Facultad de Matemáticas

Departamento de Estadística e Investigación Operativa

Grado en Matemáticas

INTRODUCCIÓN A LA PROGRAMACIÓN LINEAL MULTIOBJETIVO

Trabajo Fin de Grado

Autora:

Elvira Martínez Sánchez

Supervisado por:

Pedro Luis Luque Calvo

Septiembre 2022

Índice general

Prólogo

Me gustaría agradecer a mi familia su apoyo incondicional y su confianza en mí a lo largo de todos estos años. Sin su sostén no habría conseguido llegar hasta aquí.

Quisiera agradecer también a mi tutor, Pedro Luis Luque Calvo, por la dedicación que ha puesto en mi trabajo y el apoyo que ha supuesto para mí.

Resumen

Este trabajo pretende servir de introducción a la Programación Lineal Multiobjetivo, área de la Optimización Matemática que involucra múltiples criterios, frecuentemente conflictivos entre sí. Más concretamente, se ciñe a aquellos problemas en los que intervienen únicamente funciones de carácter lineal.

Después de hacer una breve introducción histórica, se presenta el modelo general, así como las principales definiciones y resultados. Se exponen también diferentes métodos con los que resolver problemas de Programación Multiobjetivo, siendo este el contenido de mayor peso en el trabajo. A lo largo del mismo, un ejemplo en común servirá de hilo conductor para ilustrar el funcionamiento de cada uno de los métodos expuestos.

Para finalizar, se muestra un método generalizado que resuelve problemas de Programación Lineal Multiobjetivo con ayuda de software disponible: AMPL y R.

Abstract

This study is thought to be an introduction to Multiobjective Linear Programming, an area of the Mathematical Optimisation that involves multiple criteria, usually conflicting among them. More specifically, it focuses on those problems which only involve linear functions.

After a brief historical background, the general model is introduced, as well as the main definitions and results. Different methods used to solve Multiobjective Programming problems are also presented, content which is considered to be the core one in this paper. Throughout the study, a common example is used as a guiding thread to illustrate the functioning of all of the displayed methods.

Finally, a generalised method that solves Multiobjective Linear Programming problems with the help of an available software is shown: AMPL and R.

Índice de figuras

Capítulo 1

Introducción. Programación Lineal Multiobjetivo

1.1. Contexto histórico

De acuerdo con [Luc, 2016], el concepto de **Programación Matemática** fue acuñado por primera vez en 1959 para referirse a la rama de las matemáticas que aborda el estudio de **problemas de optimización**. Este tipo de problemas ya habían sido estudiados por grandes matemáticos de la época en los siglos XVII, XVIII y XIX. Sin embargo, en sus orígenes, no existían métodos que los resolvieran de forma rápida y en grandes dimensiones.

Fue a raíz de los nuevos requerimientos militares, sociales, tecnológicos e industriales del siglo XX cuando surge la **Investigación Operativa**, disciplina que se encarga de la toma de decisiones apoyándose en la aplicación de métodos analíticos avanzados. Con ella, aparecen nuevos procedimientos matemáticos y técnicas computacionales que supusieron un cambio en el tratamiento de los problemas de Programación Matemática y brindaron la posibilidad de abordar problemas a grandes escalas en un tiempo mucho más limitado.

A día de hoy, la Programación Matemática está presente en cualquier área imaginable de la actividad humana, en mayor o menor medida. Interviene en procesos de asignación de recursos limitados, gestión de inventario, planificación de inversiones, logística militar, así como en la mejora de redes de transporte y cadenas de producción. En general, en todos aquellos procesos en los que se busque maximizar beneficios y minimizar costes o riesgos.

Tal y como se recoge en [Luc, 2016], la Programación Matemática estudia el problema de seleccionar el elemento óptimo de entre un conjunto de alternativas posibles (factibles), en base a cierto criterio (objetivo). Este problema puede representarse de la siguiente forma:

$$\begin{array}{c} \text{opt } f(x) \\ \text{s.a. } x \in X \end{array}$$

donde X es un conjunto no vacío, llamado **conjunto factible**, y $f : X \to \mathbb{R}$ es una función real, llamada **criterio** o **función objetivo**.

Siguiendo [Luc, 2016], en un problema de optimización como el anterior se busca "*maximizar*" o "*minimizar*" la función objetivo, lo que equivale a encontrar un ele-

mento $\bar{x} \in X$ de manera que $f(\bar{x}) \geq f(x)$ o $f(\bar{x}) \leq f(x)$, $\forall x \in X$, respectivamente. En caso de que exista, $\bar{x}$ se conoce como **solución óptima** del problema de programación.

En la realidad, existen multitud de aplicaciones que involucran, no solo un criterio, sino varios, que pretenden ser optimizados de manera simultánea. Además, estos suelen ser total o parcialmente conflictivos entre ellos, en el sentido de que la mejora de uno implica que otro empeore su valor.

De aquí surge el estudio de la Programación Multiobjetivo, también conocida como Decisión Multicriterio, que refleja de forma mucho más realista las distintas situaciones que se plantean en la vida real. A diferencia de la programación clásica con un único objetivo, en este caso no puede aplicarse el concepto de solución óptima pues, de forma general, no existirá ningún valor que consiga optimizar todas las funciones objetivo a la vez. En su lugar, la Programación Multiobjetivo busca el conjunto de soluciones, de entre todas las factibles, que sean más eficientes.

En última instancia, mediante los procesos de toma de decisiones, se elegirá la que se considere la mejor solución del problema Multiobjetivo, aquella que más se ajuste a las preferencias del decisor y de los interesados ([Ríos Insúa, 1996]).

Dichas soluciones eficientes se conocen históricamente como **Óptimos de Pareto**, en honor al economista italiano Vilfredo Pareto ($1848 - 1923$), a quien, de acuerdo con [Jiménez and Pintos, 2014], se le atribuyen las primeras menciones a problemas con múltiples objetivos que, tenidos en cuenta simultáneamente, resultan conflictivos.

El concepto Óptimo de Pareto, gestado en 1896, es de vital transcendencia y sirve de base para las teorías posteriores que sustentan la Programación Multiobjetivo, desarrolladas a partir de los años 50.

La primera reunión científica enfocada en la Decisión Multicriterio se celebró en La Haya (Países Bajos) en 1970, como recoge [Valdivia, 2016]. Este evento fue el punto de partida de una revolución en la ciencia multicriterio.

Cabe decir que, si bien la mayoría de situaciones reales se modelizan haciendo uso de funciones no lineales, este trabajo se ciñe a un caso particular de los problemas de Programación Multiobjetivo, aquellos cuyas funciones objetivo y restricciones son de carácter lineal.

1.2. Conceptos previos

En esta sección se hace una breve recapitulación de conceptos usados en Programación Matemática que serán necesarios para comprender y desarrollar el contenido de este trabajo.

Todas las definiciones que aparecen a continuación han sido extraídas de [Valencia].

Definición 1.2.1 Se define **recta** en $\mathbb{R}^n$ como el conjunto

$$r = \{x \in \mathbb{R}^n \colon x = \lambda x_1 + (1-\lambda)x_2;\ x_1, x_2 \in \mathbb{R}^n,\ \lambda \in \mathbb{R}\} = \{x \in \mathbb{R}^n \colon x = x_2 + \lambda(x_1 - x_2);\ x_1, x_2 \in \mathbb{R}^n,\ \lambda \in \mathbb{R}\}$$

Definición 1.2.2 Se define el **segmento lineal cerrado** (o simplemente segmento) entre dos puntos $x_1, x_2 \in \mathbb{R}^n$ como el conjunto

$$[x_1, x_2] = \{x \in \mathbb{R}^n \colon x = \lambda x_1 + (1-\lambda)x_2;\ x_1, x_2 \in \mathbb{R}^n,\ \lambda \in [0,1]\}$$

Definición 1.2.3 Un punto $x \in \mathbb{R}^n$ es una **combinación lineal convexa** de k puntos $\{x_1, \ldots, x_k\} \subset \mathbb{R}^n$ si puede expresarse de la siguiente forma

$$x = \textstyle\sum_{i=1}^{k} \lambda_i x_i,$$

con $\lambda_i \in [0,1]$, $\lambda_i \geq 0$, $\sum_{i=1}^{k} \lambda_i = 1$

Definición 1.2.4 Un **conjunto** $X \subset \mathbb{R}^n$ es **convexo** si

$$\forall x_1, x_2 \in X \text{ se verifica } \lambda x_1 + (1-\lambda)x_2 \in X,\ \forall \lambda \in [0,1],$$

es decir, si para todo par de puntos del conjunto, el segmento que los une está contenido en él.

Definición 1.2.5 Se define un **hiperplano** como el conjunto

$$H = \{x \in \mathbb{R}^n : c^T x = \alpha;\ \alpha \in \mathbb{R},\ c^T \in \mathbb{R}^n,\ c^T \neq 0\}$$

Se entiende como la generalización a $\mathbb{R}^n$ del concepto de recta en $\mathbb{R}^2$.

Definición 1.2.6 A partir del concepto de hiperplano se definen el **semiespacio cerrado inferior** como el conjunto

$$H = \{x \in \mathbb{R}^n : c^T x \leq \alpha;\ \alpha \in \mathbb{R},\ c^T \in \mathbb{R}^n,\ c^T \neq 0\}$$

y el **semiespacio cerrado superior** como

$$H = \{x \in \mathbb{R}^n : c^T x \geq \alpha;\ \alpha \in \mathbb{R},\ c^T \in \mathbb{R}^n,\ c^T \neq 0\}$$

Definición 1.2.7 Se denomina **politopo** a todo conjunto $X \subseteq \mathbb{R}^n$ que es intersección de un número finito de semiespacios cerrados.
Cuando el politopo X es acotado de denomina **poliedro**.

Se tiene que todo politopo es un conjunto convexo.

Definición 1.2.8 Sea $X \subseteq \mathbb{R}^n$ un conjunto convexo. Se dice que $x \in X$ es un **vértice** o **punto extremo** del poliedro X si se verifica que

$$\nexists x_1, x_2 \in X\ ,\ x_1 \neq x_2 \text{ tales que } x = \lambda x_1 + (1-\lambda)x_2,$$

es decir, cuando no es posible expresar x como combinación lineal convexa de otros dos puntos distintos de X.

Todo poliedro acotado (esto es, todo politopo) tiene un punto extremo.

Definición 1.2.9 Sea $X \subseteq \mathbb{R}^n$ un conjunto convexo no vacío y sea $f : X \longrightarrow \mathbb{R}$. Se dice que f es una **función convexa** en X si verifica

$$f([\lambda x_1 + (1-\lambda)x_2]) \leq \lambda f(x_1) + (1-\lambda)f(x_2) \ , \forall \lambda \in [0,1] \text{ y } \forall x_1, x_2 \in X$$

Se dice que f es una **función cóncava** en X si verifica

$$f([\lambda x_1 + (1-\lambda)x_2]) \geq \lambda f(x_1) + (1-\lambda)f(x_2) \ , \forall \lambda \in [0,1] \text{ y } \forall x_1, x_2 \in X$$

En el capítulo a continuación se desarrolla la teoría de Programación Lineal Multiobjetivo propiamente dicha.

1.3. Definiciones básicas. Planteamiento del problema

Se considera el problema de Programación Lineal Multiobjetivo, denotado (MOLP) por sus siglas en inglés (*Multiobjetive Linear Programming Problem*), y dado de la siguiente forma:

$$\begin{array}{r} \max C^T x \\ \text{s.a. } x \in X \end{array} \tag{1.1}$$

donde X es un poliedro convexo no vacío en $\mathbb{R}^n$ que define el conjunto factible del **espacio de decisiones**, $C \in \mathbb{R}^{n \times k}$ es la **matriz de objetivos** y $x \in \mathbb{R}^n$ es el vector de **variables de decisión**.

Llamamos $Y = C^T X$ al conjunto factible del **espacio de objetivos**.

El vector $C^T x$ representa el conjunto de las **funciones objetivo** y puede ser reescrito equivalentemente como $((c^1)^T x, \ldots, (c^k)^T x)^T$, donde c^i, con $i = 1, \ldots k$ (número de objetivos), son las columnas de C, o bien $f(x) = (f_1(x), \ldots, f_k(x))$ (Véase [Ehrgott et al., 1953]).

El problema (MOLP) puede venir dado en varias formas, de acuerdo con [Luc, 2016], según las ecuaciones que definan el conjunto de restricciones X:

FORMA ESTÁNDAR	FORMA CANÓNICA
$\max C^T x$	$\max C^T x$
s.a. $Ax = b$	s.a $Ax \leqq b$
$x \geqq 0$	$x \geqq 0$

donde $A \in \mathbb{R}^{m \times n}$ es la **matriz de restricciones**, cuyas entradas a_{ij}, $i = 1, \ldots, m$ (número de restricciones) , $j = 1, \ldots, n$ (número de variables), se conocen como **coeficientes tecnológicos**, y $b \in \mathbb{R}^m$ es el **vector lado derecho** o vector de recursos.

Siguiendo la línea de [Ríos Insúa, 1996], se considerará siempre un problema de "maximización". En el caso en que el problema de optimización busque minimizar alguna de las funciones objetivo, esta se multiplicará por -1 para situarse en el caso anterior.

De igual modo, se pueden modificar las restricciones convenientemente para tener el problema en una forma u otra. Por ejemplo, una restricción de igualdad puede desdoblarse en dos desigualdades, obteniendo el mismo conjunto factible. También resulta útil multiplicar por -1 una inecuación para obtener la desigualdad contraria.

A continuación se definen algunos de los elementos fundamentales de la Programación Lineal Multiobjetivo. Se han consultado, principalmente, las fuentes [Ríos Insúa, 1996], [Luc, 2016], [Ehrgott et al., 1953] y [Geoffrion, 1968].

Definición 1.3.1 Se dice que un **punto** $x \in \mathbb{R}^n$ es **factible** si verifica todas las restricciones del (MOLP).
Se dice que el (MOLP) es **infactible** si no existe ningún vector x que verifique todas las restricciones.

Definición 1.3.2 Una solución factible $x^* \in X$ es una **solución (débilmente) eficiente** si $\nexists x \in X$ tal que $C^T x \geq C^T x^*$ y $C^T x \neq C^T x^*$ ($C^T x > C^T x^*$), es decir, si no existe otra solución factible que mejore el valor de una función objetivo sin causar una disminución en alguna de las funciones objetivo restantes.
También se usa el término **Óptimo (débil) de Pareto** para referirse a una solución (débilmente) eficiente.
Se denotará por X_{ef} al conjunto de todas las soluciones eficientes o conjunto eficiente en el espacio de decisiones.

Definición 1.3.3 Una solución factible $x^* \in X$ es una **solución propiamente eficiente** si es eficiente y además $\exists M > 0$ constante tal que, para cada i, se tiene

$$\frac{f_i(x)-f_i(x^*)}{f_j(x^*)-f_j(x)} \leq M$$

para algún j de manera que $f_j(x) < f_j(x^*), \forall x \in X$, y $f_i(x) > f_i(x^*)$.
Una solución eficiente x^* que no verifica lo anterior se dice que es **impropiamente eficiente**.

Definición 1.3.4 Si x^* es una solución (débilmente) eficiente, $C^T x^*$ se llama **valor (débilmente) eficiente** asociado a x^*.
Indistintamente se usa el término **punto no (débilmente) dominado** para referirse a $C^T x^*$.
Se denotará por $Y_{ef} = C^T X_{ef}$ al conjunto de todos los valores eficientes o conjunto eficiente en el espacio de objetivos.

Definición 1.3.5 Se dirá que un **punto** $x \in X$ está **dominado** si $\exists \bar{x} \in X$ tal que $f(\bar{x}) \geq f(x)$. Es decir, si para su transformado $f(x)$ existen puntos en $Y = f(X)$ que mejoran alguna de sus coordenadas sin empeorar la otra. En ese caso se dirá que $\bar{x}$ domina a x o que x está dominado por $\bar{x}$.

Es evidente que si un punto está dominado entonces no podrá ser solución eficiente.

La Programación Multiobjetivo se centra principalmente en identificar los elementos que conforman el conjunto eficiente. Más adelante, en el Capítulo 2, se presentan algunos métodos para generar X_{ef}. Usualmente, X_{ef} está formado por muchas soluciones, de entre las que deberá escogerse la solución final.

Definición 1.3.6 Se conoce como **solución de mejor compromiso** a aquella escogida por el tomador de decisiones, de entre todas las eficientes, como la más preferida.

Esta fase final en la que se delibera acerca de qué solución se considera óptima, se conoce como Proceso de Toma de Decisiones Multicriterio.

Es tal la complejidad y tan amplio el abanico de sectores en el que puede aplicarse la Programacion Multiobjetivo que existen organizaciones dedicadas a la investigación y al constante desarrollo de métodos que faciliten al decisor la elección de la solución de mejor compromiso.

Ejemplos de ello son el *Grupo Español de Decisión Multicriterio (GEDM)* y la *International Society on MCDM (Multiple Criteria Decision Making)*. Pueden consultarse sus Páginas Web Oficiales en [GED] y [MCD], respectivamente.

1.4. Optimalidad

Un resultado fundamental de la Programación Lineal Multiobjetivo sostiene que las soluciones propiamente eficientes del (MOLP) pueden ser caracterizadas como soluciones óptimas de los problemas de programación lineal con un solo objetivo, por medio de una combinación convexa de los objetivos $f_i(x)$, $i = 1, \ldots, k$.

Se define el conjunto

$$\Lambda = \{\lambda \in \mathbb{R}^k : \lambda > 0, \sum_{i=1}^{k} \lambda_i = 1\} \ .$$

Teorema 1.4.1 Sea $X \in \mathbb{R}^n$ un conjunto convexo. Sean $f_i, i = 1, \ldots, k$, cóncavas en X. Entonces un punto factible $x^* \in X$ es una solución propiamente eficiente del (MOLP) si y sólo si $\exists \lambda \in \Lambda$ tal que x^* es solución óptima del problema de un solo objetivo, denotado (P_λ),

$$\begin{gathered} \max \ f_\lambda(x) := \sum_{i=1}^{k} \lambda_i f_i(x) \\ \text{s.a. } x \in X \end{gathered}$$

Demostración. $\Rightarrow\lrcorner$ Se supone que x^* es solución propiamente eficiente del (MOLP). Entonces, por definición, existe $M > 0$ tal que $\forall i = 1, \ldots, k$, el sistema

$$\begin{gathered} f_i(x) > f_i(x^*) \\ f_i(x) + M f_j(x) > f_i(x^*) + M f_j(x^*), \ \forall j \neq i \end{gathered}$$

no tiene solución en X.

Por ser f_i cóncava, para el i-ésimo sistema $\exists \lambda_j^i, j = 1, \ldots, k$, con $\sum_{j=1}^{k} \lambda_j^i = 1$ tal que

$$\lambda_i^i f_i(x) + \textstyle\sum_{j \neq i} \lambda_j^i (f_i(x) + M f_j(x)) \leq \lambda_i^i f_i(x^*) + \sum_{j \neq i} \lambda_j^i (f_i(x^*) + M f_j(x^*))$$

O equivalentemente

$$f_i(x) + M \textstyle\sum_{j \neq i} \lambda_j^i f_j(x) \leq f_i(x^*) + M \sum_{j \neq i} \lambda_j^i f_j(x^*)$$

$\forall x \in X$. Sumando en i y reordenando se tiene

$$\sum_{j=1}^{k}(1+M\sum_{i\neq j}\lambda_j^i)f_j(x) \leq \sum_{j=1}^{k}(1+M\sum_{i\neq j}\lambda_j^i)f_j(x^*)$$

$\forall x \in X$.

$\Leftarrow\lrcorner$ Se supone que $\exists\lambda = (\lambda_1, \ldots, \lambda_k) \in \Lambda$ tal que x^* es solución óptima del problema de un sólo objetivo (P_λ).

Es trivial que x^* es eficiente para (MOLP). Se demuestra que es propiamente eficiente, con

$$M = (k-1)\cdot \text{máx}_{i,j}\{\tfrac{\lambda_j}{\lambda_i}\},\ k \geq 2$$

Por reducción al absurdo, se supone que x^* no es propiamente eficiente para (MOLP). Entonces, para algún objetivo f_i y para algún $x \in X$ se tiene

$$f_i(x) - f_i(x^*) > M\cdot(f_j(x^*) - f_j(x)),\ \forall j \text{ tal que } f_j(x) < f_j(x^*)$$

Como consecuencia directa se tiene

$$f_i(x) - f_i(x^*) > \tfrac{k-1}{\lambda_i}\lambda_j\cdot(f_j(x^*) - f_j(x)),\ \forall j \neq i$$

Multiplicando por $\frac{\lambda_i}{k-1}$ y sumando en j se tiene

$$\lambda_i(f_i(x) - f_i(x^*)) > \sum_{j\neq i}\lambda_j(f_j(x^*) - f_j(x))$$

Esto es absurdo, pues se ha supuesto que x^* es óptimo para (P_λ). ■

Este teorema, que puede ser consultado en [Geoffrion, 1968], tiene gran utilidad desde el punto de vista computacional, pues reduce a un problema de programación paramétrico encontrar soluciones propiamente eficientes.

Teorema 1.4.2 El conjunto X_{ef} de soluciones eficientes del problema (1.1) tiene las siguientes propiedades:
(i) Si un punto interior de una cara del poliedro X es eficiente, entonces todos los puntos de esa cara lo son.
(ii) Si X tiene un vértice y (1.1) tiene una solución eficiente, entonces tiene una solución eficiente en un vértice de X.
(iii) X_{ef} consiste en caras del poliedro X y es cerrado y conexo por arco.

Este teorema es de suma importancia pues implica que a la hora de buscar el conjunto eficiente basta con examinar los puntos de la frontera del poliedro X. Esto evita tener que considerar todo el conjunto de soluciones factibles, lo que supone un ahorro enorme de tiempo y esfuerzo.

Teorema 1.4.3 Sea $x \in X$ un punto factible. Si x es solución óptima única para el problema uniobjetivo asociado a alguna de las funciones criterio del problema (1.1), entonces x es eficiente para (1.1).

1.5. Aplicaciones reales que usan Programación Lineal Multiobjetivo

De acuerdo con [Ríos Insúa, 1996], los problemas de Programación Multiobjetivo surgen en los procesos de toma de decisiones en multitud de áreas de la actividad humana tales como la economía, la ingeniería, el transporte, la medicina, etc.

En particular, la Programación Lineal Multiobjetivo tiene importantísimas aplicaciones en la vida real. En este apartado se recogen un par de ejemplos de ello.

En primer lugar, se presenta una situación real que puede ser más o menos compleja y que se desea solventar. En el planteamiento del problema intervienen ciertos objetivos a los que se pretende aspirar con la aplicación de tal supuesta solución. Además, se imponen también una serie de requerimientos que deben cumplirse para que la solución sea satisfactoria. Todo este escenario que *a priori* puede parecer difícil de manejar, se consigue modelar en forma de variables, funciones objetivo y restricciones.

De esta manera, haciendo uso de las distintas estrategias que ya existen para resolver estos problemas (algunas de ellas serán expuestas en el próximo capítulo) se obtendrán aquellas soluciones que encajen lo mejor posible con lo que se pretende conseguir en la situación real en concreto.

En última instancia, serán los expertos de la materia en cuestión o los posibles interesados los que actúen como tomadores de decisiones.

1.5.1. Diseño de tratamientos de radioterapia

De una manera simplificada se presenta un ejemplo de modelización para el diseño de un tratamiento de radioterapia para tratar casos de cáncer. Este ejemplo se menciona en [Ehrgott, 2005] y hace referencia a la investigación desarrollada en [Sonderman and Abrahamson, 1985].

La terapia de radiación se usa en medicina en el tratamiento de enfermedades tumorales. Mediante la aplicación de rayos de radiación de una forma determinada, se consigue dañar el ADN de las células cancerígenas y por consiguiente, frenar su avance en el organismo del paciente.

Existe tecnología capaz de modular de manera independiente la dirección e intensidad de cada rayo del conjunto de rayos de radiación totales.

Para concretar en qué zonas se va a aplicar el tratamiento, se discretiza el cuerpo del paciente en m puntos de dosis. Cada punto de la discretización se supone clasificado en tres grupos según se localice en el tumor T, en tejido sano S o en alguna de las l estructuras críticas $\{E_1, \ldots, E_l\}$.

La dosis tumoral prescrita por el especialista viene dada por d_T.

El objetivo utópico sería diseñar un tratamiento que aplique exactamente la dosis necesaria d_T uniformemente sobre el tumor a la vez que niguna dosis recae sobre regiones no afectadas. Pero en la práctica, esto no es factible, ya que las células tumorales se intercalan a nivel microscópico con células sanas.

Es por ello que los especialistas dan por hecho que se va a aplicar o bien una dosis por defecto en la región tumoral (denotada z_T), lo que permitirá sobrevivir a las células ma-

lignas, o bien una dosis por exceso sobre tejido sano (denotada z_S) o sobre las estructuras críticas (denotada z_{E_k}, para cada $k = 1, \ldots, l$) , lo que puede agravar aún más la salud del paciente.

Lo que se pretende conseguir entonces es minimizar todo lo posible estos errores a la hora de aplicar el tratamiento.

El vector de variables de decisión será $x \in \mathbb{R}^n$, que describe la intensidad determinada para cada rayo de radiación, donde n es el número total de rayos.

La matriz $A \in \mathbb{R}^{m \times n}$, donde la entrada a_{ij} indentifica al rayo j en el punto de dosis i, según si estos puntos se situan sobre el tumor, sobre tejido sano o sobre estructuras críticas.

Por tanto, Ax marcará el tratamiento aplicado en los puntos de dosis.

Es necesario, además, fijar unas cotas superiores para evitar una dosis demasiado elevada que podría ocasionar efectos secundarios graves en el paciente. Se denotan estas por $u_T, u_S, u_{E_k}, k = 1, \ldots, l$, según a qué regiones hagan referencia. Se asume que deben aplicarse a cada punto de la discretización.

Con todo ello se puede modelizar el problema de la siguiente manera

$$
\begin{array}{lrcll}
\min & (z_T, z_{E_1}, \ldots, z_{E_l}, z_S) & & & \\
\text{s.a} & A_T x + z_T e & \geq & d_T & \\
& A_T x & \leq & u_T & \\
& A_{E_k} x - z_{E_k} e & \leq & u_{E_k}, & k = 1, \ldots, l \\
& A_S x - z_S e & \leq & u_S & \\
& z_{E_k} & \geq & -u_{E_k}, & k = 1, \ldots, l \\
& z_S & \geq & 0 & \\
& x & \geq & 0 &
\end{array}
$$

En este modelo, el objetivo es encontrar las soluciones eficientes $(x, z) \in \mathbb{R}^{n+l+2}$ tales que minimizan simultáneamente la infradosificación en los tumores y la sobredosificación en las estructuras críticas y los tejidos sanos.

1.5.2. Compra de aviones de una compañía aérea

A continuación se muestra otro ejemplo de aplicación práctica de la Programación Lineal Multiobjetivo. Este puede ser consultado en [Morales, 2014].

Se presenta el caso de una aerolínea que desea ampliar su flota para cubrir nuevas rutas de vuelo. Para ello pretende determinar el número óptimo de aeronaves que debe comprar.

La compañía aérea baraja dos modelos diferentes de aviones, notados A y B, con diferentes características de capacidad, motor, alcance, autonomía y número de vuelos diarios máximos posibles, además de diferentes costes de adquisición, mantenimiento y consumo.

Se consideran las variables a y b como el número de aviones que se compran del modelo A y B, respectivamente.

Como objetivos, la empresa se plantea, en primer lugar, maximizar el beneficio diario, medido en euros, resultante de los ingresos por actividad menos los gastos operativos.

Por un lado, los ingresos proceden de los billetes diarios vendidos y dependen de la clase de esos pasajes; según sea turista (t) o business (s). Si se consideran los siguientes parámetros

P_t, P_s: precio unitario de cada billete según la clase,
t_A, s_A: número de pasajeros transportados por el modelo A según la clase,
t_B, s_B: número de pasajeros transportados por el modelo B según la clase,

se pueden modelar los ingresos como

$$\text{Ingresos} = a[t_A P_t + s_A P_s] + b[t_B P_t + s_B P_s].$$

Por otro lado, los gastos corresponden a los costes diarios de poner en aire a los aviones. Considerando

C_t^A, C_s^A: coste por pasajero en el modelo A según la clase,
C_t^B, C_s^B: coste por pasajero en el modelo B según la clase,

se tiene que

$$\text{Gastos} = a[C_t^A t_A + C_s^A s_A] + b[C_t^B t_B + C_s^B s_B].$$

Por tanto, este criterio puede modelarse de la siguiente forma:

$$\text{Beneficio} = \text{Ingresos} - \text{Gastos} =$$
$$= a[t_A P_t + s_A P_s] + b[t_B P_t + s_B P_s] - a[C_t^A t_A + C_s^A s_A] - b[C_t^B t_B + C_s^B s_B]$$

En segundo lugar, se quiere minimizar el consumo de combustible. Teniendo en cuenta que

c_A,c_B: consumo por pasajero en el modelo A y B, respectivamente (expresado en litros por cada 100 kilómetros),

se consigue modelar el consumo como

$$\text{Consumo} = a[(t_A + s_A)c_A] + b[(t_B + s_B)c_B].$$

Además, como en toda situación real, se dan limitaciones en forma de restricciones.

Existe un presupuesto disponible limitado en la empresa, procedente del beneficio y las amortizaciones acumuladas. Si se considera

D: capital máximo disponible,
G_A, G_B: precio de adquisición del modelo A y B respectivamente,

esta restricción puede expresarse como

$$G_A a + G_B b \leq D.$$

Es más, se puede aspirar a subvenciones si se cumplen ciertos requisitos. En este caso, se han de comprar un número mínimo, m_A, de aviones del modelo A para poder aspirar a ellas:

$$a \geq m_A.$$

Tras un estudio de la demanda, se concluye que debe realizarse un número mínimo de viajes diario para poder cubrirla. Notando

V_A, V_B: número de viajes al día que puede realizar el modelo A y B, respectivamente,
U: número mínimo de viajes necesarios para satisfacer la demanda,

se obtiene la siguiente restricción

$$V_A a + V_B b \geq U.$$

Por último cabe notar que, como es obvio, no puede comprarse un número negativo de aviones y, por tanto, se tiene la restricción de no negatividad

$$a, b \geq 0.$$

Así pues, se ha conseguido establecer un modelo matemático de la situación real planteada, y queda como sigue:

$$\begin{gathered}
\max\ a[t_A P_t + s_A P_s] + b[t_B P_t + s_B P_s] - a[C_t^A t_A + C_s^A s_A] - b[C_t^B t_B + C_s^B s_B] \\
\min\ a[(t_A + s_A)c_A] + b[(t_B + s_B)c_B] \\
\begin{aligned}
\text{s.a.} \quad G_A a + G_B b &\leq D \\
a &\geq m_A \\
V_A a + V_B b &\geq U \\
a, b &\geq 0
\end{aligned}
\end{gathered}$$

Capítulo 2

Métodos de resolución

2.1. Introducción

En este capítulo se exponen algunos de los métodos más importantes utilizados tanto para generar el conjunto eficiente como para determinar la solución de mejor compromiso. El contenido completo de este capítulo ha sido desarrollado gracias a [Ríos Insúa, 1996] y [Ríos Insúa, 1997].

En primer lugar se describe el Método Simplex Multiobjetivo (Lee, 1972), que es una extensión del Método Simplex convencional para problemas lineales con un sólo objetivo.

Seguidamente, se describen los métodos de las ponderaciones (Zadech, 1963) y de las ε-restricciones (Marglin, 1967), que tienen un amplio espectro de aplicaciones ya que también permiten ser usados en problemas de carácter no lineal. Interpretando estos métodos de forma adecuada se facilita la elección de la solución de mejor compromiso.

Por último, se menciona la Programación por Metas (Charnes y Cooper, 1961), que está basada en el Método Simplex y tiene en cuenta el criterio del tomador de decisiones, lo que hace inmediato obtener la solución eficiente que más se adecúa a las preferencias del decisor.

La diferencia entre el Método Simplex Multiobjetivo y el resto de métodos expuestos está en que el primero trabaja directamente con todas las funciones objetivo, mientras que los demás transforman el problema en uno unidimensional que puede ser resuelto convencionalmente.

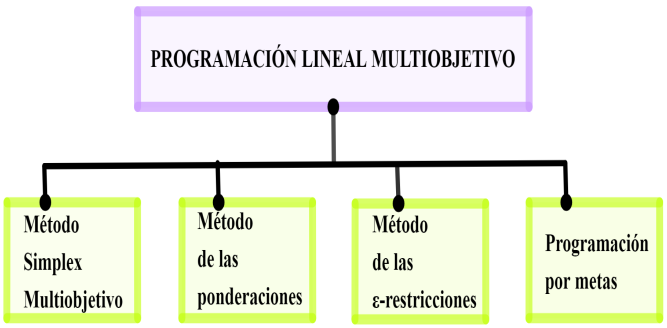


Figura 2.1: Métodos de resolución

Existen casos particulares en los que el conjunto eficiente se puede obtener de manera sencilla haciendo uso de la representación gráfica sin necesidad de aplicar uno de los métodos anteriores. A continuación se presenta un ejemplo de ello para un problema biobjetivo.

■ **Ejemplo 2.1** Un cliente quiere proveer de luz cierto espacio y para ello dispone de dos tipos de producto de iluminación A y B, con características ecológicas y potenciales distintas. A su vez, desea considerar simultáneamente los objetivos

(1): maximizar el ahorro energético
(2): maximizar la luminosidad,

que son, al menos, parcialmente contradictorios.

Se toman las variables de decisión

x_1: decenas de unidades de producto A y
x_2: decenas de unidades de producto B,

Se supone que el problema se modeliza con las funciones objetivo

$$\max f_1(x) = -x_1 + x_2 \text{ (ahorro energético)}$$
$$\max f_2(x) = x_1 + 2x_2 \text{ (luminosidad)},$$

las restricciones

$$-x_1 + 2x_2 \leq 8 \text{ (control sujeto a normativa)}$$
$$x_1 + x_2 \leq 8 \text{ (capacidad máxima total)}$$
$$x_1 \leq 6 \text{ (capacidad máxima de producto A)}$$

y las condiciones de no negatividad

$$x_1, x_2 \geq 0.$$

Se tiene, por tanto, el siguiente problema Multiobjetivo:

$$\begin{array}{rrrcl} \max f(x) = (f_1(x), f_2(x)) = (-x_1 + x_2, x_1 + 2x_2) & & & & \\ \text{s.a.} & -x_1 & +2x_2 & \leq & 8 \\ & x_1 & +x_2 & \leq & 8 \\ & x_1 & & \leq & 6 \\ & x_1, & x_2 & \geq & 0 \end{array}$$

Equivalentemente, en forma matricial

$$\max f(x) = ((c^1)^T x, (c^2)^T x)$$
$$\text{s.a. } Ax \leq b$$
$$x \geq 0$$

En la Figura 2.2 se representa la región factible X del espacio de decisiones, que queda delimitada por las restricciones. Los puntos de X constituyen el conjunto de soluciones factibles.

La región factible Y del espacio de objetivos se calcula mediante $f(X) = (f_1(X), f_2(X))$ y queda representada en la Figura 2.3.

Se observa que a cada punto extremo de X le corresponde uno y sólo un punto extremo de Y.

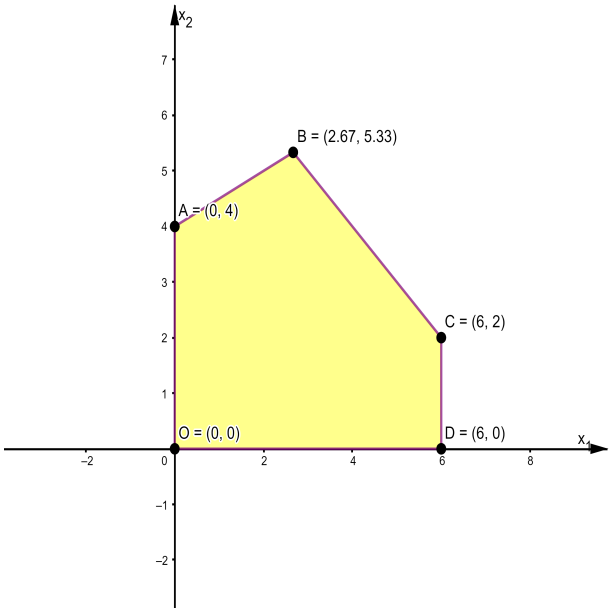


Figura 2.2: Región factible del espacio de decisiones

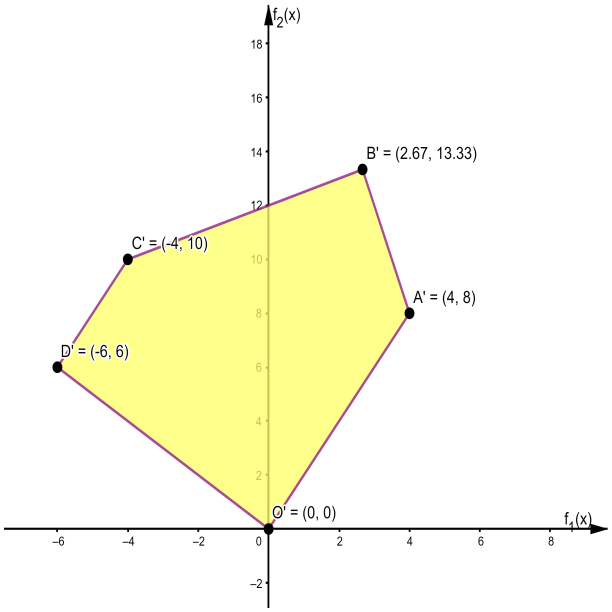


Figura 2.3: Región factible del espacio de objetivos

Para identificar los puntos eficientes analizamos la información recogida en la siguiente tabla:

Punto extremo de X	(x_1, x_2)	$f(x_1, x_2)$	Punto extremo de Y
O	(0,0)	(0,0)	O’
A	(0,4)	(4,8)	A’
B	$(\frac{8}{3}, \frac{16}{3})$	$(\frac{8}{3}, \frac{40}{3})$	B’
C	(6,2)	(-4,10)	C’
D	(6,0)	(-6,6)	D’

Conviene recordar, como se vio en el Teorema 1.4.2, que para calcular el conjunto eficiente basta con examinar los puntos que están sobre la frontera de X.

Se comienza considerando cada objetivo por separado y calculando su solución óptima. Para ello, se representan las curvas de nivel como aparecen en la Figura 2.4. Al ser dos funciones objetivo a maximizar, ambas tendrán sentido de movimiento ascendente.

En el caso de la función objetivo $f_1(x) = -x_1 + x_2$, la curva de nivel que cumple con la solución óptima es la más alejada del origen (aparece en rojo con trazo continuo en la Figura 2.4) y el punto donde se consigue el valor objetivo máximo es el punto A, con $f_1(A) = 4$.

Análogamente ocurre con el objetivo $f_2(x) = x_1+2x_2$, cuyas curvas de nivel (representadas en azul) alcanzan el valor objetivo máximo en el punto B y éste es $f_2(B) = \frac{40}{3} \approx 13.33$.

Así, los puntos A y B son óptimos para uno de los objetivos y, de acuerdo con el Teorema 1.4.3, ambos son soluciones eficientes.

Como los puntos A y B son adyacentes, se tiene que todos los puntos del segmento [A,B] son también eficientes, es decir, no existen otros puntos en el conjunto factible que mejoren el valor de una de las funciones objetivo sin causar una disminución en la otra.

Ahora bien, se comprueba que no ocurre lo mismo con el resto de segmentos. Si consideramos, por ejemplo, el segmento $[O', A']$ en la Figura 2.3, vemos que el punto A domina a todos los puntos restantes, esto es, mejora ambos objetivos simultáneamente, luego ninguno de los puntos restantes del segmento puede ser solución eficiente. De manera análoga puede verse en los demás segmentos.

Así pues, se concluye que el conjunto eficiente X_{ef} está formado todos los puntos del segmento $[A, B]$.

Como observación, se puede ver en la Figura 2.4 que el punto que maximiza ambos objetivos (intersección entre las líneas continuas roja y azul) no está dentro de la región factible. Esto concuerda con la idea de que, en general, no existen soluciones óptimas para los problemas de Programación Multiobjetivo.

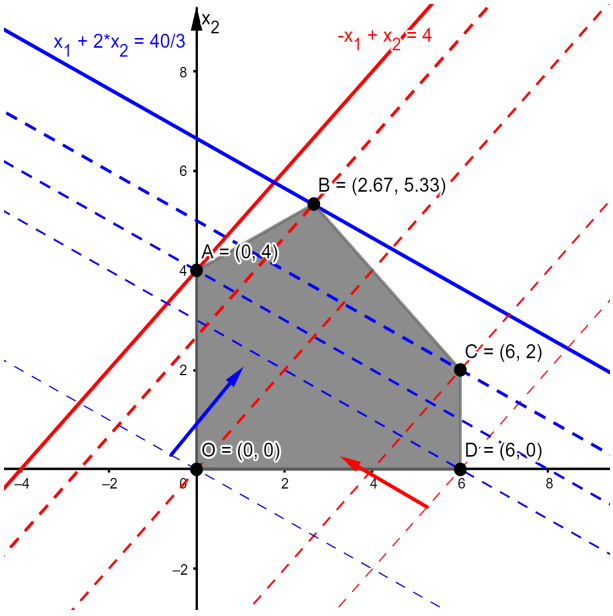


Figura 2.4: Curvas de nivel

■

Cabe decir que esta estrategia empleada es útil exclusivamente para el caso bidimensional.

2.2. Método Simplex Multiobjetivo

Se considera el siguiente problema, que viene dado en su forma estándar tras haberle añadido s variables de holgura:

$$\begin{gathered}\max\ f(x) = (f_1(x), \ldots, f_k(x)) \\ \text{s.a. } Ax = b \\ x \geq 0\end{gathered}$$

donde ahora $A \in \mathbb{R}^{m\times(n+s)}$, con entradas a_{ij}, $i = 1 \ldots m$, $j = 1 \ldots (n+s)$, y $b \in \mathbb{R}^m$.

Las funciones objetivo pueden reescribirse como $f(x) = ((c^1)^T x, \ldots, (c^k)^T x)$, donde ahora el vector de variables de decisión es $x = (x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+s})$ y las componentes de los coeficientes $c^i, i = 1, \ldots, k$ correspondientes a las variables de holgura, son nulas.

Equivalentemente, en formato tabla:

$c^k \rightarrow$	$c_1^k \ldots c_n^k$	$0 \ldots 0$	
	.	.	
	.	.	
$c^1 \rightarrow$	$c_1^1 \ldots c_n^1$	$0 \ldots 0$	
	$x_1 \ldots x_n$	$x_{n+1} \ldots x_{n+s}$	b
x_{B_1}	$a_{11} \ldots a_{1n}$	$10 \ldots 0$	b_1
.	.	$01 \ldots 0$	.
.	.	:	.
x_{B_s}	$a_{m1} \ldots a_{mn}$	$00 \ldots 1$	b_m
$z^1 \rightarrow$	$z_1^1 \ldots z_n^1$	$z_{n+1}^1 \ldots z_{n+s}^1$	y_1
	.	.	.
	.	.	.
$z^k \rightarrow$	$z_1^k \ldots z_n^k$	$z_{n+1}^k \ldots z_{n+s}^k$	y_k

donde z_j, $i = 1, \ldots, (n+s)$, son los vectores de costes reducidos de cada variable, cuyas componenentes se calculan como sigue:

$$z_j^i = y_j^i - c_j^i,$$

que corresponde al coste reducido de la variable x_j para la función objetivo $f_i(x)$. Las entradas y_i, $i = 1, \ldots, k$, son los valores objetivo.

Cuando una variable no básica con valor θ entra en la base, los valores objetivos varían de la siguiente forma:

$$\hat{y_i} = y_i - \theta z_j^i,\ i = 1, \ldots, k$$

El siguiente teorema asegura bajo ciertas hipótesis, la existencia de solución eficiente.

Teorema 2.2.1 Sea X región factible no vacía del (MOLP). Si todas las funciones objetivo están acotadas en ella, entonces existe, al menos, un punto extremo eficiente.

2.2.1. Descripción del método

La estrategia general para encontrar soluciones eficientes consiste en escoger una base factible por la que empezar y comprobar la eficiencia de su correspondiente solución básica factible, quedando almacenada en caso de serlo. Pasar a un punto extremo adyacente y repetir el proceso, de manera que se parará cuando todos los vértices del conjunto adyacente hayan sido estudiados.

Una forma de comprobar la eficiencia de una solución básica factible viene dada en el teorema siguiente.

Teorema 2.2.2 Sea B una base factible y sea $x_B = B^{-1}b$ su correspondiente solución básica facible. Si en la tabla del Simplex aparece una columna a_j no básica de manera que

$$z_j^i \leq 0, \forall i = 1, \ldots, k,$$

y al menos para algún valor de $i \in \{1, \ldots, k\}$ se verifica la desigualdad estricta $z_j^i < 0$, entonces x_B no es eficiente.

Este resultado implica que, en el caso de que moviéndose a un punto extremo adyacente introduciendo una nueva variable no básica a la base se mejore al menos uno de los valores objetivo, manteniendo constantes los demás, entonces la solución actual no es eficiente.

Bajo las hipótesis del siguiente teorema, introduciendo una nueva variable no básica se llega a una solución dominada por la actual.

Teorema 2.2.3 Sea B una base factible y sea $x_B = B^{-1}b$ su correspondiente solución básica facible. Si en la tabla del Simplex aparece una columna a_j no básica de manera que

$$z_j^i \geq 0, \forall i = 1, \ldots, k,$$

y al menos para algún valor de $i \in \{1, \ldots, k\}$ se verifica la desigualdad estricta $z_j^i > 0$, entonces introduciendo la variable x_j en la base se llega a una solución dominada por x_B.

Por tanto, si moviéndose a un punto extremo adyacente se empeora el valor de al menos un objetivo manteniendo constantes los demás, la solución actual domina a la nueva solución obtenida.

Teorema 2.2.4 Sea B una base factible y sea $x_B = B^{-1}b$ su correspondiente solución básica facible. Sea θ_j el valor de x_j si la columna a_j entra en la base. Si en la tabla del Simplex aparecen dos columnas no básicas distintas a_p y a_q de manera que

$$\theta_p z_p^i \leq \theta_q z_q^i, \forall i = 1, \ldots, k,$$

y al menos para algún valor de $i \in \{1, \ldots, k\}$ se verifica la desigualdad estricta $\theta_p z_p^i < \theta_q z_q^i$, entonces la solución básica factible que se obtiene al introducir la columna a_p en la base domina a la solución básica factible que se obtiene al introducir a_q.

2.2.2. Comprobación de la eficiencia de una solución

Se da un caso en el que la eficiencia de la solución básica factible actual puede comprobarse directamente observando la tabla. Este es el caso en que los costes reducidos de todas las variables no básicas para alguno de los objetivos son estrictamente positivos. Tal situación se interpreta como que la solución actual es máximo único para ese objetivo en concreto y, de acuerdo con el Teorema 1.4.3, es por tanto solución eficiente.

Si por el contrario algunos de los costes reducidos de las variables no básicas para alguno de los objetivos son positivos y otros nulos, entonces se interpreta como que la solución actual es máximo no único para tal objetivo, lo cual hace indicar que la solución podría no ser eficiente.

De forma general, se puede comprobar la eficiencia de una solución básica factible $\bar{x}$ resolviendo el siguiente problema de optimización uniobjetivo, denotado $(P)_E$, en el que se consideran las variables de holgura h_i:

$$\begin{array}{lrll} \max & E = \sum_{i=1}^{k} h_i & & \\ \text{s.a.} & x \in X & & \\ & f_i(x) - h_i = & f_i(\bar{x}), & i = 1, \ldots, k \\ & h_i \geq & 0, & i = 1, \ldots, k \end{array} \tag{2.1}$$

Si $E = 0$, entonces la solución básica factible $\bar{x}$ es eficiente para el problema multiobjetivo.

Si $E > 0$, entonces no es posible asegurar la eficiencia de $\bar{x}$.

2.2.3. Algoritmo del Simplex Multiobjetivo

A continuación se detallan los pasos a seguir en la aplicación del algortimo del Simplex Multiobjetivo.

PASO 1.
Construir la tabla del Simplex.

PASO 2.
Hacer $h = 1$ (índice de la base factible actual) y $g = 0$ (contador de soluciones eficientes).

PASO 3.
Determinar una base factible B^h con la que comenzar el algoritmo.

Esto puede hacerse mediante alguna estrategia del Simplex convencional, como añadiendo variables de holgura y considerando sus columnas correspondientes como base.

Se recomienda escoger como solución básica factible inicial una que sea óptima para alguno de los objetivos, pues facilitará los cálculos posteriores.

En el caso de que no exista, el problema será infactible y se para el algoritmo.

PASO 4.
Determinar la solución básica factible x^h asociada a la base B^h.

PASO 5.
(Se busca una fila de costes reducidos que tenga todas sus entradas no negativas)
Si existe un valor $i \in \{1, \ldots, k\}$ tal que $z_j^i \geq 0, \forall a_j$, entonces x^h maximiza el objetivo f_i e ir al PASO 6.

Si no existe, ir al PASO 7.

PASO 6.
Si $z_j^i > 0, \forall a_j \notin B^h$, entonces x^h maximiza el objetivo f_i de forma única luego x^h es solución eficiente e ir al PASO 9.

Si $z_j^i = 0$, para algún $a_j \notin B^h$, entonces x^h maximiza el objetivo f_i pero no de forma única, luego x^h puede no ser eficiente. Determinar las soluciones óptimas alternativas introduciendo en la base aquellas columnas con valor cero en la fila i, para ver cuáles de ellas son eficientes e ir al PASO 9 con aquellas que lo sean.

PASO 7.
(Se busca una columna no básica que tenga todas sus entradas de costes reducidos no positivas)
Si para algún $a_j \notin B^h$ se tiene que $z_j^i \leq 0, \forall i = 1, \ldots, k$, entonces ir al PASO 16.

Si no, resolver el subproblema $(P)_E$ para determinar la eficiencia de x^h e ir al PASO 8.

PASO 8.
Si x^h es eficiente, ir al PASO 9.

Si no, ir al PASO 10.

PASO 9.
Almacenar la solución eficiente x^h, hacer $g = g + 1$ e ir al PASO 10.

PASO 10.
(Se busca una columna no básica no dominada)
Si para algún $a_j \notin B^h$ se tiene que $\theta_j z_j^i \leq \theta_q z_q^i, \forall i = 1, \ldots, k, \forall a_q \notin B^h$, entonces ir al PASO 16.

Si no, ir al PASO 11.

PASO 11.
Si x^h es eficiente, entonces ir al PASO 13.

Si no, ir al PASO 12.

PASO 12.
Si $g = 0$, ir al PASO 13.

Si no, ir al PASO 15.

PASO 13.
Si $\exists a_j \notin B^h$ tal que su vector de costos reducidos z_j tiene componentes positivas y negativas, entonces ir al PASO 14.

Si no, ir al PASO 15.

PASO 14.
Almacenar las columnas no básicas obtenidas en el PASO 13 que generan bases todavía no exploradas.

PASO 15.
Si existe alguna columna almacenada que no haya sido considerada anteriormente, entonces ir al PASO 16.

Si no, parar el algoritmo.

PASO 16.
Si se obtiene una nueva base no explorada anteriormente introduciendo la columna a_j en la base, entonces ir al PASO 17.

Si no, ir al PASO 15.

PASO 17.
Hacer $h = h + 1$.
Introducir la columna a_j en la base B^h, pivotar, y volver al PASO 4.

El algoritmo terminará cuando se hayan explorado todas las bases que puedan tener asociadas soluciones básicas eficientes, ya que todos los puntos del conjunto eficiente están conectados entre sí.

■ **Ejemplo 2.2** Se considera el problema biobjetivo del Ejemplo 2.1, al que se le han añadido las variables de holgura x_3, x_4, x_5:

$$\begin{array}{rrrrrrrr} \max & f(x) = (f_1(x), f_2(x)) = (-x_1 + x_2, x_1 + 2x_2) & & & & & & \\ \text{s.a.} & -x_1 & +2x_2 & +x_3 & & & = & 8 \\ & x_1 & +x_2 & & +x_4 & & = & 8 \\ & x_1 & & & & +x_5 & = & 6 \\ & x_1, & x_2, & x_3, & x_4, & x_5 & \geq & 0 \end{array}$$

A continuación se aplica el algoritmo del Simplex para obtener el conjunto eficiente.

PASO 1.
La tabla del Simplex queda como sigue.

$c^2 \rightarrow$	1	2	0	0	0	
$c^1 \rightarrow$	-1	1	0	0	0	
	x_1	x_2	x_3	x_4	x_5	b
x_3	-1	2	1	0	0	8
x_4	1	1	0	1	0	8
x_5	1	0	0	0	1	6
$z^1 \rightarrow$	1	-1	0	0	0	0
$z^2 \rightarrow$	-1	-2	0	0	0	0

PASO 2.
$h = 1$ y $g = 0$

PASO 3.
$B^1 = \{x_3, x_4, x_5\}$

PASO 4.
Solución básica factible asociada a B^1:

$$x^1 = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

PASO 5.
No existe ninguna fila de costes reducidos que tenga todas sus entradas no negativas.

PASO 7.
La columna no básica a_2 tiene todas sus entradas de costes reducidos no positivas.

PASO 16.
Si x_2 entra en la base, entonces
$\theta_2 = min\{\frac{8}{2}, \frac{8}{1}\} = \frac{8}{2} = 4$, luego sale x_3.
Se obtiene una nueva base no explorada.

PASO 17.
$h = h + 1 = 1 + 1 = 2$
$B^2 = \{x_2, x_4, x_5\}$
Pivotar y construir la siguiente tabla del Simplex.

$c^2 \rightarrow$	1	2	0	0	0	
$c^1 \rightarrow$	-1	1	0	0	0	
x_2	$-1/2$	$\boxed{1}$	$1/2$	0	0	4
x_4	$3/2$	0	$-1/2$	1	0	4
x_5	1	0	0	0	1	6
$z^1 \rightarrow$	$1/2$	0	$1/2$	0	0	4
$z^2 \rightarrow$	-2	0	1	0	0	8

PASO 4.
Solución básica factible asociada a B^2:

$$x^2 = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 4 \end{pmatrix}$$

PASO 5.
La fila de costes reducidos z^1 tiene todas sus entradas no negativas, luego x^2 maximiza el objetivo f_1.

PASO 6.
Además, todas las entradas de z^1 correspondientes a columnas no básicas (a_1 y a_3) son estrictamente positivas luego x^2 maximiza el objetivo f_1 de forma única.
Por tanto, x^2 es solución eficiente.

PASO 9.
Almacenar x^2 en el conjunto eficiente: $X_{ef} = \{x^2\}$.
$g = g + 1 = 0 + 1 = 1$.

PASO 10.
Se busca una columna no básica (a_1 o a_3) no dominada, es decir, si $\exists \theta_1, \theta_3$ para las variables no básicas x_1 y x_3, respectivamente, tales que verifiquen

$$\theta_1 z_1^i \leq \theta_3 z_3^i$$

o

$$\theta_1 z_1^i \geq \theta_3 z_3^i$$

para $i = 1, 2$, con al menos una desigualdad estricta.

Se determina el mayor valor posible con el que pueden entrar x_1 y x_3 en la base:

Si x_1 entra en la base, entonces
$\theta_1 = min\{\frac{4}{3/2}, \frac{6}{1}\} = \frac{4}{3/2} = 8/3$, luego sale x_4.

Si x_3 entra en la base, entonces
$\theta_3 = min\{\frac{4}{1/2}\} = \frac{4}{1/2} = 8$, luego sale x_2.

Como

$$\theta_1 z_1^1 < \theta_3 z_3^1 \Longrightarrow 8/3 \cdot 1/2 = 4/3 < 4 = 8 \cdot 1/2$$

y

$$\theta_1 z_1^1 < \theta_3 z_3^1 \Longrightarrow 8/3 \cdot (-2) = -16/3 < 8 = 8 \cdot 1$$

la columna a_1 es no dominada. Por tanto, introducir x_1 en la base proporciona una solución que domina a la solución que se obtiene al introducir x_3 en la base.

PASO 16.
Si a_1 entra en la base y x_4 sale se forma una base nueva no explorada anteriormente.

PASO 17.
$h = h + 1 = 2 + 1 = 3$
$B^3 = \{x_2, x_1, x_5\}$
Pivotar y construir la siguiente tabla del Simplex.

$c^2 \rightarrow$	1	2	0	0	0	
$c^1 \rightarrow$	-1	1	0	0	0	
x_2	0	1	1/3	1/3	0	16/3
x_1	$\boxed{1}$	0	-1/3	2/3	0	8/3
x_5	0	0	1/3	-2/3	1	10/3
$z^1 \rightarrow$	0	0	2/3	-1/3	0	8/3
$z^2 \rightarrow$	0	0	1/3	4/3	0	40/3

PASO 4.
Solución básica factible asociada a B^3:

$$x^3 = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 8/3 \\ 16/3 \end{pmatrix}$$

PASO 5.
La fila de costes reducidos z^2 tiene todas sus entradas no negativas, luego x^3 maximiza el objetivo f_2.

PASO 6.
Además, todas las entradas de z^2 correspondientes a columnas no básicas (a_3 y a_4) son estrictamente positivas luego x^3 maximiza el objetivo f_2 de forma única.
Por tanto, x^3 es solución eficiente.

PASO 9.
Almacenar x^3 en el conjunto eficiente: $X_{ef} = \{x^2, x^3\}$.
$g = g + 1 = 1 + 1 = 2$.

PASO 10.
Se busca una columna no básica (a_3 o a_4) no dominada, al igual que hicimos anteriormente.

Si x_3 entra en la base, entonces
$\theta_3 = min\{\frac{16/3}{1/3}, \frac{10/3}{1/3}\} = \frac{10/3}{1/3} = 10$, luego sale x_5.

Si x_4 entra en la base, entonces
$\theta_4 = min\{\frac{16/3}{1/3}, \frac{8/3}{2/3}\} = \frac{8/3}{2/3} = 4$, luego sale x_1 (esto daría una base ya explorada)

En este caso se tiene

$$\theta_3 z_3^1 > \theta_4 z_4^1 \Longrightarrow 10 \cdot 2/3 = 20/3 > -4/3 = 4 \cdot (-4/3)$$

pero

$$\theta_3 z_3^2 < \theta_4 z_4^2 \Longrightarrow 10 \cdot 1/3 = 10/3 < 16/3 = 4 \cdot 4/3$$

luego ninguna columna domina sobre la otra.

PASO 11.
x^3 sí es eficiente.

PASO 13.
La columna no básica a_4 tiene sus entradas de costes reducidos positivos y negativos.

PASO 14.
No almacenamos la columna a_4 pues genera la base B^2, que ya ha sido explorada.

PASO 15.
Se para el algoritmo.

Se concluye, por tanto, que el conjunto eficiente está formado por los puntos extremos

$$x^2 = \begin{pmatrix} 0 \\ 4 \end{pmatrix}, \; x^3 = \begin{pmatrix} 8/3 \\ 16/3 \end{pmatrix},$$

que corresponden a los puntos A y B, respectivamente, del Ejemplo 2.1, como era de esperar, y todos los puntos del segmento que los une:

$x = \alpha \cdot x^2 + (1 - \alpha) \cdot x^3$, $\alpha \in (0, 1)$.

En la Figura 2.5 se aprecia el camino que recorre el algoritmo por la región factible hasta generar todo el conjunto eficiente.

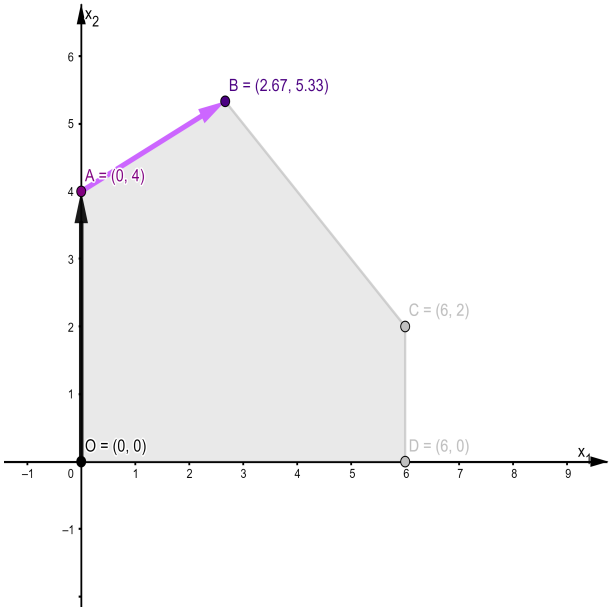


Figura 2.5: Conjunto eficiente. Método Simplex

■

2.3. Método de las ponderaciones

Se considera la siguiente notación para un problema lineal con k funciones objetivo:

$$\begin{array}{c} \max\ f(x) = (f_1(x), \ldots, f_k(x)) \\ \text{s.a. } x \in X \end{array}$$

2.3.1. Descripción del método

Sea el vector $\lambda = (\lambda_1, \ldots, \lambda_k)$, con λ_i el peso asociado al objetivo $f_i(x)$. Consideremos el siguiente problema, denotado P(λ):

$$\begin{array}{c} \max\ p(x) = \sum_{i=1}^{k} \lambda_i f_i(x) \\ \text{s.a. } x \in X \end{array} \tag{2.2}$$

El elemento λ_i se interpreta como la relevancia o el peso relativo que el decisor da al objetivo i-ésimo $f_i(x)$ en relación a los demás. De esta manera, los objetivos tomarán importancia en el problema en orden de preferencia.

El problema Multiobjetivo ha quedado transformado en un problema con un único objetivo p(x) a optimizar. Habiendo asignado los valores de λ de manera razonable y una vez resuelto por alguno de los métodos convencionales, la solución óptima obtenida será directamente la de mejor compromiso para el decisor.

El siguiente teorema da una condición suficiente para que la solución óptima de P(λ) sea eficiente.

Teorema 2.3.1 Sea x^* solución óptima de P(λ). Si $\lambda_i > 0, \forall i = 1, \ldots, k$, entonces x^* es eficiente para el problema Multiobjetivo original.

El recíproco se verifica sólo bajo las hipótesis del siguiente teorema.

Teorema 2.3.2 Sea X un poliedro convexo en $\mathbb{R}^n$. Sean $f_i(x), \forall i = 1, \ldots, k$ las funciones objetivo. Sea x^* solución eficiente. Entonces existen valores $\lambda_i > 0, \forall i = 1, \ldots, k$ para los cuales x^* es solución óptima de P(λ).

■ **Ejemplo 2.3** Se retoma el Ejemplo 2.1, suponiendo que el decisor determina el vector de pesos $\lambda = (\lambda_1, \lambda_2) = (1, 2)$, esto es, da el doble de importancia al segundo objetivo respecto del primero.

El problema uniobjetivo P(λ) queda como sigue:

$$\begin{array}{c} \max\ p(x) = \lambda_1 f_1(x) + \lambda_2 f_2(x) = 1 \cdot (-x_1 + x_2) + 2 \cdot (x_1 + 2x_2) = x_1 + 5x_2 \\ \begin{array}{llcl} \text{s.a.} & -x_1 & +2x_2 & \leq & 8 \\ & x_1 & +x_2 & \leq & 8 \\ & x_1 & & \leq & 6 \\ & x_1, & x_2 & \geq & 0 \end{array} \end{array}$$

Resolviéndolo por el Método Simplex, se obtiene la solución óptima $x^* = \begin{pmatrix} x_1^* \\ x_2^* \end{pmatrix} = \begin{pmatrix} 8/3 \\ 16/3 \end{pmatrix}$, que coincide con el punto extremo x^3 obtenido mediante el algoritmo del Simplex en el Ejemplo 2.2.

Además, por el Teorema 2.3.1, queda asegurada su eficiencia pues el vector de pesos es estrictamente positivo.

En la Figura 2.6 puede verse gráficamente que el punto x^* pertenece al conjunto eficiente.

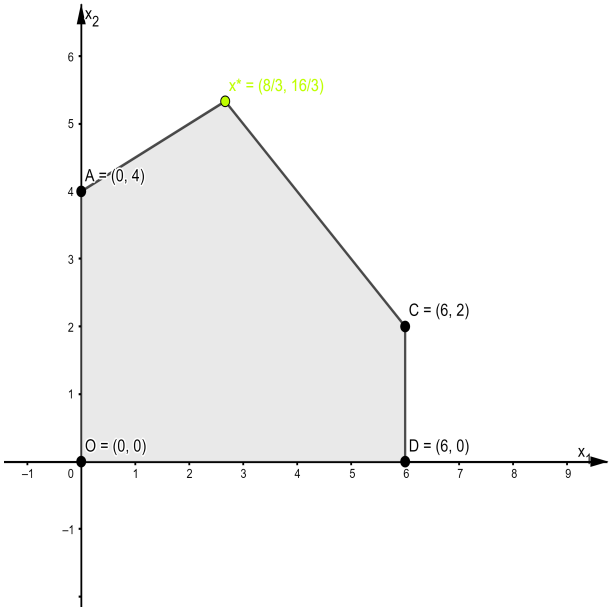


Figura 2.6: Solución eficiente. Método de las ponderaciones

■

2.3.2. Generación del conjunto eficiente

El método de las ponderaciones puede usarse para la generación del conjunto eficiente de la siguiente manera.

Se comienza usualmente considerando los pesos $\lambda = (1, 0, \ldots, 0), (0, 1, \ldots, 0), \ldots, (0, 0, \ldots, 1)$. De esta forma obtendremos k problemas de optimización unidimensionales, uno para cada función objetivo. Seguidamente se hacen variar los valores de λ convenientemente. De esta manera se van generando puntos extremos del conjunto factible X.

No es demasiado adecuado hacer uso del método de las ponderaciones para obtener el conjunto eficiente, pues en muchos casos se obtiene solo una aproximación de este. Surgen inconvenientes, por ejemplo, en el momento en que conjuntos de pesos distintos generan el mismo punto, o este es obtenido a partir de algún peso nulo, pues el Teorema 2.3.1 dice que no puede asegurarse su eficiencia. También puede darse el caso en que no se consigan generar todos los puntos extremos y por tanto se consideren como eficientes los puntos del segmento que une dos puntos no adyacentes.

■ **Ejemplo 2.4** Se considera ahora el problema genérico $P(\lambda)$:

$$\max\ p(x) = \lambda_1 \cdot (-x_1 + x_2) + \lambda_2 \cdot (x_1 + 2x_2)$$
$$\text{s.a. } x \in X$$
$$x \geq 0$$

Si se hace variar sistemáticamente el vector de pesos $\lambda = (\lambda_1, \lambda_2)$ y se resuelven aquellos problemas unidimensionales que se van generando se consigue, al menos, una aproximación del conjunto eficiente. Conviene tomar valores de λ estrictamente positivos para garantizar la eficiencia de las soluciones obtenidas.

(λ_1, λ_2)	(x_1^*, x_2^*)
(1,1)	$(\frac{8}{3}, \frac{16}{3})$ = B
(2,1)	$(\frac{8}{3}, \frac{16}{3})$ = B
(1,3)	$(\frac{8}{3}, \frac{16}{3})$ = B
(3,2)	$(\frac{8}{3}, \frac{16}{3})$ = B
(4,1)	(0,4) = A
(5,1)	(0,4) = A

En base a los resultados se puede intuir que el conjunto eficiente lo forman los puntos del segmento que une A y B, ambos inclusive.

■

2.3.3. Existencia de soluciones óptimas alternativas

Se presenta una estrategia para evaluar la eficiencia de las soluciones óptimas alternativas, en el caso de que se hayan obtenido a partir de algún peso igual a cero.

Se suponen nulos, sin pérdida de generalidad, todos los valores λ_i a partir de un cierto p, esto es,

$$\lambda_i \begin{cases} > 0, \text{ si } i = 1, \ldots, p \\ = 0, \text{ si } i = p+1, \ldots, k \end{cases}$$

Resolviéndose el problema de optimización unidimensional (2.2) con la ponderación escogida, se obtiene una solución alternativa $x^* = (x_1^*, \ldots, x_n^*)$ que verifica

$$f_i(x_1^*, \ldots, x_n^*) = z_i^*,\ i = 1, \ldots, p.$$

Tal solución alternativa se ha obtenido a partir de ciertos pesos nulos luego debe comprobarse su eficiencia. Para ello, se resuelve el siguiente problema de ponderaciones, denotado $P_0(\lambda)$, para un subconjunto de X.

$$\max\ p_0(x) = \sum_{i=1}^{p} \lambda_i f_i(x)$$
$$\text{s.a. } x \in X$$
$$f_i(x_1, \ldots, x_n) = z_i^*, i = 1, \ldots, p$$

Teorema 2.3.3 En las condiciones anteriores, si $x^* = (x_1^*, \ldots, x_n^*)$ es solución óptima de $P_0(\lambda)$, entonces es eficiente.

■ **Ejemplo 2.5** Se supone ahora que el decisor da un vector de pesos $\lambda = (2, 0)$, en el que una de las componentes es nula. Resuelto por el método Simplex el problema del Ejemplo 2.4 para tales valores, se obtiene la solución óptima $x^* = \begin{pmatrix} 0 \\ 4 \end{pmatrix}$, que verifica $f_1(x^*) = 4$.

En este caso sabemos, por lo visto en los ejemplos anteriores, que se trata de un punto eficiente. De forma general, es necesario comprobar la eficiencia de la solución resolviendo el problema de ponderaciones $P_0(\lambda)$:

$$\max\ p_0(x) = 2 \cdot (-x_1 + x_2) + 0 \cdot (x_1 + 2x_2) = -2x_1 + 2x_2$$
$$\text{s.a. } x \in X$$
$$f_1(x) = -x_1 + x_2 = 4$$

Como cabe esperar, x^* es solución óptima luego es eficiente.

■

2.4. Método de las ε-restricciones

Se considera la siguiente notación para un problema lineal con k funciones objetivo:

$$\max\ f(x) = (f_1(x), \ldots, f_k(x))$$
$$\text{s.a. } x \in X$$

2.4.1. Descripción del método

Para aplicar este método, se requiere que una de las funciones objetivo tenga más relevancia para el decisor que el resto. Supongamos $f_r(x)$ tal objetivo.

Se plantea el siguiente problema, denotado $P_r(\varepsilon)$, en el que se busca optimizar el objetivo de mayor importancia y que añade una restricción en forma de cota inferior para cada uno de los demás:

$$\max\ f_r(x)$$
$$\text{s.a. } x \in X$$
$$f_i(x) \geq \varepsilon_i, i = 1, \ldots, r-1, r+1, \ldots, k$$

donde ε_i, $i = 1, \ldots, r-1, r+1, \ldots, k$ son números reales que quedan a elección del decisor.

De esta manera vuelve a obtenerse directamente la solución de mejor compromiso. En el caso en que $P_r(\varepsilon)$ no tenga solución, habrá de reajustarse el problema relajando al menos una de las cotas inferiores impuestas.

El siguiente teorema da una condición suficiente para que la solución óptima sea eficiente.

Teorema 2.4.1 Si la solución óptima x^* de $P_r(\varepsilon)$ es única, entonces x^* es eficiente.

■ **Ejemplo 2.6** Para resolver el problema considerado hasta ahora por el método de las $\varepsilon-$ restricciones se supondrá que el objetivo más importante para el tomador de decisiones es el f_2 a la vez que se impone $\varepsilon_1 = 5/2$ como cota inferior para el objetivo f_2.

Por lo visto previamente, se debe plantear y resolver el problema $P_2(\varepsilon)$:

$$\begin{gathered}\max\ f_2(x) = x_1 + 2x_2 \\ \text{s.a. } x \in X \\ f_1(x) = -x_1 + x_2 \geq 5/2\end{gathered}$$

De nuevo, haciendo uso del Simplex, se obtiene el óptimo $x^* = \begin{pmatrix} 8/3 \\ 16/3 \end{pmatrix}$ que, por ser único, es eficiente (Teorema 2.4.1). Este punto es, de hecho, el punto B visto en el Ejemplo 2.1. ■

2.4.2. Existencia de soluciones óptimas alternativas

En el caso de que existan soluciones óptimas alternativas de $P_r(\varepsilon)$ no puede asegurarse nada acerca de la eficiencia de éstas.

Se presenta una estrategia para comprobar la eficiencia de una solución alternativa obtenida $x^* = (x_1^*, \ldots, x_n^*)$.

Se supone, sin pérdida de generalidad, que los objetivos $f_1, \ldots, f_p$ son aquellos que verifican las restricciones como igualdades en la solución óptima, es decir, $f_i(x^*) = \varepsilon_i, \forall i = 1, \ldots, p$, incluido el objetivo escogido como el más importante (objetivo f_r).

Se resuelve el siguiente problema unidimensional, denotado $P_0(\varepsilon)$, que es análogo al presentado en el método de las ponderaciones, donde los objetivos $f_1, \ldots, f_p$ pasan a ser restricciones:

$$\begin{gathered}\max\ f_r(x) \\ \text{s.a. } x \in X \\ f_i(x_1, \ldots, x_n) = \varepsilon_i, i = 1, \ldots, p\end{gathered}$$

Si $x^* = (x_1^*, \ldots, x_n^*)$ es solución óptima de $P_0(\varepsilon)$, entonces queda comprobada su eficiencia.

Si se escogen adecuadamente tanto la función objetivo f_r a optimizar como las cotas inferiores ε_i $(i \neq r)$, el siguiente teorema asegura que toda solución eficiente es solución del problema $P_r(\varepsilon)$.

Teorema 2.4.2 Sea x^* solución eficiente. Entonces para todo valor de r existen cotas inferiores ε_i $(i \neq r)$ de manera que x^* sea solución óptima de $P_r(\varepsilon)$.

2.4.3. Generación de soluciones eficientes

El siguiente algoritmo conduce a soluciones eficientes.

PASO 1.
Tomar $r = 1$.
Fijar cotas inferiores $\varepsilon_i, i = 1, \ldots, k$, arbitrariamente.

PASO 2.
Obtener la solución óptima x^r de $P_r(\varepsilon)$.

PASO 3.
Si $r = k$, entonces se para el algoritmo.

Si $r < k$, entonces si $\varepsilon_r < f_r(x^r)$; cambiar ε_r por $f_r(x^r)$.
Hacer $r = r + 1$ e ir al PASO 2.

■ **Ejemplo 2.7** Un ejemplo de cómo aplicar el algortimo visto es el siguiente.

PASO 1.
$r = 1$.
$\varepsilon = (\varepsilon_1, \varepsilon_2) = (3, 11)$

PASO 2.
$x^1 = \begin{pmatrix} 3/2 \\ 19/4 \end{pmatrix}$ solución óptima única de $P_1(\varepsilon)$:

$$\begin{gathered} \max\ f_1(x) = -x_1 + x_2 \\ \text{s.a. } x \in X \\ f_2(x) = x_1 + 2x_2 \geq 11 \end{gathered}$$

PASO 3.
$r = 1 < 2 = k$ y $\varepsilon_1 = 3 < 13/4 = f_1(x^1)$ luego cambiar ε_1 por $f_1(x^1)$.

$\varepsilon = (\varepsilon_1, \varepsilon_2) = (13/4, 11)$

$r = r + 1 = 1 + 1 = 2$

PASO 2.
$x^2 = \begin{pmatrix} 3/2 \\ 19/4 \end{pmatrix}$ solución óptima única de $P_2(\varepsilon)$:

$$\begin{gathered} \max\ f_2(x) = x_1 + 2x_2 \\ \text{s.a. } x \in X \\ f_1(x) = -x_1 + x_2 \geq 13/4 \end{gathered}$$

PASO 3.
$r = 2 = k$ luego se para el algoritmo.

El algoritmo ha conducido en dos iteraciones a la misma solución $x^1 = x^2$, que como se observa en la Figura 2.7, pertenece al conjunto eficiente ya conocido.

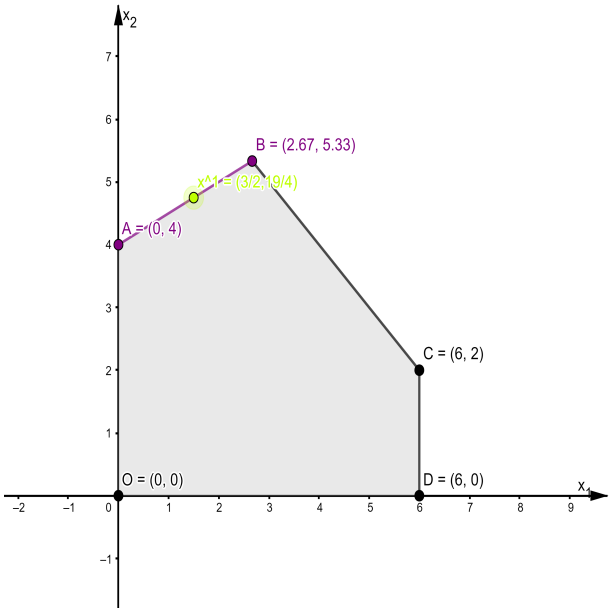


Figura 2.7: Solución eficiente. Método de las $\varepsilon-$ restricciones

■

2.5. Programación por metas

En este método, el tomador de decisiones fija un propósito a alcanzar en cada uno de los objetivos, de manera que se toma como solución óptima aquella que queda "lo más cerca posible" al mismo tiempo de todas las metas prefijadas.

2.5.1. Descripción del método

Se considera el siguiente problema de programación uniobjetivo, denotado $(P)_M$, donde $\tilde{m}_i$ es el nivel de aspiración que el decisor tiene para el objetivo f_i:

$$\begin{aligned} \min\ & d = \sum_{i=1}^{k} |f_i(x) - \tilde{m}_i| \\ \text{s.a. } & x \in X \end{aligned}$$

Se pretende, por tanto, minimizar la suma de las diferencias entre los valores objetivo y sus metas, en valor absoluto.

Hay que tener en cuenta que la función objetivo del problema $(P)_M$ es de carácter no lineal. Alternativamente, puede transformarse en un problema lineal haciendo uso de las siguientes variables:

$$\begin{aligned} d_i^+ &= \tfrac{1}{2}(|f_i(x) - \tilde{m}_i| + (f_i(x) - \tilde{m}_i) \\ d_i^- &= \tfrac{1}{2}(|f_i(x) - \tilde{m}_i| - (f_i(x) - \tilde{m}_i) \end{aligned}$$

que representan las ramas positiva (por exceso) y negativa (por defecto) de las diferencias entre cada valor objetivo y su meta propuesta.

De esta manera, queda el siguiente problema equivalente, denotado $(P')_M$:

$$\begin{array}{lrcl} \min & d' = \sum_{i=1}^{k} (d_i^+ + d_i^-) & & \\ \text{s.a.} & x \in X & & \\ & f_i(x) - d_i^+ + d_i^- & = & \tilde{m}_i \\ & d_i^+, d_i^- & \geq & 0, \quad \forall i = 1, \ldots, k \end{array} \tag{2.3}$$

La restricción $f_i(x) - d_i^+ + d_i^- = \tilde{m}_i$ se conoce como restricción de meta.

Ahora sí que puede aplicarse el método del Simplex para obtener una solución óptima de $(P')_M$ que, en general, no será eficiente para el (MOLP) original, por lo que habrá de comprobarse.

■ **Ejemplo 2.8** Por último, se resuelve de nuevo el Ejemplo 2.1, esta vez haciendo uso de la programación por metas.

Se supone que el decisor fija las metas $\tilde{m}_1 = 3, \tilde{m}_2 = 15$ para los objetivos f_1 y f_2, respectivamente.

El problema $(P)_M$ para estos datos concretos queda

$$\begin{array}{c} \min d = |-x_1 + x_2 - 3| + |x_1 + 2x_2 - 15| \\ \text{s.a. } x \in X \end{array}$$

y haciendo uso de las variables auxiliares

$$\begin{aligned} d_1^+ &= \tfrac{1}{2}(|-x_1 + x_2 - 3| + (-x_1 + x_2 - 3)) \\ d_1^- &= \tfrac{1}{2}(|-x_1 + x_2 - 3| - (-x_1 + x_2 - 3)) \\ d_2^+ &= \tfrac{1}{2}(|x_1 + 2x_2 - 15| + (x_1 + 2x_2 - 15)) \\ d_2^- &= \tfrac{1}{2}(|x_1 + 2x_2 - 15| - (x_1 + 2x_2 - 15)) \end{aligned}$$

se transforma en $(P')_M$

$$\begin{array}{lrrrrrrcl} \min & d' = (d_1^+ & +d_1^-) & +(d_2^+ & +d_2^-) & & & & \\ \text{s.a.} & -x_1 & +2x_2 & & & & & \leq & 8 \\ & x_1 & +x_2 & & & & & \leq & 8 \\ & x_1 & & & & & & \leq & 6 \\ & -x_1 & +x_2 & -d_1^+ & +d_1^- & & & = & 3 \\ & x_1 & +2x_2 & & & -d_2^+ & +d_2^- & = & 15 \\ & x_1, & x_2, & d_1^+, & d_1^-, & d_2^+, & d_2^- & \geq & 0 \end{array} \tag{2.4}$$

Bastaría, por tanto, resolver este problema de Programación Lineal con un solo objetivo para obtener la solución de mejor compromiso. Esto queda pendiente para más adelante. ■

Este método será de utilidad para resolver cualquier problema de Programación Lineal Multiobjetivo mediante Software y se verá en el próximo Capítulo.

Capítulo 3

Resolución con software

Hasta ahora se han manejado ejemplos con un número de variables, restricciones y objetivos muy reducido de forma que resolverlos algebraicamente resulta bastante sencillo.

Sin embargo, los problemas que se presentan en la vida real manejan dimensiones muy grandes. Es por ello que resulta necesario apoyarse en algún software para su resolución.

En este último capítulo se presenta una forma general de resolver problemas de Programación Lineal Multiobjetivo de cualquier dimensión haciendo uso de la Programación por Metas. Para ello se recurrirá a los lenguajes de programación AMPL y R.

3.1. AMPL

Figura 3.1: Logo de AMPL

AMPL (*A Mathematical Programming Language*) es un lenguaje de modelado algebraico para programación matemática: un lenguaje capaz de expresar en notación algebraica problemas de optimización tales como los problemas de programación lineal. Puede descargarse una versión DEMO (limitada) de AMPL en [AMP, a].

Lo que hace a AMPL de gran utilidad es que se definen el modelo y los datos de forma independiente, de manera que es posible resolver problemas distintos con la misma estructura haciendo uso del mismo modelo general y particularizando los conjuntos y parámetros a cada problema en concreto.

Al igual que un problema de Optimización, los modelos definidos en AMPL constan de variables, restricciones y funciones objetivo.

A continuación se describe como queda el problema (2.3), visto en el desarrollo de la Programación por Metas, que da la solución de mejor compromiso. Se asume, sin pérdida de generalidad, que el problema original está en forma canónica:

$$\begin{aligned} & \min d' = \sum_{i=1}^{k} (d_i^+ + d_i^-) \\ \text{s.a} \quad & \mathrm{Ax} \leqq \mathrm{b} \\ & f_i(x) - d_i^+ + d_i^- = \tilde{m}_i \\ & d_i^+, d_i^-, x_i \geq 0, \forall i = 1, \ldots, k \end{aligned}$$

Expresado en lenguaje AMPL quedaría como sigue:

Primero de todo, se definen como números enteros estrictamente positivos los parámetros *número de variables*, *número de restricciones* y *número de funciones objetivo*:

```
param nvar integer, > 0;
param mres integer, > 0;
param nobj integer, > 0;
```

Seguidamente se concretan los conjuntos cuyos elementos son las *variables*, *restricciones* y *objetivos*:

```
set sVAR := 1..nvar;
set sRES := 1..mres;
set sOBJ := 1..nobj;
```

En lo que sigue, lo que se hace es definir la *matriz de objetivos*, C, y *de restricciones*, A, del problema original, así como el *vector de recursos*, b, (o vector lado derecho) y el *vector de metas*, $(\tilde{m}_1, \ldots, \tilde{m}_k)$, cuyas dimensiones quedan especificadas entre corchetes.

```
param mc {sOBJ, sVAR};
param mA {sRES, sVAR};
param vb {sRES};
param vMetas {sOBJ};
```

Una vez denotados todos los parámetros y conjuntos necesarios, se procede a la definición de *variables de decisión*, donde se incluyen las *condiciones de no negatividad*,

```
var dmas {sOBJ} >= 0;
var dmenos {sOBJ} >= 0;
var x {sVAR} >= 0;
```

del *objetivo*,

```
minimize Obj_d:
    sum {k in sOBJ} (dmas[k] + dmenos[k]);
```

y, por último, de las *restricciones*, tanto *del problema original* como las *restricciones de meta*.

```
s.t. rest_A {i in sRES}:
    sum {j in sVAR}  mA[i,j] * x[j] <= vb[i];

s.t. rest_metas {k in sOBJ}:
    ( sum {j in sVAR} mc[k,j] * x[j] ) +
    ( - dmas[k] + dmenos[k] ) = vMetas[k];
```

■ **Ejemplo 3.1** A continuación se muestra como aplicar el modelo diseñado utilizando los

datos del Ejemplo 2.8. Previamente, se expresa el enunciado del problema equivalente en forma matricial:

$$\begin{array}{ll} \min & d' = 0 + 0 + d_1^+ + d_1^- + d_2^+ + d_2^- \\ \text{s.a.} & \end{array}$$

$$\begin{pmatrix} -1 & 2 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & -1 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ d_1^+ \\ d_1^- \\ d_2^+ \\ d_2^- \end{pmatrix} \begin{pmatrix} \leq \\ \leq \\ \leq \\ = \\ = \end{pmatrix} \begin{pmatrix} 8 \\ 8 \\ 6 \\ 3 \\ 15 \end{pmatrix}$$

$$x_1, x_2, d_1^+, d_1^-, d_2^+, d_2^- \geq 0$$

La manera de declarar el fichero de datos es tal y como se muestra a continuación:

Por un lado, se determinan el *número de variables*, el *número de restricciones* y el *número de funciones objetivo*:

```
param nvar:= 2;
param mres:= 3;
param nobj:= 2;
```

Por otro lado, se concretan las entradas de la *matriz de objetivos*

```
param mc : 1 2 :=
1 -1 1
2  1 2;
```

y *de restricciones*,

```
param mA : 1 2 :=
1 -1 2
2  1 1
3  1 0;
```

así como el *vector de recursos*

```
param vb:=
1 8
2 8
3 6;
```

y el *vector de metas*

```
param vMetas:=
1 3
2 15;
```

Como puede observarse en la Figura 3.2, si se lee de derecha a izquierda, en primer lugar, se tiene el fichero **.mod**, el modelo generalizado de resolución de problemas de Programación Lineal Multiobjetivo de cualquier dimensión mediante Programación por Metas. A continuación, en el fichero **.dat**, se particulariza el problema a los datos del Ejemplo 2.8 para ver que, en efecto, se obtienen los mismos resultados que los generados con anterioridad. Por último, en el fichero **.run** se ordena su resolución.

Tras tres iteraciones, la consola ha devuelto los siguientes resultados:

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2.66667 \\ 5.33333 \end{pmatrix}, \; d^+ = \begin{pmatrix} d_1^+ \\ d_2^+ \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \; d^- = \begin{pmatrix} d_1^- \\ d_2^- \end{pmatrix} = \begin{pmatrix} 0.333333 \\ 1.66667 \end{pmatrix}$$

Como cabe esperar, la solución x devuelta pertenece al conjunto eficiente y coincide con el punto B del Ejemplo 2.1 y la solución de mejor compromiso hallada por el Método de las ϵ-restricciones del Ejemplo 2.6.

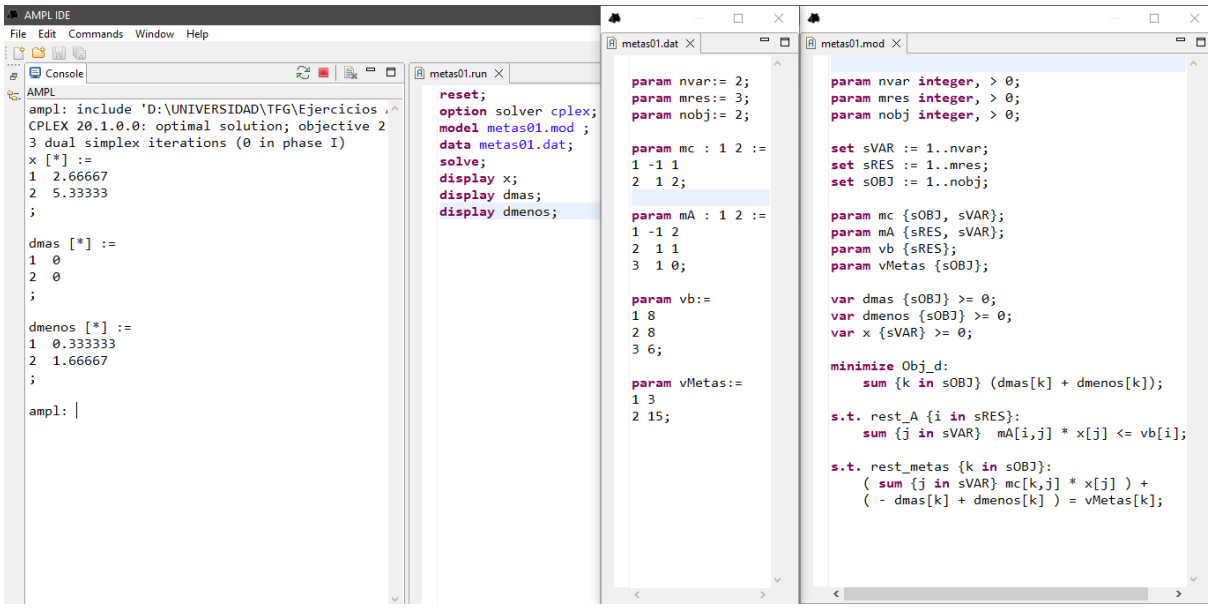


Figura 3.2: Pantalla AMPLide

■

Para poder desarrollar tanto el contenido de esta sección como los ficheros de código se han consultado [Luque-Calvo, 2000] y [AMP, b].

3.2. R

Figura 3.3: Logo de RStudio

El lenguaje de programación R, de libre distribución, proporciona la posibilidad de trabajar con problemas de Optimización Matemática. RStudio, aplicación que facilita el trabajo con R, está disponible libremente para su descarga en [Rst].

Este trabajo destaca dos formas diferentes de resolver un problema de Programación Lineal Multiobjetivo en R, el paquete LpSolve y la librería rAMPL, de nuevo aprovechando el método de Programación por Metas. Para finalizar se hace mención al paquete rMOIP, que brinda la posibilidad de trabajar gráficamente con R y que será de gran utilidad a la hora de representar los conjuntos factibles.

3.2.1. Paquete LpSolve

El paquete LpSolve de R contiene varias funciones para resolver problemas de programación lineal. Entre ellas se encuentra el comando lp, que tiene la siguiente sintaxis:

lp (direction = "min", objective.in, const.mat, const.dir, const.rhs)

Se continúa con el ejemplo anterior para ilustrar como se resolvería dicho problema haciendo uso de esta función de R.

■ **Ejemplo 3.2** Con los datos del Ejemplo 3.1 se procede a resolver mediante el comando lp el problema uniobjetivo procedente de aplicar Programación por Metas.

Se comienza cargando la librería lpSolve.

```
library(lpSolve)
```

Se define la variable donde se almacenarán los resultados y se introducen los argumentos necesarios, empezando por la dirección de optimización, que es de minimización en el caso que ocupa, seguido del vector cuyas entradas son los coeficientes de la función objetivo, la matriz de restricciones, el vector cuyas entradas son las direcciones de las restricciones, en este caso ($\leq$, $\leq$, $\leq$, $=$, $=$) y, para terminar, el vector lado derecho.

```
solucion = lp(direction = "min",
                  objective.in = c(1,1,1,1,0,0),
                  const.mat = matrix(c(
                     0, 0,0,0,-1,2,
                     0, 0,0,0, 1,1,
                     0, 0,0,0, 1,0,
                    -1, 0,1,0,-1,1,
                     0,-1,0,1, 1,2),
                  nrow = 5, ncol = 6, byrow = TRUE),
                  const.dir = c(rep("<=",3), rep("=",2)),
                  const.rhs = c(8,8,6,3,15))
```

Se puede pedir el valor objetivo.

```
solucion$objval
```

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

Con el siguiente comando se devuelve la solución de mejor compromiso.

```
solucion$solution
```

```
## [1] 0.0000000 0.0000000 0.3333333 1.6666667 2.6666667 5.3333333
```

Como puede verse en la Figura 3.4, con esta otra opción alternativa a AMPL se obtienen los mismos resultados.

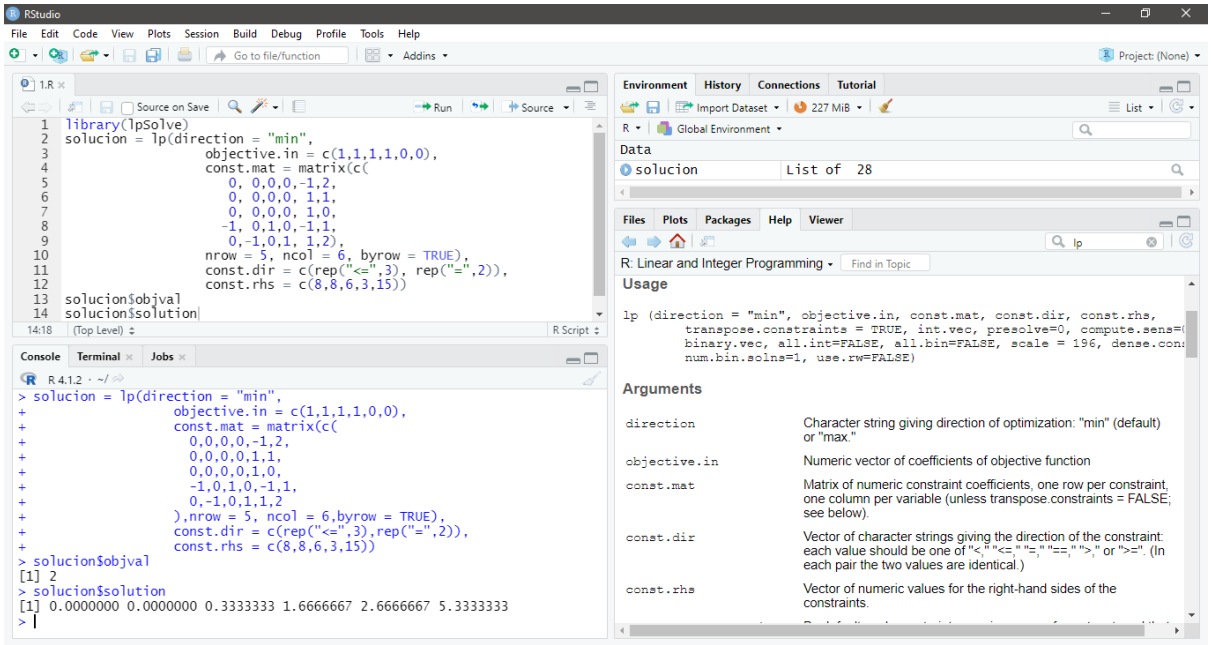


Figura 3.4: Pantalla Rstudio. Uso de lpSolve

■

3.2.2. Librería rAMPL

La segunda herramienta de R que se quiere destacar en este trabajo para resolver problemas de Optimización es el uso de la librería rAMPL. Puede descargarse a través de [rAM, a].

AMPL API es una interfaz que permite acceder a AMPL directamente desde otro lenguaje de programación, como puede ser R. Toda la generación de modelos y la interacción del solver se maneja directamente desde AMPL, lo que conlleva una gran estabilidad y velocidad. La librería actúa únicamente como intermediario, y la sobrecarga adicional (referente a uso de memoria y CPU) depende principalmente de la cantidad de datos que se leen desde AMPL, el tamaño del modelo como tal es irrelevante.

Se proporcionan funciones para asignar datos directamente a los parámetros y conjuntos AMPL, que se pueden utilizar en lugar de los procedimientos normales de lectura de datos AMPL. AMPL API ha sido escrita con la usabilidad en mente, y es fácil acceder a sus funcionalidades desde R, entre muchos otros lenguajes de programación.

■ **Ejemplo 3.3** Veamos cómo se resolvería utilizando la herramienta que proporciona AMPL API.

Se almacena en una variable el modelo que fue desarrollado en el fichero **.mod** entre comillas.

```
modelo = "
param nvar integer, > 0;
param mres integer, > 0;
param nobj integer, > 0;
```

```
set sVAR := 1..nvar;
set sRES := 1..mres;
set sOBJ := 1..nobj;

param mc {sOBJ, sVAR};
param mA {sRES, sVAR};
param vb {sRES};
param vMetas {sOBJ};

var dmas {sOBJ} >= 0;
var dmenos {sOBJ} >= 0;
var x {sVAR} >= 0;

minimize Obj_d:
    sum {k in sOBJ} (dmas[k] + dmenos[k]);

s.t. rest_A {i in sRES}:
    sum {j in sVAR}  mA[i,j] * x[j] <= vb[i];

s.t. rest_metas {k in sOBJ}:
    ( sum {j in sVAR} mc[k,j] * x[j] ) +
    ( - dmas[k] + dmenos[k] ) = vMetas[k];

"
```

Análogamente para los datos que fueron definidos en el fichero **.dat**.

```
data = "
param nvar:= 2;
param mres:= 3;
param nobj:= 2;

param mc : 1 2 :=
1 -1 1
2  1 2;

param mA : 1 2 :=
1 -1 2
2  1 1
3  1 0;

param vb:=
1 8
2 8
3 6;

param vMetas:=
1 3
2 15;
```

```
"
```

Otra opción es guardar en una variable cada uno de los ficheros directamente.

```
fic_ampl_modelo = "metas01.mod"
fic_ampl_data = "metas01.dat"
writeLines(modelo,fic_ampl_modelo)
writeLines(data,fic_ampl_data)
```

Por último, se ordena la resolución mediante los siguientes comandos:

Se llama a la librería

```
library(rAMPL)
```

y se direcciona la ubicación de instalación. Debe ser la misma donde se instaló AMPL.

```
dirinstall_ampl = "/home/rstudio/ampl.linux-intel64/"
```

Mediante el comando new se crea el objeto,

```
ampl = new(AMPL, new(Environment, dirinstall_ampl))
```

se interpretan las variables que contienen a los ficheros,

```
ampl$reset()
ampl$read(fic_ampl_modelo)
ampl$readData(fic_ampl_data)
```

se especifica el solver que se quiere utilizar,

```
ampl$setOption("solver", "cplex")
```

y se ordena resolver.

```
ampl$eval("option cplex_options 'sensitivity';")
ampl$solve()
```

```
## CPLEX 20.1.0.0: sensitivity
## CPLEX 20.1.0.0: optimal solution; objective 2
## 3 dual simplex iterations (0 in phase I)
##
## suffix up OUT;
## suffix down OUT;
## suffix current OUT;
```

Se puede pedir también que imprima por pantalla la solución óptima obtenida

```
ampl$eval("display x,dmas,dmenos;")
```

```
## :       x    dmas     dmenos     :=
## 1   2.66667   0     0.333333
## 2   5.33333   0     1.66667
## ;
```

y el enunciado del problema resuelto.

```
ampl$eval("expand;")
```

```
## minimize Obj_d:
##  dmas[1] + dmas[2] + dmenos[1] + dmenos[2];
##
## subject to rest_A[1]:
##  -x[1] + 2*x[2] <= 8;
##
## subject to rest_A[2]:
##  x[1] + x[2] <= 8;
##
## subject to rest_A[3]:
##  x[1] <= 6;
##
## subject to rest_metas[1]:
##  -dmas[1] + dmenos[1] - x[1] + x[2] = 3;
##
## subject to rest_metas[2]:
##  -dmas[2] + dmenos[2] + x[1] + 2*x[2] = 15;
```

Véase en la Figura 3.5 cómo se obtienen los mismos resultados haciendo uso de la librería rAMPL.

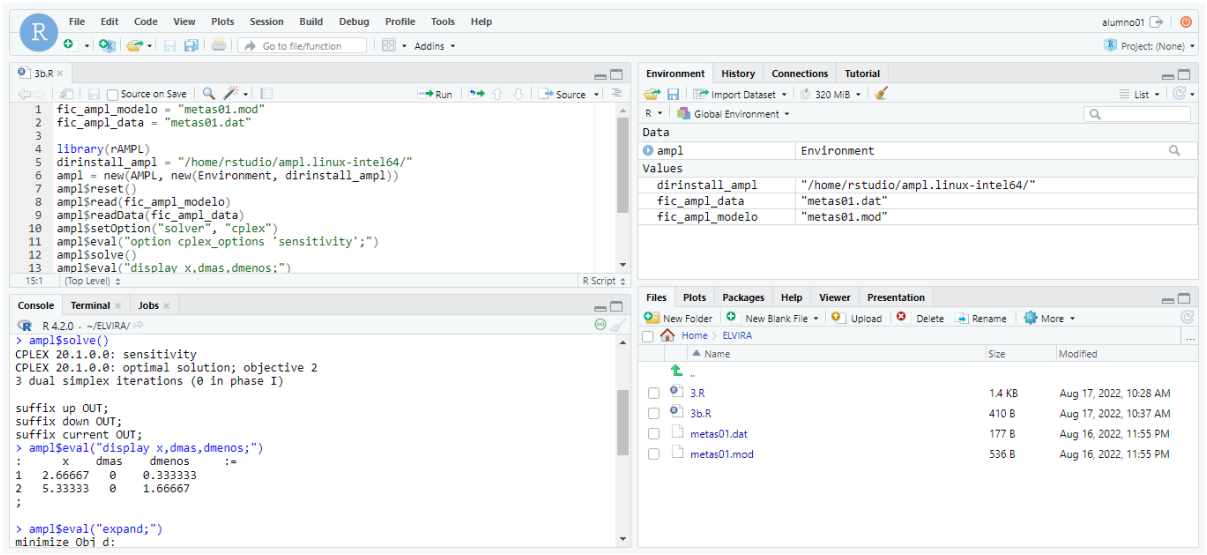


Figura 3.5: Pantalla Rstudio. Uso de rAMPL

■

Para el desarrollo de esta sección y la comprensión de esta librería se ha consultado la información disponible en [rAM, b]. Cabe mencionar que para el uso de rAMPL es necesario tener instalado previamente el paquete "Rcpp".

3.2.3. Paquete rMOIP

Existe un paquete en R, llamado "gMOIP", capaz de trabajar gráficamente (2D y 3D) con problemas lineales, ya sean continuos, enteros o mixtos, y permite hacer representaciones del espacio de decisiones y del espacio de objetivos de problemas bicriterio. Se presenta a continuación su funcionamiento recurriendo de nuevo al Ejemplo 2.1.

■ Ejemplo 3.4 Representación del conjunto factible

Se carga la librería

```
library(gMOIP)
```

Se definen los elementos del problema lineal biobjetivo con 2 variables

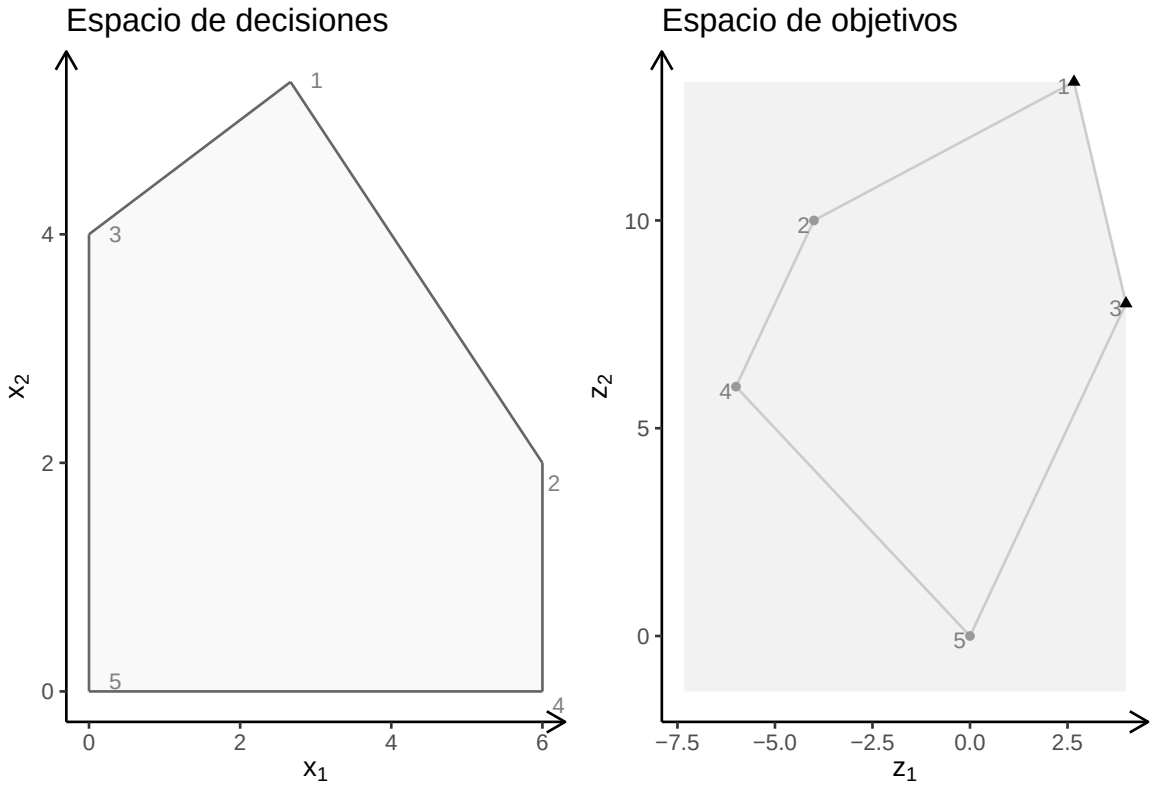
```
#Matriz de restricciones
A <- matrix(c(-1,2,
              1,1,
              1,0), ncol = 2, byrow = TRUE)

#Vector lado derecho
b <- c(8,8,6)

#Matriz de objetivos
obj <- matrix(c(-1,  1,    #primer objetivo
                 1,  2),   #segundo objetivo
               nrow = 2)
```

Se define una nueva función para representarlo

```
plotBiObj2D <- function(A, b, obj,
                        type = rep("c", ncol(A)),  #Variables continuas
                        crit = "max",          #Criterio de optimización
                        faces = rep("c", ncol(A)),
                        plotFaces = TRUE,
                        plotFeasible = TRUE,
                        plotOptimum = FALSE,
                        labels = "numb",
                        addTriangles = TRUE,
                        addHull = TRUE)
{
  p1 <- plotPolytope(A, b, type = type, crit = crit, faces = faces,
                     plotFaces = plotFaces, plotFeasible = plotFeasible,
                     plotOptimum = plotOptimum, labels = labels) +
    ggplot2::ggtitle("Espacio de decisiones")
  p2 <- plotCriterion2D(A, b, obj, type = type, crit = crit,
                        addTriangles = addTriangles, addHull = addHull,
                        plotFeasible = plotFeasible, labels = labels) +
    ggplot2::ggtitle("Espacio de objetivos")
  gridExtra::grid.arrange(p1, p2, nrow = 1)
}
```

Se representa el conjunto factible del problema biobjetivo haciendo uso de la función previamente definida

```
plotBiObj2D(A, b, obj, addTriangles = FALSE)
```



Se observa que la función dibuja tanto la región factible del espacio de decisiones como la región factible del espacio de objetivos.

La representación en dos dimensiones del espacio de objetivos permite identificar fácilmente los vértices eficientes y descartar los que no lo son. Así, en las gráficas obtenidas para este ejemplo, se aprecia que los puntos 1 y 3 en el espacio de objetivos son los objetivos asociados a las soluciones eficientes 1 y 3 (B y A respectivamente en la Figura 2.2) del espacio de decisiones. Esto concuerda con el hecho de que, si se pasa de 1 a 3 en el espacio de objetivos, mientras que el primer objetivo, considerado en el eje X, aumenta (mejora) su valor, el segundo objetivo, considerado en el eje Y, disminuye (empeora) su valor. Luego ambos son puntos no dominados, y sus respectivas soluciones en el espacio de decisiones, eficientes. Sin embargo, si se pasa de 5 a 3, o de 4 a 2, se ve que ambos objetivos aumentan (mejoran). Por tanto 3 domina a 5 y 2 domina a 4, por lo que ni 5 ni 4 serán soluciones eficientes.

Generación de los puntos extremos del conjunto factible

Otra gran utilidad a destacar del paquete gMOIP es la posibilidad de generar los puntos extremos del conjunto factible haciendo uso de la orden cornerPoints.

```
cornerPoints(A, b, type = c("c", "c"))
```

```
##           x1       x2
## [1,] 2.666667 5.333333
## [2,] 6.000000 2.000000
## [3,] 0.000000 4.000000
```

```
## [4,] 6.000000 0.000000
## [5,] 0.000000 0.000000
```

El vector type = c("c", "c") indica que se tienen dos variables continuas.

■

Se ha consultado la documentación disponible en [Nielsen, 2021] y [rMO] referente al uso del paquete rMOIP.

3.3. Conclusiones

El presente trabajo ha tratado de introducir al lector, de una forma gradual, en el ámbito de la Optimización Multiobjetivo; empezando por una primera parte eminentemente teórica, en la que se configura la estructura y sintaxis del modelo general; seguido de una parte central, en la que se exponen algunas de las distintas formas de abordar los problemas planteados en la parte precedente. Además, todo esto acompañado de ejemplos para favorecer la comprensión.

Mediante el planteamiento de algunas situaciones reales que hacen uso de ella, se ha pretendido ilustrar la transversalidad que tiene la Programación Multiobjetivo en cualquier disciplina humana.

Como bien es sabido, a día de hoy, la Programación Matemática no se entiende sin el soporte de un Software que facilite el manejo de grandes volúmenes de datos. Es por ello que, a modo de cierre de este trabajo, se han querido presentar algunas de las diversas herramientas de programación mediante Software disponibles, AMPL y R, para ejemplificar, de una manera tangible, lo que puede ser su uso a grandes escalas.

El contenido de este trabajo abre las puertas al estudio de otras variantes de problemas Multiobjetivo, como pueden ser los de carácter no lineal, o aquellos que involucran variables enteras o mixtas.

Apéndice A

Apéndice: Comprobación de la eficiencia de una solución

A modo de Apéndice se ha querido implementar con Software el problema (2.1) para comprobar si una solución es eficiente o no para un problema Multiobjetivo, visto en el Método Simplex.

Se recuerda su estructura

$$\begin{aligned} \max \quad & E = \sum_{i=1}^{k} h_i \\ \text{s.a.} \quad & x \in X \\ & f_i(x) - h_i = f_i(\bar{x}), \quad i = 1, \ldots, k \\ & h_i \geq 0, \quad i = 1, \ldots, k \end{aligned}$$

donde h_i son las variables de holgura consideradas.

Como ya se vio, si $E = 0$, entonces la solución básica factible $\bar{x}$ es eficiente para el problema multiobjetivo, y si $E > 0$, entonces no es posible asegurar la eficiencia de $\bar{x}$.

Se procede con la definición de la función para unos datos con dimensiones y valores genéricos. En el desarrollo de esta, se recurre al paquete "lpSolve" visto en la sección 3.2.1.

```
func_Comprobar_eficiencia_PLMult =
  function(mc, mA, vb, vdes, v_pto_c_Efi){
    num_obj = nrow(mc)                    #Número objetivos
    num_filas_orig = nrow(mA)             #Número restricciones

    criterio = "max"                      #Criterio de optimización

    mDmenos = - diag(num_obj)
    mA_amp_01 = cbind(mA,diag(0, nrow = num_filas_orig, ncol = num_obj))
    mA_amp_02 = cbind(mc, mDmenos)
    mA_amp = rbind(mA_amp_01, mA_amp_02)  #Nueva matriz restricciones

    vdes_amp = c(vdes,rep("=",num_obj))   #Nuevo vector desigualdades

    v_obj_x = as.numeric(mc %*% v_pto_c_Efi)
```

```
    vb_amp = c(vb,v_obj_x)                  #Nuevo vector lado derecho

    vc_amp = c(rep(0,ncol(mc)), rep(1,num_obj)) #Nuevo vector objetivos

    #Resolución del problema uniobjetivo mediante el paquete lpSolve
    library(lpSolve)
    solucion = lp(direction = criterio,
                  objective.in = vc_amp,
                  const.mat = mA_amp,
                  const.dir = vdes_amp,
                  const.rhs = vb_amp)

    if (solucion$objval == 0) {
      mensaje = "AVISO: SÍ es una solución Eficiente!!"} #E = 0
    else {
      mensaje = "AVISO: NO es una solución Eficiente!!"} #E > 0

    print(mensaje)
}
```

■ **Ejemplo A.1** Ahora se concretarán los datos para el Ejemplo que se ha usado a lo largo de todo el trabajo.

En primer lugar, se definen los elementos del problema multiobjetivo original.

```
mA = matrix(c(-1,2,                         #Matriz de restricciones
              1,1,
              1,0), nrow = 3, byrow = T)
vb = c(8,8,6)                               #Vector lado derecho

vdes = c("<=","<=","<=")                    #Vector de desigualdades

mc = matrix(c(-1,1,                         #Matriz de objetivos
              1,2), nrow = 2, byrow = T)
```

Tal y como se vio, este par de puntos generados por el método del Simplex son eficientes, pues son óptimos para cada uno de los objetivos por separado. Esto se comprobó en el Ejemplo 2.2 viendo que la fila de costes reducidos asociada al objetivo en concreto tenía todas sus entradas no negativas.

```
#Coincide con el punto x_2 y maximiza el objetivo f_1.
pto_efi_01 = c(0,4)

#Coincide con el punto x_3 y maximiza el objetivo f_2.
pto_efi_02 = c(8/3,16/3)
```

Haciendo uso de la función previamente definida, se ve que ambos son eficientes.

```
func_Comprobar_eficiencia_PLMult(mc, mA, vb, vdes,
                                 v_pto_c_Efi = pto_efi_01)
```

```
## [1] "AVISO: SÍ es una solución Eficiente!!"
```

```
func_Comprobar_eficiencia_PLMult(mc, mA, vb, vdes,
                                 v_pto_c_Efi = pto_efi_02)
```

```
## [1] "AVISO: SÍ es una solución Eficiente!!"
```

A continuación se puede ver que, efectivamente, los puntos de la cara del poliedro que une los puntos anteriores son eficientes. Se comprueba dando valores al parámetro alfa.

```
alfa = 0.6 #0.2, 0.3, 0.4, 0.5, 0.9, 1

func_Comprobar_eficiencia_PLMult(mc, mA, vb, vdes,
                v_pto_c_Efi = alfa * pto_efi_01 + (1-alfa)*pto_efi_02)
```

```
## [1] "AVISO: SÍ es una solución Eficiente!!"
```

Sin embargo, las filas de costes reducidos para el siguiente punto no tenían todas sus entradas estrictamente positivas luego es necesario comprobar su eficiencia.

```
pto_01 = c(0,0) #Coincide con el punto x_1
```

Como puede verse, el punto no es eficiente para el problema multiobjetivo.

```
func_Comprobar_eficiencia_PLMult(mc, mA, vb, vdes,
                                 v_pto_c_Efi = pto_01)
```

```
## [1] "AVISO: NO es una solución Eficiente!!"
```

De igual modo puede comprobarse la eficiencia de los vértices restantes del conjunto factible.

```
pto_02 = c(6,0)
pto_03 = c(6,2)
```

En este caso, tampoco se trata de soluciones eficientes.

```
func_Comprobar_eficiencia_PLMult(mc, mA, vb, vdes,
                                 v_pto_c_Efi = pto_02)
```

```
## [1] "AVISO: NO es una solución Eficiente!!"
```

```
func_Comprobar_eficiencia_PLMult(mc, mA, vb, vdes,
                                 v_pto_c_Efi = pto_03)
```

```
## [1] "AVISO: NO es una solución Eficiente!!"
```

■

Bibliografía

Página de descarga de AMPL versión DEMO. Disponible en https://ampl.com/try-ampl/download-a-free-demo/, a.

Recursos gratuitos de AMPL para estudiantes. Disponible en https://ampl.com/products/ampl/ampl-for-students/#Demo, b.

Página Web del Grupo Español de Decisión Multicriterio. Disponible en http://vps155.cesvima.upm.es/index.htmltop.

Página Web de la Sociedad Internacional de la Toma de Decisiones Multicriterio. Disponible en http://www.mcdmsociety.org/content/mission-society.

Página Oficial de descarga de Rstudio. Disponible en https://www.rstudio.com/products/rstudio/.

Página de descarga de rAMPL. Disponible en https://github.com/ampl/rAMPL, a.

Documentación sobre rAMPL. Disponible en https://rampl.readthedocs.io/en/latest/, b.

Paquete rMOIP. Ejemplos de uso. Disponible en https://github.com/relund/gMOIP.

JJ Allaire, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. *rmarkdown: Dynamic Documents for R*, 2022. https://github.com/rstudio/rmarkdown.

M. Ehrgott, J. Puerto, and A.M. Rodríguez-Chía. Primal-Dual Simplex Method for Multiobjective Linear Programming. *Springer Science+Business Media*, 1953.

Matthias Ehrgott. *Multicriteria Optimization*. Springer Berlin Heidelberg, 2005.

Arthur M. Geoffrion. Proper efficiency and the theory of vector maximization. *Journal of Mathematical Analysis and Applications*, 22, 1968.

M. Arana Jiménez and A. Burgos Pintos. *Optimalidad en programación matemática multiobjetivo. Software*. Servicio de Publicaciones de la Universidad de Cádiz, 2014.

Dinh The Luc. *Multiobjective Linear Programming. An Introduction*. Springer International Publishing AG Switzerland, 2016.

Pedro L. Luque-Calvo. *Lenguaje AMPL*, 2000.

Sebastián Soler Morales. Programación Multiobjetivo. Caso práctico aplicado a una compañía aérea. Disponible en https://digitum.um.es/digitum/bitstream/10201/40606/1/TrabajoRC5.pdf, 2014.

Lars Relund Nielsen. *Tools for 2D and 3D Plots of Single and Multi-Objective Linear/Integer Programming Models*, 2021.

Sixto Ríos Insúa. *Investigación operativa: programación lineal y aplicaciones*. Madrid: Centro de Estudios Ramón Areces, 1996.

Sixto Ríos Insúa. *Programación lineal y aplicaciones: ejercicios resueltos*. Madrid : RaMa, 1997.

David Sonderman and Philip G. Abrahamson. Radiotherapy Treatment Design Using Mathematical Programming Models. *informs*, 33, 1985.

María Lopez Valdivia. Programación Lineal Multiobjetivo aplicada al mundo empresarial. Disponible en https://tauja.ujaen.es/bitstream/10953.1/6975/1/TFG_Maria_Lopez_Valdivia.pdf#:~:text=La%20Programaci%C3%B3n%20Lineal%20Multiobjetivo%20%28PLMO%29%20o%20tambi%C3%A9n%20conocida,al%20problema%20m%C3%A1s%20de%20un%20objetivo%20que%20optimizar., 2016.

Universidad Valencia. Convexidad. Conceptos básicos. Disponible en https://pages.uv.es/sala/convexidad.pdf.

Hadley Wickham, Winston Chang, Lionel Henry, Thomas Lin Pedersen, Kohske Takahashi, Claus Wilke, Kara Woo, Hiroaki Yutani, and Dewey Dunnington. *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*, 2022a. https://ggplot2.tidyverse.org.

Hadley Wickham, Romain François, Lionel Henry, and Kirill Müller. *dplyr: A Grammar of Data Manipulation*, 2022b. https://dplyr.tidyverse.org.

Yihui Xie. *knitr: A General-Purpose Package for Dynamic Report Generation in R*, 2022. URL https://yihui.org/knitr/. R package version 1.39.