
                        3,1,1,3,2,0,0,
                        -1,0,0,0,0,1,0,
                        1,0,0,0,0,0,0,
                        0,-1,0,0,0,1,0,
                        0,1,0,0,0,0,0,
                        0,0,-1,0,0,1,0,
                        0,0,1,0,0,0,0,
                        0,0,0,-1,0,0,0,
                        0,0,0,1,0,0,1,
                        0,0,0,0,-1,0,0,
                        0,0,0,0,1,0,0
                        ),ncol=12)
#vector b , valores restricciones
brestricciones=c(24,18,10,21,12,0,0)
#desigualdades
des=c("=","=","=","=","=","=","=")
solucion1=solveLP(objetivo,brestricciones,mrestricciones,maximum=FALSE,
                  des,lpSolve=TRUE)
summary(solucion1)
```

```
##
##
## Results of Linear Programming / Linear Optimization
##
## Objective function (Minimum): 2
##
## Solution
```

```
##    opt
## 1    0
## 2    7
## 3    0
## 4    3
## 5    0
## 6   11
## 7    0
## 8    3
## 9    0
## 10   0
## 11   2
## 12   0
```

```
solucion1$solution
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12
##  0  7  0  3  0 11  0  3  0  0  2  0
```

```
solucion1$opt
```

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

Ahora lo resolveremos con la función lp del paquete lpSolve:

```
solucion2=lp("min",objetivo,mrestricciones,des,brestricciones)
solucion2$solution
```

```
##  [1]  0  7  0  3  0 11  0  3  0  0  2  0
```

Y por último con el paquete lpSolveAPI:

```
library(lpSolveAPI)
#Dimensionalizamos
lprec <- make.lp(7, 12)
#Fijamos las columnas
set.column(lprec, 1, c(1,2,1,2,4,0,0))
set.column(lprec, 2, c( 3,1,1,3,2,0,0))
set.column(lprec, 3, c(-1,0,0,0,0,1,0))
set.column(lprec, 4, c( 1,0,0,0,0,0,0))
set.column(lprec, 5, c(0,-1,0,0,0,1,0))
set.column(lprec, 6, c( 0,1,0,0,0,0,0))
set.column(lprec, 7, c(0,0,-1,0,0,1,0))
set.column(lprec, 8, c( 0,0,1,0,0,0,0))
set.column(lprec, 9, c(0,0,0,-1,0,0,0))
set.column(lprec, 10, c(0,0,0,1,0,0,1))
set.column(lprec, 11, c(0,0,0,0,-1,0,0))
set.column(lprec, 12, c(0,0,0,0,1,0,0))
#Función objetivo
set.objfn(lprec, c(0,0,0,0,0,0,0,0,0,0,1,0)   )
#Desigualdades
 set.constr.type(lprec, c("=","=","=","=","=","=","="))
```

```
#Vector b
 set.rhs(lprec,c(24,18,10,21,12,0,0))
solve(lprec)
```

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

```
get.objective(lprec)
```

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

```
get.variables(lprec)
```

```
##  [1]  0  7  0  3  0 11  0  3  0  0  2  0
```

- El comando "solución1" nos da la solución obtenida empleando la función "solveLp" de la librería linprog.
- Por otro lado la "solución2" nos da la solución obtenida empleando la función "lp" de lpSolve.
- Finalmente; la función "lprec" nos dará la solución obtenida usando la librería lpSolveAPI.

Conclusiones:

De nuevo podemos observar que $\lambda_2^* = 3$ como vimos en el capítulo 2 gracias a la información de los libros y por tanto la solución óptima del problema 2.12 será $x_1^* = 0$ y $x_2^* = 7$ (esto lo podemos corroborar gracias a las salidas de los comandos: "solucion1$solution", "solucion2$solution" y get.variables(lprec)). Además podemos ver que el valor de la función objetivo será 2.

3.2. Introducción AMPL

La siguiente imagen se corresponde con el logo de AMPL cuya página web oficial es https://www.ampl.com.

Figura 3.2: Logo AMPL

AMPL (A Mathematical Programming Language) es un lenguaje de programación algebraica (AML) para describir y solucionar problemas de gran complejidad para computación matemática de gran escala. Fue desarrollado por Robert Fourer, David Gay, y Brian Kernighan en los Laboratorios Bell. AMPL soporta docenas de solvers, tanto de código abierto como software comercial, incluyendo CBC, CPLEX, FortMP, Gurobi, MINOS, IPOPT, SNOPT, KNITRO, y LGO.

Una ventaja de AMPL es la semejanza de su sintaxis a la notación matemática de problemas de optimización. Esto permite una definición muy concisa y legible de problemas en el ámbito de optimización.

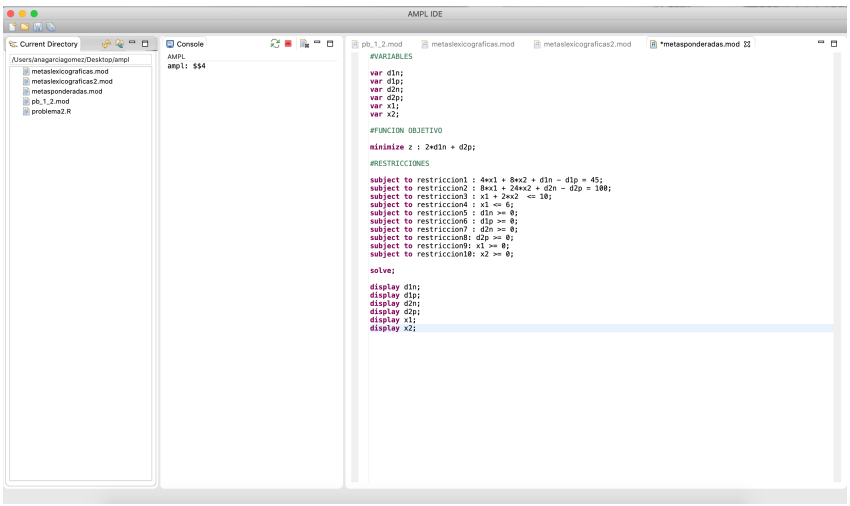

Figura 3.3: Captura aplicación AMPL

3.2.1. Modelo general

A continuación se explica brevemente como usar AMPL para resolver problemas de programación lineal (véase [12]).

- **Paso 1**: Fijar las variables (suponemos que el problema tiene n variables)

$$var\ x_1 >= 0;$$

$$\vdots$$

$$var\ x_n >= 0;$$

- **Paso 2**: Función objetivo; tendrá una de las siguientes formas
  - maximize z: $c_1 * x_1 +, \ldots, +c_n * x_n$;
  - minimize z: $c_1 * x_1 +, \ldots, +c_n * x_n$;
- **Paso 3**: Definir las restricciones (suponemos que tenemos m)
  - subject to restriccion1 : $a_{11} * x_1 +, \ldots, +a_{1n} * x_n (<=, >= \text{ ó } =) b_1$;

    $$\vdots$$

  - subject to restriccionm : $a_{m1} * x_1 +, \ldots, +a_{mn} * x_n (<=, >= \text{ ó } =) b_m$;
- **Paso 4**: Resolución
  - option solve cplex;
  - solve; (da el valor de la función objetivo)
  - display (alguna variable); (te devuelve el valor de la variable)

3.2.2. Ejemplos AMPL

En esta sección resolveremos el ejemplo 2.7 con AMPL usando la forma descrita anteriormente y comprobaremos que la solución obtenida coincide con la solución citada en el capítulo 2.

3.2.2.1. Problema 3 (Ejemplo 2.7)

Para finalizar este capítulo resolveremos el ejemplo 2.7 con AMPL; usando las modificaciones siguientes:

- Metas ponderadas
- Metas lexicográficas

3.2.2.1.1. Metas ponderadas

Al igual que en el capítulo anterior supondremos que la meta de exposición es dos veces más importante que la meta de presupuesto obteniendo así el siguiente problema:

$$
\begin{gathered}
Minimizar\ 2d_1^- + d_2^+ \\
s.a.\ 4x_1 + 8x_2 + d_1^- - d_1^+ = 45\ (Meta\ de\ exposición) \\
8x_1 + 24x_2 + d_2^- - d_2^+ = 100\ (Meta\ de\ presupuesto) \\
x_1 + 2x_2 \leq 10\ (Límite\ de\ presonal) \\
x_1 \leq 6\ (Límite\ de\ radio) \\
x_j, d_i^+, d_i^- \geq 0,\ j = 1, 2,\ i = 1, \ldots 2
\end{gathered}
\tag{3.7}
$$

Una vez que el ejemplo ya tiene forma de problema de programación lineal usaremos Ampl para resolverlo. Teniendo en cuenta que denotaremos d_1^- como d1n, d_1^+ como d1p, d_2^- como d2n y d_2^+ como d2p y que el hecho de que las variables sean positivas lo hemos puesto como restricciones; el código será el siguiente:

```
#Variables
  var d1n;
  var d1p;
  var d2n;
  var d2p;
  var x1;
  var x2;

#Funcion objetivo
  minimize z : 2*d1n + d2p;

#Restricciones
  subject to restriccion1 : 4*x1 + 8*x2 + d1n - d1p = 45;
  subject to restriccion2 : 8*x1 + 24*x2 + d2n - d2p = 100;
  subject to restriccion3 : x1 + 2*x2  <= 10;
  subject to restriccion4 : x1 <= 6;
  subject to restriccion5 : d1n >= 0;
  subject to restriccion6 : d1p >= 0;
  subject to restriccion7 : d2n >= 0;
  subject to restriccion8: d2p >= 0;
  subject to restriccion9: x1 >= 0;
  subject to restriccion10: x2 >= 0;

 #Resolucion
   option solve cplex;
   solve;
   display d1n;
   display d1p;
   display d2n;
   display d2p;
```

```
display x1;
display x2;
```

Cuya solución resolviendo con AMPL será:

```
CPLEX 12.10.0.0: optimal solution; objective 10
3 dual simplex iterations (0 in phase I)
d1n = 5

d1p = 0

d2n = 4

d2p = 0

x1 = 6

x2 = 2

z = 10
```

Este problema también podemos resolverlo de una segunda forma; separando en un fichero ".mod" con la formulación del problema y otro fichero ".run" con los comandos de resolución:

Fichero "ponderadas1.mod":

```
param n := 2;
param s :=2;
param l:=2;

set PRODUCTOS := 1..n;
set PRODUCTOS2 :=1..s;
set PRODUCTOS3 :=1..l;


var x {j in PRODUCTOS} >=0;
var dmenos {i in PRODUCTOS2} >=0;
var dmas {k in PRODUCTOS3} >=0;

minimize z:
2*dmenos[1] + dmas[2];

subject to restriccionigualdad1:
4*x[1] + 8*x[2] + dmenos[1] - dmas[1] = 45;

subject to restriccionigualdad2:
8*x[1] + 24*x[2] + dmenos[2] - dmas[2] = 100;

subject to restriccion1:
x[1] + 2*x[2] <= 10;
```

```
subject to restriccion2:
x[1] <= 6;
```

Fichero "ponderadas1.run":

```
reset;
option solver cplex;
model ponderadas1.mod ;
solve;
display x;
display dmenos;
display dmas;
display z;
```

Cuya solución resolviendo con AMPL será:

```
ampl: include '/Users/anagarciagomez/Desktop/correcciones/ponderadas1.run';
CPLEX 12.10.0.0: optimal solution; objective 10
3 dual simplex iterations (0 in phase I)
x [*] :=
1  6
2  2
;

dmenos [*] :=
1  5
2  4
;

dmas [*] :=
1  0
2  0
;

z = 10
```

Conclusiones:

La solución que nos dará AMPL (en ambos casos) se corresponderá a la segunda solución propuesta en el capítulo dos, en la que si tenemos en cuenta la optimización. Esta solución como vemos es $d_1^- = 5$ significa que la meta de exposición (de al menos 45 millones de personas) falla por 5 millones de personas. Por otra parte, la meta de presupuesto (de no exceder 100,000 dólares) no se viola porque $d_2^+ = 0$. Además el valor de la función objetivo será 10.

3.2.2.1.2. Metas lexicográficas

Al igual que en el capítulo anterior supondremos que tendremos las prioridades $G_1 > G_2$. Por tanto; usando el método de las metas lexicográficas tendremos que resolver dos

subproblemas:

Subproblema 1:

$$\begin{gathered} Minimizar\ G_1^- = d_1^- \\ s.a.\ 4x_1 + 8x_2 + d_1^- - d_1^+ = 45\ (Meta\ de\ exposición) \\ 8x_1 + 24x_2 + d_2^- - d_2^+ = 100\ (Meta\ de\ presupuesto) \\ x_1 + 2x_2 \leq 10\ (Límite\ de\ presonal) \\ x_1 \leq 6\ (Límite\ de\ radio) \\ x_j, d_i^+, d_i^- \geq 0,\ j = 1, 2,\ i = 1, \ldots 2 \end{gathered} \tag{3.8}$$

De nuevo como el ejemplo ya tiene forma de problema de programación lineal usaremos AMPL para resolverlo. Teniendo en cuenta que denotaremos d_1^- como d1n, d_1^+ como d1p, d_2^- como d2n y d_2^+ como d2p; el código será el siguiente:

```
# Variables
   var d1n;
   var d1p;
   var d2n;
   var d2p;
   var x1;
   var x2;

#Funcion objetivo
   minimize z : d1n;

#Restricciones
  subject to restriccion1 : 4*x1 + 8*x2 + d1n - d1p = 45;
  subject to restriccion2 : 8*x1 + 24*x2 + d2n - d2p = 100;
  subject to restriccion3 : x1 + 2*x2  <= 10;
  subject to restriccion4 : x1 <= 6;
  subject to restriccion5 : d1n >= 0;
  subject to restriccion6 : d1p >= 0;
  subject to restriccion7 : d2n >= 0;
  subject to restriccion8: d2p >= 0;
  subject to restriccion9: x1 >= 0;
  subject to restriccion10: x2 >= 0;

#Resolucion
 option solver cplex;
 solve;
 display d1n;
 display d1p;
```

```
  display d2n;
  display d2p;
  display x1;
  display x2;
```

La solución del problema será:

```
CPLEX 12.10.0.0: optimal solution; objective 5
0 dual simplex iterations (0 in phase I)
d1n = 5

d1p = 0

d2n = 0

d2p = 20

x1 = 0

x2 = 5

z = 5
```

Conclusiones:

En dicho caso la solución de x_1 y x_2 no coincide con la nombrada en el capítulo 2, aunque sí la de $d_1^- = 5$, esto se debe a la existencia de infinitas soluciones. Además el valor de la función objetivo será cinco.

Subproblema 2: Para finalizar tendremos que resolver el siguiente problema

$$
\begin{gathered}
Minimizar\ G_2^+ = d_2^+ \\
s.a.\ 4x_1 + 8x_2 + d_1^- - d_1^+ = 45\ (Meta\ de\ exposición) \\
8x_1 + 24x_2 + d_2^- - d_2^+ = 100\ (Meta\ de\ presupuesto) \\
x_1 + 2x_2 \leq 10\ (Límite\ de\ presonal) \\
x_1 \leq 6\ (Límite\ de\ radio) \\
d_1^- = 5 \\
x_j, d_i^+, d_i^- \geq 0,\ j = 1, 2,\ i = 1, \ldots 2
\end{gathered}
\tag{3.9}
$$

Usando AMPL con las notaciones anteriores; tendremos el código:

```
#Variables
  var d1n;
  var d1p;
  var d2n;
```

```
    var d2p;
    var x1;
    var x2;

  #Funcion objetivo
     minimize z : d2p ;

  #Restricciones
     subject to restriccion1 : 4*x1 + 8*x2 + d1n - d1p = 45;
     subject to restriccion2 : 8*x1 + 24*x2 + d2n - d2p = 100;
     subject to restriccion3 : x1 + 2*x2  <= 10;
     subject to restriccion4 : x1 <= 6;
     subject to restriccion5 : d1n >= 0;
     subject to restriccion6 : d1p >= 0;
     subject to restriccion7 : d2n >= 0;
     subject to restriccion8: d2p >= 0;
     subject to restriccion9: x1 >= 0;
     subject to restriccion10: x2 >= 0;
     subject to restriccion11: d1n = 5;

   #Resolucion
    option solver cplex;
    solve;
    display d1n;
    display d1p;
    display d2n;
    display d2p;
    display x1;
    display x2;
    display z;
```

La solución del problema será:

```
CPLEX 12.10.0.0: optimal solution; objective 0
0 dual simplex iterations (0 in phase I)
d1n = 5

d1p = 0

d2n = 4

d2p = 0

x1 = 6

x2 = 2
```

Conclusión:

Aquí ya observamos, que la solución coincide con la del capítulo 2, $x_1 = 6$ y $x_2 = 2$; con $d_2^- = 0$. Además esta solución será la solución de nuestro problema original.

Capítulo 4

Aplicaciones

En este último capítulo y para así cerrar este trabajo, veremos tres aplicaciones de la programación por metas en la vida cotidiana; para así conocer la utilidad y el alcance del tema que hemos tratado en este trabajo.

En cada una de las aplicaciones reales, se va a hacer una introducción al problema, una descripción del problema de programación por metas y se finalizará con una breve conclusión que obtuvieron los autores de los artículos.

4.1. Distribución de servicios entre empresas operadoras del sistema de transporte masivo (SITM)

A continuación vamos a describir esta aplicación real de las programación por metas que se recoge en el artículo [9].

Figura 4.1: Megabus S.A.

4.1.1. Desarrollo

En la planeación y operación de los sistemas de transporte masivo[1] existe una jerarquía de problemas interrelacionados entre sí, que abarcan desde el nivel estratégico hasta el operacional. Dichos problemas son:

- El diseño de la red de ruta.
- La determinación de frecuencias y horarios.
- La distribución de tablas entre empresas operadoras (cada tabla está compuesta por ciclos que debe realizar el bus articulado desde un origen i hasta un destino j, resaltando que cada ciclo se conoce como un servicio.

[1]Imagen obtenida de https://www.megabus.gov.co

- La asignación de flota a cada servicio.
- La asignación de personal y recursos disponibles.

El modelo que se estudia en este artículo se enfoca en la distribución de tablas entre empresas operadoras del SITM, el cual surge como respuesta para regular los problemas anteriores en Promasivo S.A. empresa operadora de Megabús S.A. En lo que respecta a la distribución de las tablas, las empresas operadoras requieren de una herramienta de decisión que les permita obtener una asignación óptima, de tal forma, que los entes participantes queden satisfechos con los resultados en términos de minimización de costos y maximización de utilidades.

Por ello, se usará un modelo de programación por metas donde se quiere:

- Minimizar el tiempo de operación de cada vehículo.
- Minimizar el kilometraje vacío de los vehículos (tiempo que recorren sin transportar personas).
- Maximizar el acercamiento al porcentaje del kilometraje asignado para el respectivo operador.
- Cumplimiento del número de tablas asignadas.

4.1.2. Descripción del modelo de programación por metas

Suponemos que tenemos una tabla con 32 elementos distintos; a partir de ella planteamos el problema de programación por metas.

- **Definición de Variables**:

$$X_i = \begin{cases} 0 & \text{No se asigna a Promasivo} \\ 1 & \text{Se asigna a Promasivo} \end{cases}$$

$$Y_i = \begin{cases} 0 & \text{No se asigna a Integra} \\ 1 & \text{Se asigna a Integra} \end{cases}$$

Donde i= 1,2,. . . ,32

- **Metas**:
  - 1. Minimizar kilometraje en vacío (kmv conocido)

$$min\ z = \sum_{i=1}^{32} Kmv_i X_i$$

  - 2. Minimizar tiempos de operación de vehículos (t conocido)

$$min\ z = \sum_{i=1}^{32} t_i X_i$$

  - 3. Minimizar tablas partidas (Se entiende por tabla partida un servicio discontinuo, es decir que el articulado realizará un parte en la jornada de la mañana y el resto en otra jornada “tarde o noche”, esto ocasiona

un aumento en el kmv y por ende disminución en las utilidades que proporciona una tabla normal). Suponemos que conocemos las tablas que son partidas y las que no. Si le asignamos 1 a las tablas partidas y 0.5 a las completas obtendremos:

$$Min\ Z = 1(X_1 + X_4 + X_6 + X_9 + X_{12} + X_{15} + X_{17} + X_{20} + X_{22}) +$$
$$+0.5(X_2 + X_3 + X_5 + X_7 + X_8 + X_{10} + X_{11} + X_{13} + X_{14} + X_{16} + X_{18} + X_{19} + X_{21} +$$
$$+X_{22} + X_{23} + X_{24} + X_{25} + X_{26} + X_{27} + X_{28} + X_{29} + X_{30} + X_{31} + X_{32})$$

- **Restricciones**: Se impondrán restricciones de los siguientes tipos:
  - 1. Porcentaje de Kilometraje Asignado. (Kmc)
  - 2. Número de tablas asignadas.
  - 3. No Negatividad.
  - X_i, Y_i variables binarias.

4.1.3. Conclusiones

La programación por metas es un modelo de optimización altamente efectivo cuando se busca obtener resultados factibles en determinadas operaciones que involucren varios objetivos a la vez.

Se diseñó un modelo de Programación con Metas Múltiples, en las cuales se puede jerarquizar y priorizar las diferentes metas intervinientes en el modelo. Este modelo se implementó en el sistema de transporte masivo Megabús S.A. y se logró optimizar la distribución de los 32 servicios para un día normal de operación entre las dos empresas operadoras del sistema logrando para cada una el cumplimiento de sus objetivos.

4.2. Un modelo de programación por metas para el plan de producción de un hospital del servicio vasco de salud

A continuación vamos a describir esta aplicación real de las programación por metas que se recoge en el artículo [7].

Figura 4.2: Hospital Vasco

4.2.1. Introducción

En el caso del País Vasco, cada hospital[2] de Osakidetza (Servicio Vasco de Salud), en colaboración con el Gobierno Vasco, debe desarrollar un Plan de Gestión anual.

En general en España, para medir la producción de un hospital se utilizan los **GRDs** (grupos relacionados por el diagnóstico)

El Plan de Producción de un hospital puede ser considerado como un problema de selección del nivel de actividad de los diversos GRDs (case mix).

La determinación del Plan de Producción de un hospital es un problema complejo en el que es preciso considerar diversos objetivos que entran en conflicto, por lo que no pueden ser satisfechos simultáneamente, de manera que normalmente los responsables sanitarios se conforman con alcanzar ciertos niveles de logro en cada objetivo. Por lo tanto, el Plan de Producción se presta a ser formulado como un problema de programación lineal por metas, lo que permitirá al decisor evaluar la actividad del hospital en relación con metas que proporcionen servicios de calidad y que tengan en cuenta la capacidad y los intereses de los diferentes Servicios Médicos.

4.2.2. Descripción modelo de programación por metas

Desarrollaremos un modelo de programación por **metas lexicográficas**; es decir; clasificaremos previamente los objetivos según un orden de prioridad. Estos niveles serán:

1. Puesto que se trata de establecer un Plan de Producción pactado con el Gobierno el primer nivel de prioridad será ajustarse al presupuesto asignado.

[2]Imagen obtenida de https://www.elconfidencial.com/empresas/2020-05-08/choque-gobierno-vasco-hospitales-privados-pago-atencion-covid_2584964/

2. Como segundo nivel de prioridad el decisor, establece que los niveles de actividad en cada GRD no deben ser inferiores a los correspondientes niveles de aspiración establecidos para el próximo ejercicio.

Además se desea que la carga de actividad (o peso de complejidad) asumida por cada servicio médico sea cuanto menos igual a la históricamente realizada.

3. Por último en el tercer nivel de prioridad se pretende que algunos indicadores de actividad del conjunto del hospital se mantengan en determinados valores, en concreto que la estancia media no exceda del nivel de aspiración establecido por el decisor y que el peso medio de complejidad asumido por hospital no sea inferior al establecido como nivel de aspiración.

Consideraremos las siguientes restricciones:

1. El número de estancias disponibles.
2. El tiempo de quirófano total.
3. El tiempo del personal médico disponible por cada Servicio Médico.

De manera general, previa consulta con los responsables de los Servicios Médicos, los niveles de aspiración de las diferentes metas los establece la Dirección del hospital conciliando las necesidades de la población y los recursos limitados del hospital. Para ello se tienen en cuenta, entre otras cosas, los datos históricos, por ejemplo aumentando el nivel de aspiración de las actividades con mayor lista de espera, o las situaciones que alteren el contexto de periodos anteriores, como la adquisición de tecnología que mejore la eficiencia en una determinada actividad.

4.2.3. Conclusiones

El modelo expuesto en el apartado anterior se llevó a cabo en un pequeño hospital del país vasco donde se empleó el **método lineal secuencial** para su resolución.

El número de variables de decisión dependió del número de servicios seleccionados del hospital; en este caso fueron 8 (Medicina Interna, Cardiología, Neumología, Cirugía General, Oftalmología, Otorrinolaringología, Traumatología y Urología); siendo estas variables x_{iGRDn} número de procesos del GRD n, determinado por el modelo, realizados por el Servicio Médico i.

Finalmente se pudo comprobar su eficacia, obteniéndose que el modelo permitió mejorar el análisis de la eficiencia en el uso de los recursos del hospital, como por ejemplo, la posibilidad de mejorar el rendimiento de los quirófanos, el adaptar los recursos humanos a las necesidades de atención sanitaria, el desarrollar un estudio de las posibilidades de reducción de las listas de espera con los recursos actuales y en general la mejora de los resultados del hospital.

4.3. Evolución de un modelo de programación por metas en el contexto forestal cubano

A continuación vamos a describir esta aplicación real de la programación por metas que se recoge en el artículo [8].

Figura 4.3: Bosque Cubano

4.3.1. Introducción

Los bosques cubanos[3] sufrieron una explotación indiscriminada en el pasado. A pesar de los esfuerzos realizados con los planes de reforestación, y el establecimiento de plantaciones, la estructura por edades de estos bosques está muy desequilibrada, lo que dificulta un flujo estable de bienes y servicios. En este ejemplo se presenta la evolución de un modelo de planificación forestal con criterios múltiples, aplicado a una plantación de Pinus caribaea de la Empresa Forestal Integral Pinar del Río, Cuba, con el objetivo de lograr una estructura equilibrada de edades al finalizar el horizonte de planificación y con ello garantizar una estabilidad en el flujo de bienes y servicios sujeto a restricciones de área, volumen y manejo en general.

4.3.2. Descripción del modelo de programación por metas

En este ejemplo elaboraremos un problema de programación por **metas lexicográficas**.

- **Metas**:
  - **Metas para la superficie a regenerar de equilibrio**

    Con el fin de estabilizar los niveles de producción **(Premisa 1)**, en cada periodo se ha de intentar mantener la superficie en la que se aplica la tala rasa en niveles próximos a lo silvícola y ecológicamente aceptable, evitando que se supere este nivel.
  - **Metas para el control de la posibilidad volumen**

    El volumen que se puede extraer sin riesgos de degradación es el que la plantación sea capaz de incrementar en el periodo correspondiente. Así, establecemos la meta que trata de mantener los niveles de extracción en ese límite, tratando de que no se sobrepase **(Premisa 2)**.

[3]Imagen obtenida de http://www.acn.cu/medio-ambiente/66606-cuenta-cuba-con-un-indice-de-boscosidad-del-31-66-por-ciento-de-sus-areas/

- **Metas para el control de la edad de tala**

  La tala de árboles jóvenes significa un sacrificio en todos los aspectos, económicos, ecológicos y silvícolas. No obstante, y debido a las características de nuestro caso, hemos tenido que admitir la posibilidad de aplicar tala rasa a cualquier edad si fuera necesario para el propósito final de ordenar el bosque. Ahora bien, lo deseable sería que ello no ocurriera y por esta razón definimos la meta **(Premisa 3)**.

- **Metas para el control del VAN**

  El nivel de aspiración de esta meta es un indicador económico de la empresa que se debe cumplir y si fuera posible sobrepasar, aunque lo más importante será lograr una estabilización a largo plazo de ese valor **(Premisa 4)**.

- **Restricciones**

Las metas que acabamos de detallar constituyen un conjunto de restricciones blandas o débiles, ya que el decisor desea que se alcancen, pero un punto que no las satisfaga no se considera no admisible para nuestro problema. Para que un punto sea admisible, en nuestro contexto, ha de cumplir otras restricciones que se conocen como duras o fuertes y que son las condiciones técnicas que, bajo ningún concepto, pueden ser violadas. Estas restricciones son las siguientes.

- Restricciones de control de tratamientos por grupo de edad y por índice de sitio.
- Restricciones para el control del límite inferior de la superficie de tala.
- Restricciones para el control del límite inferior del Valor Actualizado Neto (VAN).

4.3.3. Conclusiones

Desarrollando el modelo anterior para un modelo concreto se logró obtener soluciones que permitieron determinar la superficie a talar para cada índice de sitio en cada período, se maximiza el beneficio sin deterioro del ecosistema y se asegura el equilibrio por edades en la plantación al finalizar la planificación, con satisfacción plena de los deseos del decisor, es decir, se resuelve eficientemente el problema económico y ecológico en cuestión.

Bibliografía

[1] Allaire, J., Xie, Y., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., Wickham, H., Cheng, J., Chang, W., & Iannone, R. (2021). *rmarkdown: Dynamic Documents for R*. R package version 2.7.
URL https://CRAN.R-project.org/package=rmarkdown

[2] Berkelaar, M. (2020). *Package lpSolve*. R package version 5.5.2.0-17.7.
URL https://cran.r-project.org/web/packages/lpSolve/lpSolve.pdf

[3] Caro Merchante, A. (2000). Programación lineal.
URL http://recursostic.educacion.es/descartes/web/Descartes1/Bach_HCS_1/Programacion_lineal/Pl_historia.htm

[4] Departamento de Matemáticas UNLP (s.f.). Capítulo 4 método del simplex.
URL http://www.mate.unlp.edu.ar/practicas/66_13_0804200912835.pdf

[5] Gibrán García C. (2017). Programación por metas: explicación y ejemplo.
URL https://naps.com.mx/blog/programacion-por-metas-explicacion-y-ejemplo/

[6] Henningsen, A. (2012). *linprog: Linear Programming / Optimization*. R package version 0.9-2.
URL http://linprog.r-forge.r-project.org/

[7] Jiménez, M., Rivas, J. A., & Zubia, M. (2005). Un modelo de programación por metas para el plan de producción de un hospital del servicio vasco de salud. *Cuadernos del CIMBAGE*, (7), 1–24.

[8] León, M., Hernández, M., Gómez, T., Guelmes, J., Molina, J., & Caballero, R. (2013). Evolución de un modelo de programación por metas en el contexto forestal cubano. *Investigación Operacional*, *29*(2), 130–139.

[9] Lopez, J. F., Fernández Henao, S., & Morales, M. M. (2007). Aplicación de la programación por metas en la distribución de servicios entre empresas operadoras del sistema de transporte masivo. *Scientia et technica*, *13*(37), 339–343.

[10] lp_solve, Konis, K., & Schwendinger, F. (2020). *lpSolveAPI: R Interface to lp_solve Version 5.5.2.0*. R package version 5.5.2.0-17.7.
URL https://CRAN.R-project.org/package=lpSolveAPI

[11] Lugo Marín, J. (2013). Investigación de operaciones: Programación de metas y objetivos.
URL https://es.slideshare.net/juanlugomarin/programacion-de-metas-y-objetivos-16646467

[12] Luque Calvo, P. L. (2010). Lenguaje ampl.
URL http://destio.us.es/calvo/descargas/ampl2a_modimp_subidoEV2.pdf

[13] Luque Calvo, P. L. (2017). *Escribir un Trabajo Fin de Estudios con R Markdown*.
URL http://destio.us.es/calvo

[14] R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.
URL https://www.R-project.org/

[15] RStudio Team (2015). *RStudio: Integrated Development Environment for R*. RStudio, Inc., Boston, MA.
URL http://www.rstudio.com/

[16] Ríos Insua, S., Mateos Caballero, A., Bielza Lozoya, M. C., & Jiménez Martín, A. (2004). *Investigación operativa: modelos determinísticos y estocásticos*. Editorial universitaria Ramón Areces.

[17] Taha, H. A. (2004). *Investigación de operaciones*. Pearson Educación.

[18] Taha, H. A. (2012). *Investigación de operaciones*. Pearson Educación.

[19] Vitoriano, B. (2009). Modelos operativos de gestión.
URL http://www.mat.ucm.es/~bvitoria/Archivos/Apuntes%20MOG_UCM.pdf

[20] Vitoriano, B. (2010). Programación matemática: métodos de optimización.
URL http://www.mat.ucm.es/~bvitoria/Archivos/MM_PMII.pdf

[21] Xie, Y. (2021). *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.33.
URL https://yihui.org/knitr/