Optimización bi-objetivo aplicada a la formulación de piensos compuestos

Marta Rubio Latasa

Trabajo de Fin de Grado en Matemáticas

Directores del trabajo: Pedro Mateo Collazos
Joaquín Surra Muñoz

28 de junio de 2017

Abstract

This project is a continuation and extension of the Final Degree Project (FDP) "*Development of a Computer Program for Compound Feeds Formulation*". It belongs to the Operational Research field, in particular to the Multi-Objective Lineal Programming area and its application to the development of a Java code to formulate compound feeds.

The main goal of this project is to implement an additional module in the program created for the previous FDP. It will permit to construct a formulation bi-objective of compound feeds. Previous knowledge of multi-objetive optimization has been required, especially of lineal bi-objetive optimization. It will be applied to design and implement the program.

First chapter corresponds to the introduction. Second and third chapters discuss the basic foundations of multi-objective optimization in general, and particularly the lineal bi-objective optimization. They are necessary to implement the new module of the program. Fourth chapter describes the program and its design. Finally, in Chapter 5 it is founded a real example of bi-objective formulation of compound feeds, and some conclusions are included at the end of this chapter too. The mathematical approach of the real example and the Java code created are included as two appendices.

In the Introduction, the importance of compound feeds formulation for monogastric animals and the main factors of this formulation, as how to satisfy the animals needs while minimizing the cost and restricting the proteins use, are explained, in addition it clarifies why it is necessary to limit the proteins in the feeds. Then, it explains the starting point and the goals of the project, as gaining knowledge of multi-objective optimization and applying it to a software program while dealing with a real problem.

Chapter 2 details some relevant results of multi-objective optimization. It starts with a reminder of what the uni-objective optimization and its problems are, and then it extends it to the multi-objective case. In addition, it is defined the concept of efficiency, that it is used to consider an optimum solution of this kind of problems. Finally, some methods (weighted sum and ε-constraint) are explained to obtain some efficient solutions. Some prominent theorems and its demonstrations or some simple examples of how to use these methods are detailed too.

Chapter 3 deals with the lineal problems, which are the interesting ones for the feeds formulation. In this case, the method used for obtaining solutions is the weighted sum method. Because of this, some lemmas and theorems ensuring that all the efficient solutions of a lineal multi-objective problem are found, together with their demonstrations, are detailed. Then, the Simplex algorithm for bi-objetive problems is presented together with a simple example solved by hand.

Chapter 4 is focused on the development of the program. First, it explains briefly how the previous version of the software worked. That version optimized the solution just with the criterion of minimizing the cost. Then, it explains how the new version works. This version can determine the bi-objective formulation that maximizes a specific nutrient as a second criterion. Finally, it describes how the Java code has been done, explaining the constructor and the new class methods added by using the library GLPK. The Java code of the created class is in Appendix B.

Chapter 5 shows an example of a real diet for chicken. It is solved by using the new module and it shows some efficient solutions. The mathematical approach can be consulted in Appendix A. Finally, it contains the conclusions obtained at the end of this project.

Índice general

Capítulo 1

Introducción

La importancia económica del sector de fabricación de piensos en España y Aragón es trascendental, de hecho España es el segundo país en producción de piensos en la UE-28 (con 23.325.771 toneladas de piensos) por detrás de Alemania y en volumen de facturación (que alcanzan los 890 y 6.540 millones de euros respectivamente) [2]. Sin embargo, este sector industrial genera un producto con escaso valor añadido, por lo que para que resulte rentable, debe ajustarse a una economía de escala, en la que se requieren grandes volúmenes de dinero. Como medio de reducir los costes producción, para ser competitivos, una de las herramientas disponible es la optimización en la formulación de los piensos. En este sentido, un programa informático capaz de formular piensos al mínimo coste es fundamental en una fábrica de piensos o en una granja que fabrique sus propios piensos. Pero el coste no es el único factor a tener en cuenta, otro factor determinante que afecta expresamente al sector de la fabricación de piensos, es el medioambiental, esto es debido a la contaminación ambiental por exceso de Nitrógeno (N) en los excretas (purines), aspecto en el que la legislación europea se torna cada vez más exigente, lo cual demanda una mayor precisión en la formulación de los piensos y acompañado de una reducción de los niveles de proteína de los mismos. Se sabe que el N es uno de los elementos esenciales en la composición de los aminoácidos constituyentes de las proteínas, por tanto, cuanto mas eficiente y menor sea el uso de aminoácidos, menor será la excreción de N en heces y orina de los animales [10][11]. Hay que considerar también que de los nutrientes que integran los piensos, aquellos que aportan mayor contenido en proteína son los más caros, por lo que interesa indudablemente reducir la inclusión de materias primas ricas en proteínas, es decir será importante optimizar la utilización de ciertos aminoácidos esenciales como Lisina, Metionina, Triptófano, Treonina o Valina.

Unido a los dos elementos anteriores hay que tener en cuenta que se espera que la demanda mundial de alimentos aumente substancialmente. Según la Organización de las Naciones Unidas para la Agricultura y la Alimentación (FAO), para el año 2050 habrá un aumento del 73 % en el consumo de carne y huevo, y un aumento del 58 % en el consumo de productos lácteos en comparación con los niveles de 2011, sobre todo propiciado por el incremento de necesidades en Asia, Africa mayoritariamente y América Latina y el Caribe en menor media.

Como se aprecia en la figura 1.1, la demanda de proteína de origen animal se incrementará un 35 % en el período 2015/2035, de ellas, las proteínas procedentes de animales monogástricos (aves y cerdos) serán las que mayor incremento tendrán. Debido a que las producciones con estos animales se consiguen en sistemas intensivos en los que la base de su alimentación son los piensos compuestos, necesariamente estos últimos también se verán incrementados en su demanda de producción [10][11].

1.1. Planteamiento del trabajo

Este trabajo es una continuación y ampliación del trabajo fin de grado (TFG) "Desarrollo de un Programa Informático para la Formulación de Piensos Compuestos"defendido el 24 de Junio de 2016 por Dña. Andrea Aguilar Ibáñez. El objetivo principal de este era *"...el de desarrollar un programa informático destinado a la formulación de piensos compuestos de animales monogástricos (cerdos y aves)*

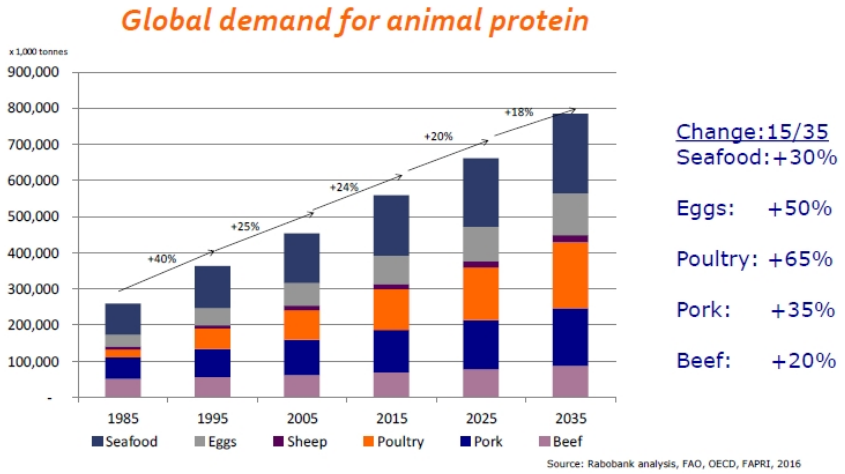

Figura 1.1: Previsión de la demanda mundial (2015/2035) de proteína de origen animal [7].

suficientemente potente y versátil para poder ser utilizado para fines docentes, en pequeñas empresas, ganaderías, etc.,..."[1]. En esta primera versión del software, la formulación del pienso se realizaba atendiendo a un único criterio, el coste económico (mínimo coste), al que se le añadían unas restricciones (máximos y mínimos) de ingredientes y nutrientes. La ampliación que se propone en este trabajo permitirá la inclusión de un segundo criterio de selección, adicional al coste, éste podrá ser cualquiera de los nutrientes presentes en los ingredientes, si bien, desde el punto de vista del problema real éste será habitualmente el nivel de ciertos tipos de aminoácidos: Lisina, Metionina, Triptófano, etc.

1.2. Objetivos del trabajo

A pesar de que la primera experiencia en la formulación y diseño de dietas resultó satisfactoria [1], las necesidades reales existentes en un sector económico muy competitivo, hacen necesario dotar a dicha aplicación de otras funcionalidades que la hagan más completa. La ampliación propuesta en este trabajo consiste en introducir un nuevo módulo que permita ajustar la formulación con dos objetivos, es decir, incluir al objetivo ya presente de minimizar coste, un nuevo criterio de selección adicional, lo que supone enfrentarse a un problema multi-objetivo, siendo ésta cuestión laboriosa ya que durante el grado no se han abordado los estudios de este tipo de modelos.

Por tanto, la propuesta actual tiene como objetivos generales:

- Obtener y explicar los conocimientos teóricos mínimos que permitan entender, modelar y resolver este tipo de problemas multi-objetivo.
- Plasmar estos nuevos conocimientos en el software previamente desarrollado. Para ello se programará un módulo adicional para gestionar la incorporación de un segundo criterio de selección (este segundo objetivo puede ser un nutriente esencial y limitante como la Metionina, que es el primer aminoácido limitante en aves y segundo o tercero en porcino [9], aunque podría ser cualquier otro nutriente de interés para el nutricionista) y su posterior resolución y presentación visual de resultados.

Hay que destacar que en el trabajo se aúnan los nuevos conocimientos y competencias adquiridos en optimización multi-objetivo junto con los conocimientos previamente adquiridos en el grado como son: programación lineal (Investigación Operativa), programación en Java (Informática II) y modelado de problemas (Investigación Operativa, Grafos y Combinatoria, Modelización matemática, etc.). A todo ello hay que añadir el hecho de haberme enfrentado a un problema real que se sale del ámbito académico y que pertenece a un área de conocimiento totalmente ajeno a mi formación.

Capítulo 2

Optimización Multi-Objetivo

Como se ha comentado en la Introducción, en este trabajo se van a utilizar los Problemas de Optimización Multi-Objetivo para modelar la formulación de piensos compuestos. Debido a que este tipo de técnicas no se han estudiado en el grado, en este capítulo, se realizará una presentación detallada de sus resultados más relevantes junto a sus demostraciones.

La optimización matemática, es una herramienta muy utilizada en numerosos ámbitos científicos y en particular es una herramienta clave en la Investigación Operativa. Optimizarconsiste en determinar el valor de unas variables (las variables de decisión) que intervienen en un problema con el fin de encontrar la mejor solución posible de este. El objetivo principal es minimizar o maximizar una función (la función objetivo), cuyas variables están sujetas a unas restricciones, las cuales debe cumplir, es decir:

$$\begin{array}{lll} \textit{minimizar/maximizar} & f(x_1,\ldots,x_n) & \\ \textit{sujeto a}: & g_i(x_1,\ldots,x_n) \quad \leq / = / \geq \quad 0 \qquad \textit{con} \qquad i=1,\ldots,m \\ & (x_1,\ldots,x_n) \in \mathcal{R} \subseteq \mathbb{R}^n & \end{array} \tag{2.1}$$

donde:

- n es el número de variables y m es el número de restricciones.
- $x_1,\ldots,x_n$ son las variables de decisión,
- $f(x_1,\ldots,x_n)$ es la función objetivo,
- cada una de las $g_i(x_1,\ldots,x_n) \ \leq / = / \geq \ 0$ es una restricción del sistema.

Además, las variables deben de pertenecer a $\mathcal{R}$, un subconjunto no vacío de $\mathbb{R}^n$.

Definición. Se denomina región factible en el espacio de decisiones a $X = \{\mathbf{x} \in \mathbb{R}^n : g_i(\mathbf{x}) \leq / = / \geq 0,\ \ i = 1,\ldots,m,\ \ \mathbf{x} \in \mathcal{R}\}$ y su imagen en el espacio de objetivos es $Y = \{y = f(\mathbf{x}) \in \mathbb{R} : \mathbf{x} \in X\}$.

Antes de extender esta noción al ámbito multi-objetivo, se va a presentar un sencillo ejemplo. Este muestra cómo modelar un problema de formulación de piensos con el único objetivo de minimizar el coste, proporcionando los nutrientes necesarios:

$$\begin{array}{ll} \textit{min} & 0.1086x_1 + 0.1483x_2 \\ s.a: & r_1 : x_1 + x_2 = 1 \\ & r_2 : 86 \leq 89.9x_1 + 86.4x_2 \leq 100 \\ & r_3 : 2700 \leq 2780x_1 + 3285x_2 \leq 2900 \\ & c_1 : 0 \leq x_1 \leq 1.0 \\ & c_2 : 0.2 \leq x_2 \leq 1.0 \end{array}$$

En este problema se ha planteado la obtención de un pienso que requiere entre un 0 % y un 100 % (c_1) de "cebada 2C 11.3 PB" (variable x_1), y entre un 20 % y un 100 % (c_2) de "maíz nacional" (variable x_2).

La composición de nutrientes que necesita es entre un 86 % y un 100 % de "materia seca" (restricción r_2), y entre 2700 Kcal/Kg y 2900 Kcal/Kg de "energía en aves" (restricción r_3). Los coeficientes que se utilizan en la función objetivo corresponden a los costes por kilogramo de cada ingrediente y en las restricciones a la cantidad de nutrientes de cada tipo que tiene cada ingrediente. La restricción r_1 garantiza que entre los dos ingredientes se alcance el 100 %.
La solución óptima de este problema se obtiene con un 80 % de cebada y un 20 % de maíz, y cuyo coste es de 0.1166 euros por kilogramo de pienso. Este compuesto tiene un 89.2 % de materia seca y 2881Kcal/Kg de energía, valores pertenecientes al rango adecuado.

2.1. ¿Qué es la Optimización Multi-Objetivo?

En la vida real, cuando se tiene que tomar una decisión, rara vez se toma atendiendo a un único objetivo, por ejemplo, si una persona desea comprarse un coche, obviamente tendrá en cuenta el precio, pero también le interesarán otras cuestiones como el consumo, los elementos de seguridad o la garantía. Como se puede intuir, estos criterios en general van a ser de carácter conflictivo (mejorar cada uno de ellos afectará de forma negativa a los otros). Por tanto, de forma natural surgen problemas que encajan en el campo multi-objetivo, es decir, Problemas de Optimización Multi-Objetivo (POMO's). En términos matemáticos se puede plantear como:

$$\begin{array}{lll} \text{mín} \,/\, \text{máx} & \mathbf{f}(x_1,\ldots,x_n) = (f_1(x_1,\ldots,x_n),\ f_2(x_1,\ldots,x_n),\ \ldots,\ f_p(x_1,\ldots,x_n))^T & \\ s.a: & g_i(x_1,\ldots,x_n) \quad \leq / = / \geq \quad 0, \qquad i = 1,\ldots,m & \\ & (x_1,\ldots,x_n) \in \mathcal{R} \subseteq \mathbb{R}^n & \end{array} \tag{2.2}$$

A continuación se analiza un sencillo ejemplo en el que se puede observar el carácter conflictivo de los objetivos[1]. La región factible en el espacio de decisiones y su imagen en el espacio de objetivos se muestran en la figura 2.1.

$$\begin{cases} min & 2x_1 + x_2 \\ max & x_1 + x_2 \end{cases} \quad \Rightarrow \quad min \begin{cases} 2x_1 + x_2 \\ -x_1 - x_2 \end{cases}$$

$$s.a: \begin{cases} x_1 + 2x_2 \leq 6 \\ x_1 - x_2 \geq 0 \\ x_1, x_2 \geq 0 \end{cases} \quad \Rightarrow \quad \begin{cases} x_1 + 2x_2 - 6 \leq 0 \\ -x_1 + x_2 \leq 0 \\ x_1, x_2 \geq 0 \end{cases}$$

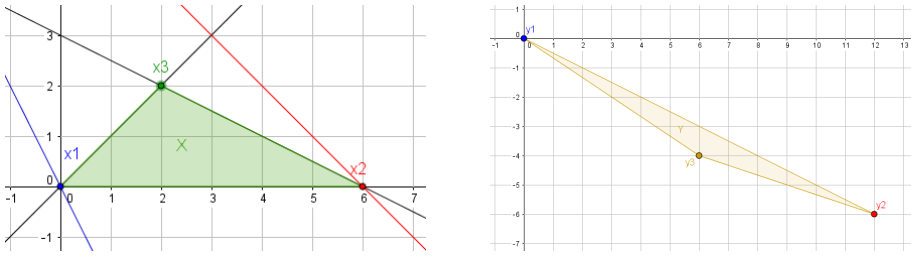

Figura 2.1: Región factible en el espacio de decisiones e imagen en el espacio de objetivos.

Al realizar la optimización de las funciones objetivo por separado se obtiene que la primera (azul) tiene su punto óptimo en el (0,0), dando valores 0 y 0 a las funciones objetivo; sin embargo, la segunda (rojo) tiene su punto óptimo en el (6,0), dando valores 12 y -6 a las funciones objetivo. Se observa que al evaluar el punto óptimo de una función en la contraria no sólo se obtiene un resultado distinto al óptimo, sino que, en este caso, es el peor posible que puede tomar esta función objetivo dentro de la región factible. Por ello parece difícil determinar cual es la solución óptima, pues no existe un orden

[1] Sin pérdida de generalidad se asumirá a lo largo de todo el documento que los problemas son de mínimo y las restricciones de tipo $g_i(x_1,\ldots,x_n) \leq 0 \ \ \forall i = 1,\ldots,m$

total en las soluciones factibles. Por lo tanto, la primera cuestión importante e imprescindible para poder resolver POMO's es determinar lo que se entenderá por solución óptima en una situación en la que no es posible ordenar las potenciales soluciones del problema.

2.2. Soluciones eficientes

Dado el inconveniente que se acaba de comentar, se va a proporcionar una definición o aproximación de lo que se considerará una solución óptima. La aproximación más utilizada consiste en utilizar el concepto de *eficiencia* y, en particular, en esta memoria se introducirán los conceptos de *solución eficiente* y *débilmente eficiente*.

Definición. Dadas dos soluciones $\hat{\mathbf{x}}, \mathbf{x} \in X \subseteq \mathbb{R}^n$ tal que $f_k(\hat{\mathbf{x}}) \leq f_k(\mathbf{x})$, $\forall k = 1, \ldots, p$ con $f_i(\hat{\mathbf{x}}) < f_i(\mathbf{x})$ para algún $i \in \{1, \ldots, p\}$, se dirá que $\hat{\mathbf{x}}$ *domina* a $\mathbf{x}$ (o $\mathbf{x}$ está *dominada* por $\hat{\mathbf{x}}$) y se denotará como $\hat{\mathbf{x}} \prec \mathbf{x}$. Siendo $\mathbf{f}(\hat{\mathbf{x}}) = (f_1(\hat{\mathbf{x}}), \ldots, f_p(\hat{\mathbf{x}}))^T, \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_p(\mathbf{x}))^T \in Y \subseteq \mathbb{R}^p$, también se dice que el punto $\mathbf{f}(\hat{\mathbf{x}})$ *domina* a $\mathbf{f}(\mathbf{x})$ (o $\mathbf{f}(\mathbf{x})$ está *dominado* por $\mathbf{f}(\hat{\mathbf{x}})$) y se denota como $\mathbf{f}(\hat{\mathbf{x}}) \prec \mathbf{f}(\mathbf{x})$.

Definición. $\hat{\mathbf{x}} \in X \subseteq \mathbb{R}^n$ es una solución eficiente (u óptima Pareto) si y solo si no existe ningún $\mathbf{x} \in X$ tal que $f_k(\mathbf{x}) \leq f_k(\hat{\mathbf{x}})$ $\forall k = 1, \ldots, p$, con $f_i(\mathbf{x}) < f_i(\hat{\mathbf{x}})$ para algún $i \in \{1, \ldots, p\}$, es decir si no existe ninguna otra solución de la región que la domine.
Se denota mediante X_E al conjunto de soluciones eficientes del problema y mediante $Y_N \in \mathbb{R}^p$ al conjunto de puntos no-dominados, imagen de X_E en el espacio de funciones $Y_N = \mathbf{f}(X_E)$.

Definición. $\hat{\mathbf{x}} \in X \subseteq \mathbb{R}^n$ es una solución débilmente eficiente (o débilmente óptima Pareto) si y solo si no existe ningún $\mathbf{x} \in X$ tal que $f_k(\mathbf{x}) < f_k(\hat{\mathbf{x}})$ $\forall k = 1, \ldots, p$.
Se denota por X_{wE} a la región de soluciones débilmente eficientes del problema y por $\mathbf{f}(X_{wE}) = Y_{wN} \in \mathbb{R}^p$ al conjunto de puntos débilmente no-dominados.

Dadas las definiciones de X_E y X_{wE} se deduce que $X_E \subseteq X_{wE}$.

Para continuar se va a mostrar en un ejemplo (extraído de [8]) cuáles son los conjuntos de soluciones eficientes y débilmente eficientes. En la figura 2.2 puede observarse en el espacio de soluciones el conjunto X_E en rojo (parte izquierda) y X_{wE} en azul (parte derecha).

$$min \begin{cases} -x_1 \\ -x_2 \end{cases}$$

$$s.a: \begin{cases} x_2 - \sqrt{4-(x_1-2)^2} - 2 \leq 0 \quad con \quad x_1 \in [2,4] \\ 0 \leq x_1, x_2 \leq 4 \end{cases}$$

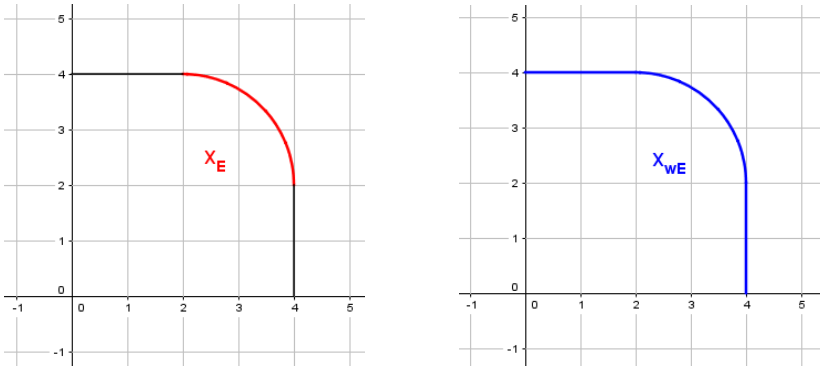

Figura 2.2: Soluciones eficientes y débilmente eficientes.

2.3. Métodos de resolución

Existen diferentes métodos para la obtención del conjunto de soluciones eficientes X_E (o débilmente eficientes X_{wE}). Un conjunto de métodos muy utilizados consiste en la escalarización de estos POMO's. Estos transforman el problema en uno con un único objetivo para así poder resolverlo con algún método uni-objetivo. A continuación se explican algunos de ellos y se ilustran con el ejemplo de la sección 2.1.

2.3.1. Método de la suma ponderada (o de los pesos)

Consiste en asignar a cada función objetivo $f_k(\mathbf{x})$, un peso $\lambda_k \geq 0$ (sin ser todos 0 simultáneamente), y minimizar su suma $\sum_{k=1}^{p} \lambda_k f_k(\mathbf{x})$ sujeto a las restricciones originales del problema: [2]

$$\begin{array}{ll} min & \sum\limits_{k=1}^{p} \lambda_k f_k(x_1,\ldots,x_n) \\ s.a: & g_i(x_1,\ldots,x_n) \leq 0, \quad i=1,\ldots,m \\ & (x_1,\ldots,x_n) \in \mathcal{R} \subseteq \mathbb{R}^n \end{array} \tag{2.3}$$

Con este método se pueden encontrar tanto soluciones débilmente eficientes como eficientes.

Teorema 2.1. *Sea $\hat{\boldsymbol{x}}$ una solución óptima de* (2.3) *entonces:*
· Si $\boldsymbol{\lambda} \geq \mathbf{0}$ entonces $\hat{\boldsymbol{x}}$ es una solución débilmente eficiente para (2.2).
· Si $\boldsymbol{\lambda} > \mathbf{0}$ entonces $\hat{\boldsymbol{x}}$ es una solución eficiente para (2.2).

Demostración. · Sea $\hat{\mathbf{x}}$ una solución óptima de (2.3), se supone que no es una solución débilmente eficiente para (2.2), luego $\exists \mathbf{x} \in X$ tal que $f_k(\mathbf{x}) < f_k(\hat{\mathbf{x}}) \ \ \forall k = 1,\ldots,p$. Cuando $\lambda_i \neq 0$ $\lambda_i f_i(\mathbf{x}) < \lambda_i f_i(\hat{\mathbf{x}})$, al no poder ser todos simultáneamente 0, $\sum_{k=1}^{p} \lambda_k f_k(\mathbf{x}) < \sum_{k=1}^{p} \lambda_k f_k(\hat{\mathbf{x}})$, llegando a contradicción, por tanto es débilmente eficiente para (2.2).
· Sea $\hat{\mathbf{x}}$ una solución óptima de (2.3), se supone que no es una solución eficiente para (2.2), luego $\exists \mathbf{x} \in X$ tal que $f_k(\mathbf{x}) \leq f_k(\hat{\mathbf{x}}) \ \ \forall k = 1,\ldots,p$ y $f_i(\mathbf{x}) < f_i(\hat{\mathbf{x}})$ para algún $i \in \{1,\ldots,p\}$, entonces $\sum_{k=1}^{p} \lambda_k f_k(\mathbf{x}) < \sum_{k=1}^{p} \lambda_k f_k(\hat{\mathbf{x}})$, llegando a contradicción, por tanto es eficiente para (2.2). □

Teorema 2.2. *Si $\hat{\boldsymbol{x}}$ es la única solución óptima de* (2.3) *entonces $\hat{\boldsymbol{x}}$ es eficiente para* (2.2).

Demostración. Sea $\hat{\mathbf{x}} \in X$ la única solución óptima de (2.3), se supone que no es una solución eficiente para (2.2), luego $\exists \mathbf{x} \in X$ tal que $f_k(\mathbf{x}) \leq f_k(\hat{\mathbf{x}}) \ \ \forall k = 1,\ldots,p$ y $f_i(\mathbf{x}) < f_i(\hat{\mathbf{x}})$ para algún $i \in \{1,\ldots,p\}$, y como $\boldsymbol{\lambda} \geq \mathbf{0}$ se tiene que $\sum_{k=1}^{p} \lambda_k f_k(\mathbf{x}) \leq \sum_{k=1}^{p} \lambda_k f_k(\hat{\mathbf{x}})$; por otro lado al ser $\hat{x}$ la única solución del problema de los pesos $\sum_{k=1}^{p} \lambda_k f_k(\hat{\mathbf{x}}) < \sum_{k=1}^{p} \lambda_k f_k(\mathbf{x}) \ \ \forall \mathbf{x} \in X$, llegando a contradicción, por tanto es eficiente para (2.2). □

La mayor debilidad del método de los pesos es que si no es el problema convexo no se va a poder encontrar todas las soluciones eficientes del problema multi-objetivo.

Teorema 2.3. *Sea el problema lineal multi-objetivo convexo. Si $\hat{\boldsymbol{x}}$ es una solución eficiente de* (2.2) *entonces existe un vector de pesos $\boldsymbol{\lambda} \geq \mathbf{0}$ tal que $\sum_{k=1}^{p} \lambda_k = 1$ con $\hat{\boldsymbol{x}}$ una solución de* (2.3).

Demostración. La demostración se apoya en los teoremas 2.6 y 2.7. □

Una de las características de este método es que el resultado obtenido depende de los pesos elegidos y esto debe de decidirse al principio del problema, tal y como se observa en los siguientes ejemplos.

$$\begin{array}{ll} min & 0.5(2x_1+x_2)+0.5(-x_1-x_2) \\ s.a: & \left\{ \begin{array}{l} x_1+2x_2-6 \leq 0 \\ -x_1+x_2 \leq 0 \\ x_1,x_2 \geq 0 \end{array} \right. \end{array} \qquad \begin{array}{ll} min & 0.25(2x_1+x_2)+0.75(-x_1-x_2) \\ s.a: & \left\{ \begin{array}{l} x_1+2x_2-6 \leq 0 \\ -x_1+x_2 \leq 0 \\ x_1,x_2 \geq 0 \end{array} \right. \end{array}$$

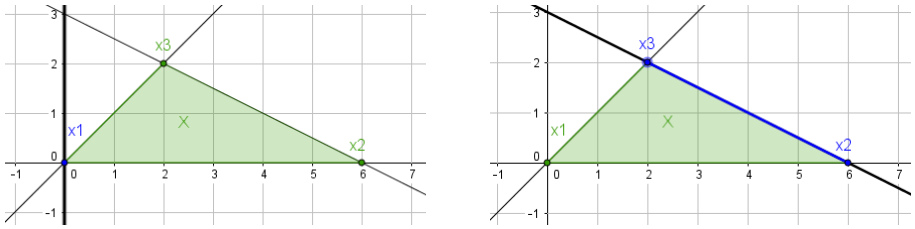

Figura 2.3: Ejemplos del método de la suma ponderada.

En caso de la izquierda se ha elegido $\lambda_1 = 0.5$ y $\lambda_2 = 0.5$, por ello se debe minimizar $0.5x_1$, obteniendo que la solución óptima está en (0,0). En el caso de la derecha se ha elegido $\lambda_1 = 0.25$ y $\lambda_2 = 0.75$, luego se debe minimizar $-0.25x_1 - 0.5x_2$, obteniendo que la solución óptima está en la recta $x_1 + 2x_2 = 6$ con $x_1 \in [2,6]$ y $x_2 \in [0,2]$. Ambas soluciones son eficientes del problema MO, por tanto, es claro que esa elección condiciona totalmente el resultado.

Más adelante, en el caso lineal que es en el que se profundiza en este trabajo, se demostrará que todas soluciones eficientes del problema MO pueden ser encontradas con este método, es decir que si $\hat{\mathbf{x}}$ es una solución eficiente de (2.2) entonces existirá un vector de pesos $\boldsymbol{\lambda} \geq \mathbf{0}$ tal que $\hat{\mathbf{x}}$ será una solución del problema (2.3). La condición suficiente para que esto ocurra es que el problema multi-objetivo sea convexo, como ya se ha adelantado en el teorema 2.3, por ello se cumple para el caso lineal.

2.3.2. Método ε-Restricción

Se basa en minimizar una única función $f_j(\mathbf{x})$ seleccionada previamente y añadir el resto de ellas a las restricciones tal que $f_k(\mathbf{x}) \leq \varepsilon_k \ \forall k \neq j$, así ε_k será el valor que se elige como el peor que puede tomar dicha función:

$$\begin{array}{lll} min & f_j(x_1,\ldots,x_n) & \\ s.a: & f_k(x_1,\ldots,x_n)) \leq \varepsilon_k & \forall k \neq j \\ & g_i(x_1,\ldots,x_n) \leq 0, & i = 1,\ldots,m \\ & (x_1,\ldots,x_n) \in \mathcal{R} \subseteq \mathbb{R}^n & \end{array} \tag{2.4}$$

Con este método también se pueden encontrar soluciones débilmente eficientes y eficientes.

Teorema 2.4. *Sea $\hat{\mathbf{x}}$ una solución óptima de* (2.4) *para algún j, entonces $\hat{\mathbf{x}}$ es débilmente eficiente para* (2.2).

Demostración. Sea $\hat{\mathbf{x}}$ una solución óptima de (2.4), se supone que no es una solución débilmente eficiente para (2.2), luego $\exists \mathbf{x} \in X$ tal que $f_k(\mathbf{x}) < f_k(\hat{\mathbf{x}}) \ \forall k = 1,\ldots,p$. Entonces, $f_k(\mathbf{x}) < f_k(\hat{\mathbf{x}}) \leq \varepsilon_k \ \forall k \neq j$, luego $\mathbf{x}$ es la solución óptima ya que $f_j(\mathbf{x}) < f_j(\hat{\mathbf{x}})$, llegando a contradicción, por tanto es débilmente eficiente para (2.2). □

Teorema 2.5. *Sea $\hat{\mathbf{x}}$ la única solución óptima de* (2.4) *para algún j, entonces $\hat{\mathbf{x}}$ es eficiente para* (2.2).

Demostración. Sea $\hat{\mathbf{x}}$ la única solución óptima de (2.4), se supone que no es una solución eficiente para (2.2), luego $\exists \mathbf{x} \in X$ tal que $f_k(\mathbf{x}) \leq f_k(\hat{\mathbf{x}}) \ \forall k = 1,\ldots,p$ y $f_i(\mathbf{x}) < f_i(\hat{\mathbf{x}})$ para algún $i \in \{1,\ldots,p\}$. Como podría ocurrir que $f_j(\mathbf{x}) = f_j(\hat{\mathbf{x}})$ y se tiene que $f_k(\mathbf{x}) \leq f_k(\hat{\mathbf{x}}) \leq \varepsilon_k \ \forall k \neq j$, entonces $\mathbf{x}$ también sería solución óptima, llegando a contradicción, por tanto es eficiente para (2.2). □

Teorema 2.6. *Sea $\hat{\mathbf{x}}$ una solución factible de* (2.2)*, entonces es eficiente para* (2.2) *si y solo si existe un $\hat{\boldsymbol{\varepsilon}} \in \mathbb{R}^p$ tal que $\hat{\mathbf{x}}$ es una solución óptima para* (2.4) *para todo $j = 1,\ldots,p$.*

[2] Sin pérdida de generalidad se elegirá un vector $\boldsymbol{\lambda} \geq \mathbf{0}$ que cumpla $\sum_{k=1}^{p} \lambda_k = 1$

Demostración. $\Rightarrow$) Sea $\hat{\boldsymbol{\varepsilon}} = \mathbf{f}(\hat{\mathbf{x}})$ y $\hat{\mathbf{x}}$ solución eficiente de (2.2). Suponiendo que no es una solución óptima de (2.4) para algún j, entonces debe de haber algún $\mathbf{x}$ factible tal que $f_j(\mathbf{x}) < f_j(\hat{\mathbf{x}})$ y $f_k(\mathbf{x}) \leq \hat{\varepsilon}_k = f_k(\hat{\mathbf{x}})$ para todo $k \neq j$, llegando a contradicción, luego si que es solución óptima de (2.4).
$\Leftarrow$) Sea $\hat{\boldsymbol{\varepsilon}} \in \mathbb{R}^p$ tal que $\hat{\mathbf{x}}$ es solución óptima de (2.4) y factible de (2.2). Suponiendo que no es solución eficiente de (2.2), entonces existe un índice $j \in \{1,\ldots,p\}$ y una solución factible $\mathbf{x}$ tal que $f_j(\mathbf{x}) < f_j(\hat{\mathbf{x}})$ y $f_k(\mathbf{x}) \leq f_k(\hat{\mathbf{x}})$ para $k \neq j$, luego $\hat{\mathbf{x}}$ no es solución óptima de (2.4) para $\hat{\boldsymbol{\varepsilon}}$, llegando a contradicción, por tanto si que es solución eficiente de (2.2). □

Además, se puede relacionar este método con el de los pesos gracias al siguiente teorema.

Teorema 2.7. *Sea un problema de optimización multi-objetivo convexo. Si $\hat{\boldsymbol{x}} \in X$ es una solución óptima de* (2.4) *para algún j y $\varepsilon_k = f_k(\hat{\boldsymbol{x}})$ $\forall k \neq j$, entonces existe un vector de pesos $\mathbf{0} \leq \boldsymbol{\lambda} \in \mathbb{R}^p$ tal que $\sum_{k=1}^{p} \lambda_k = 1$ con $\hat{\boldsymbol{x}}$ una solución óptima de* (2.3).

Demostración. Para realizar la demostración se necesitan conocimientos que no se pueden abarcar en este trabajo. (Ver Chankong and Haimes (1983, p.121)). □

Este teorema junto con el 2.6 sirven como base para la demostración del teorema 2.3 de las condiciones necesarias de eficiencia del método de las ponderaciones.

Al igual que en el método anterior, las decisiones tomadas al principio del método condicionan el resultado, además en este caso esto puede dar problemas de factibilidad como se observa en los siguientes ejemplos.

$$\begin{array}{ll} min & 2x_1 + x_2 \\ s.a: & \begin{cases} -x_1 - x_2 \leq -2 \\ x_1 + 2x_2 - 6 \leq 0 \\ -x_1 + x_2 \leq 0 \\ x_1, x_2 \geq 0 \end{cases} \end{array} \qquad \begin{array}{ll} min & -x_1 - x_2 \\ s.a: & \begin{cases} 2x_1 + x_2 \leq -1 \\ x_1 + 2x_2 - 6 \leq 0 \\ -x_1 + x_2 \leq 0 \\ x_1, x_2 \geq 0 \end{cases} \end{array}$$

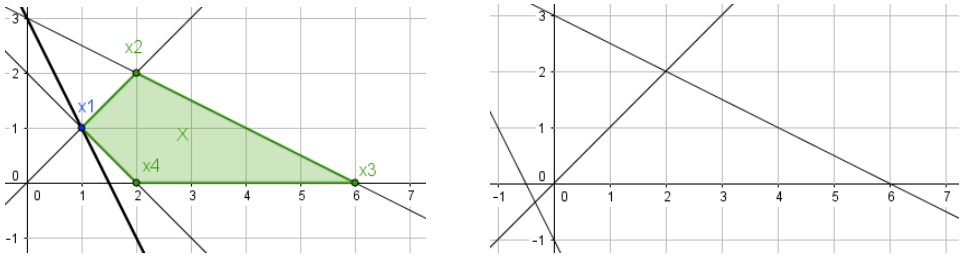

Figura 2.4: Ejemplos del método de ε-Restricción.

En el caso de la izquierda se desea minimizar $2x_1 + x_2$, añadiendo como restricción $-x_1 - x_2 \leq -2$ y así se obtiene que la solución óptima está en (1,1). En el caso se la derecha se intenta minimizar $-x_1 - x_2$, añadiendo como restricción $2x_1 + x_2 \leq -1$, pero se obtiene que es un problema no factible.

2.3.3. Otros métodos de resolución

Además de los métodos de resolución previamente descritos, también existen otros métodos como: el Método Híbrido, que combina el de los pesos y el ε-Restricción; el Método de la Restricción Elástica, que es una variante del ε-Restricción; el Método de Benson; o el Método de la Función Logro.[3].

Además, también se han desarrollado generalizaciones multi-objetivo de las condiciones necesarias y/o suficientes de Pareto optimalidad de Karush Kuhn y Tucker[6].

Capítulo 3

Optimización Lineal Multi-Objetivo

La optimización lineal multi-objetivo es un caso particular de la optimización multi-objetivo, en la que tanto las funciones objetivo como las restricciones son lineales. En lo que sigue se considerará el problema escrito en forma estándar:

$$\begin{array}{ll} min & C\mathbf{x} \\ s.a: & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geqq \mathbf{0} \end{array} \tag{3.1}$$

siendo $\mathbf{x} \in \mathbb{R}^n$ el vector de las variables de decisión, $C \in \mathbb{R}^{p \times n}$ la matriz de criterios (denotando sus filas mediante $C_k, k = 1, \ldots, p$), $A \in \mathbb{R}^{m \times n}$ la matriz de coeficientes de las restricciones (matriz de coeficientes tecnológicos) y $\mathbf{b} \in \mathbb{R}^m$ el vector de recursos.[1]
La región factible en el espacio de decisión es el poliedro

$$X = \{\mathbf{x} \in \mathbb{R}^n : A\mathbf{x} = \mathbf{b}, \mathbf{x} \geqq \mathbf{0}\}$$

y su imagen en el espacio de objetivos es

$$Y = CX = \{C\mathbf{x} \in \mathbb{R}^p : \mathbf{x} \in X\}$$

Tanto X como Y son espacios convexos y cerrados.

Sea

$$X_k = \{\hat{\mathbf{x}} \in X : C_k\hat{\mathbf{x}} \leq C_k\mathbf{x}\ \forall \mathbf{x} \in X\}$$

el espacio de soluciones óptimas de la función objetivo k-ésima, se asumirá que $\bigcap_{k=1}^{p} X_k = \emptyset$, es decir, que los objetivos del PPLMO tendrán conflictos unos con otros y por tanto no existirá una solución que los minimice a todos simultáneamente, tal y como ya se ha mencionado anteriormente.

3.1. Obtención de soluciones eficientes

Antes de explicar la forma de encontrar las soluciones eficientes se va a recordar el planteamiento del método de los pesos en un problema, pues es el que se va a emplear para resolver los problemas de formulación bi-objetivo de piensos comentados inicialmente, pero en esta ocasión se hará para el caso particular de un problema lineal.

$$\begin{array}{ll} min & \boldsymbol{\lambda}^T C\mathbf{x} \\ s.a: & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geqq \mathbf{0} \end{array} \tag{3.2}$$

[1] Cuando se trabaje con vectores se va a utilizar la siguiente notación:
$\mathbf{x} \geqq \mathbf{y} \Leftrightarrow x_i \geq y_i\ \forall i = 1, \ldots, n$
$\mathbf{x} \geq \mathbf{y} \Leftrightarrow x_i \geq y_i\ \forall i = 1, \ldots, n$ y $\mathbf{x} \neq \mathbf{y}$

siendo $\boldsymbol{\lambda} \geq \mathbf{0}$ el vector de los pesos tal que $\sum_{k=1}^{p} \lambda_k = 1$.

Como ya se explicó en el teorema 2.1 del capítulo anterior, se pueden encontrar soluciones eficientes mediante el método de los pesos. Para continuar se va a mostrar que no solo se pueden encontrar algunas soluciones eficientes con el método de los pesos sino todas ellas, es decir, con un vector de pesos apropiado se puede encontrar cualquier solución eficiente de (3.1) como solución óptima de (3.2); para ello se utilizará la teoría de dualidad.

Lema 3.1. *Una solución factible $\boldsymbol{x}^0 \in X$ del problema* (3.1) *es eficiente si y solo si el problema lineal*

$$\begin{array}{ll} max & \boldsymbol{e}^T \boldsymbol{z} \\ s.a: & A\boldsymbol{x} = \boldsymbol{b} \\ & C\boldsymbol{x} + \boldsymbol{z} = C\boldsymbol{x}^0 \\ & \boldsymbol{x}, \boldsymbol{z} \geqq \mathbf{0} \end{array} \tag{3.3}$$

con $\boldsymbol{e}^T = (1, \ldots, 1) \in \mathbb{R}^{1\times p}$, tiene una solución óptima $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{z}})$ con $\hat{\boldsymbol{z}} = \mathbf{0}$.

Demostración. Es claro que $\mathbf{x} = \mathbf{x}^0$, $\mathbf{z} = \mathbf{0}$ es solución factible de (3.3) de valor 0 en la función objetivo.
$\Leftarrow$) Sea $(\hat{\mathbf{x}}, \hat{\mathbf{z}})$ la solución óptima con $\hat{\mathbf{z}} = \mathbf{0}$ entonces $C\mathbf{x}^0 - C\hat{\mathbf{x}} = \mathbf{0}$, es decir $C\mathbf{x}^0 = C\hat{\mathbf{x}}$, y no existe $\mathbf{x} \in X$ tal que $C\mathbf{x} \leq C\mathbf{x}^0 = C\hat{\mathbf{x}}$ ya que $(\mathbf{x}, C\mathbf{x}^0 - C\mathbf{x})$ debería ser mejor solución que $(\hat{\mathbf{x}}, \hat{\mathbf{z}})$ y no lo es, por eso $\mathbf{x}^0$ es eficiente.
$\Rightarrow$) Sea $\mathbf{x}^0$ eficiente, entonces no existe $\mathbf{x} \in X$ tal que $C\mathbf{x} \leq C\mathbf{x}^0$, luego, como se tiene que cumplir $C\mathbf{x} + \mathbf{z} = C\mathbf{x}^0$, no existe $\mathbf{z}$ tal que $\mathbf{z} = C\mathbf{x}^0 - C\mathbf{x} \geq \mathbf{0}$, por eso *max* $\mathbf{e}^T\mathbf{z} \leq 0$ entonces *max* $\mathbf{e}^T\mathbf{z} = 0$. □

Lema 3.2. *Una solución factible $\boldsymbol{x}^0 \in X$ del problema* (3.1) *es eficiente si y solo si el problema lineal*

$$\begin{array}{ll} min & \boldsymbol{u}^T\boldsymbol{b} + \boldsymbol{w}^T C\boldsymbol{x}^0 \\ s.a: & \boldsymbol{u}^T A + \boldsymbol{w}^T C \geqq \mathbf{0} \\ & \boldsymbol{w} \geqq \boldsymbol{e} \\ & \boldsymbol{u} \in \mathbb{R}^m \end{array} \tag{3.4}$$

tiene una solución óptima $(\hat{\boldsymbol{u}}, \hat{\boldsymbol{w}})$ con $\hat{\boldsymbol{u}}^T\boldsymbol{b} + \hat{\boldsymbol{w}}^T C\boldsymbol{x}^0 = 0$.

Demostración. Como (3.4) es el dual de (3.3) entonces $(\hat{\mathbf{x}}, \hat{\mathbf{z}})$ es solución óptima del primal de costo 0 si y solo si el dual tiene en $(\hat{\mathbf{u}}, \hat{\mathbf{w}})$ una solución óptima con $\hat{\mathbf{u}}^T\mathbf{b} + \hat{\mathbf{w}}^T C\mathbf{x}^0 = 0$. □

Teorema 3.3. *Una solución factible $\boldsymbol{x}^0 \in X$ del problema* (3.1) *es eficiente si y solo si existe un $\boldsymbol{\lambda} \in \mathbb{R}^p > \mathbf{0}$ tal que*

$$\boldsymbol{\lambda}^T C\boldsymbol{x}^0 \leq \boldsymbol{\lambda}^T C\boldsymbol{x}$$

para todo $\boldsymbol{x} \in X$.

Demostración. $\Rightarrow$) Sea $\mathbf{x}^0 \in X_E$ en (3.1), entonces por el lema 3.2, hay una solución óptima $(\hat{\mathbf{u}}, \hat{\mathbf{w}})$ con $\hat{\mathbf{u}}^T\mathbf{b} + \hat{\mathbf{w}}^T C\mathbf{x}^0 = 0$ en (3.4).
Entonces, se toma $\hat{\mathbf{u}}$ como solución óptima de $min\{\mathbf{u}^T\mathbf{b} : \mathbf{u}^T A \geqq -\hat{\mathbf{w}}^T C,\ \mathbf{u} \in \mathbb{R}^m\}$ con un $\hat{\mathbf{w}}$ fijo. El dual de este es $max\{-\hat{\mathbf{w}}^T C\mathbf{x} : A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geqq \mathbf{0}\}$, el cual se sabe que tiene solución óptima. Con el teorema de la dualidad débil en un problema de mínimo se sabe que, $\mathbf{u}^T\mathbf{b} \geq -\hat{\mathbf{w}}^T C\mathbf{x}\ \ \forall \mathbf{u}, \mathbf{x}$ soluciones factibles de estos problemas primal y dual respectivamente. Y como $\hat{\mathbf{u}}^T\mathbf{b} + \hat{\mathbf{w}}^T C\mathbf{x}^0 = 0$, es decir, $\hat{\mathbf{u}}^T\mathbf{b} = -\hat{\mathbf{w}}^T C\mathbf{x}^0$ entonces $\mathbf{x}^0$ es solución óptima del dual, pues $\hat{\mathbf{u}}$ es la del primal.
Finalmente, al ser $max\{-\hat{\mathbf{w}}^T C\mathbf{x} : A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geqq \mathbf{0}\} \equiv min\{\hat{\mathbf{w}}^T C\mathbf{x} : A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geqq \mathbf{0}\}$, se debe tomar un $\hat{\mathbf{w}} \geq \mathbf{e} > \mathbf{0}$ para satisfacer la restricción de (3.4). Por lo tanto, se tomará $\hat{\mathbf{w}} = \boldsymbol{\lambda}$ en el problema (3.2).
$\Leftarrow$) El resultado se deduce del teorema 2.1. □

Con este último teorema se ha conseguido demostrar lo que se deseaba, es decir que con el problema (3.2) se pueden encontrar todas las soluciones eficientes de (3.1). Con todo ello, se llega a la conclusión que utilizando el método de la suma ponderada con un cierto vector de pesos, se puede resolver el problema lineal multi-objetivo con cualquier método uni-objetivo. Por esto, el problema se va reducir a ir generando los $\boldsymbol{\lambda}'s$ apropiados e ir resolviendo los PPL uni-objetivo con el método del Simplex.

3.2. Optimización Lineal Bi-Objetivo: Método Simplex

3.2.1. Problema Bi-Objetivo

Únicamente se va a considerar el caso bi-objetivo, pues es este caso el que interesa para el desarrollo del modelo correspondiente en el programa de diseño de dietas. Por lo tanto se va a tener que encontrar las soluciones eficientes de:

$$\begin{array}{ll} min & (C_1\mathbf{x}, C_2\mathbf{x})^T \\ s.a: & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geqq \mathbf{0} \end{array} \tag{3.5}$$

Por el teorema 3.3 se sabe que equivale a resolver:

$$min\{\lambda_1 C_1\mathbf{x} + \lambda_2 C_2\mathbf{x} : A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geqq \mathbf{0}\} \tag{3.6}$$

siendo $\boldsymbol{\lambda} \in \mathbb{R}^2_>$ tal que $\lambda_1 + \lambda_2 = 1$.

Es claro que $\lambda_2 = 1 - \lambda_1$, es decir se va a trabajar con un único parámetro[2], siendo $(\lambda_1, \lambda_2)^T = (\lambda, 1-\lambda)^T$ tal que $0 \leq \lambda \leq 1$.[3] Considerando:

$$C(\lambda) = \lambda C_1 + (1-\lambda)C_2$$

se llega finalmente a que el PPL que hay que resolver es:

$$min\{C(\lambda)\mathbf{x} : A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geqq \mathbf{0}\} \tag{3.7}$$

así, una vez determinado el parámetro λ, pasa a ser un problema de programación lineal común.

En el caso de dos únicos objetivos se puede resolver gráficamente. Utilizando un número finito de vectores peso es posible encontrar todos los puntos no dominados, y con ello las soluciones eficientes, ya que X e Y son poliedros.

3.2.2. Método Simplex en PPL Bi-Objetivo

Antes de explicar el algoritmo se van a recordar las características en las que se apoya el método Simplex en programación lineal, siendo $\mathcal{B}$ y $\mathcal{N}$ las bases asociadas a las variables básicas (B) y no básicas (N) respectivamente, y $\bar{C}(\lambda) = \lambda\bar{C}_1 + (1-\lambda)\bar{C}_2$ el vector de costos marginales[4] del PPL (3.7):

- Si el PPL (3.7) es factible, entonces existe al menos una solución factible básica (s.f.b.).
- Si la función objetivo $C(\lambda)\mathbf{x}$ está acotada inferiormente en X, entonces existe una s.f.b. óptima.
- Si $\bar{C}_\mathcal{N} \geqq \mathbf{0}$, entonces la s.f.b. $(\mathbf{x}_\mathcal{B}, \mathbf{0})$ es óptima.

[2] A partir de este punto se hablará de λ como un escalar

[3] Se asume que el problema tiene solución única al considerar λ=0 y λ=1, y que por el teorema 2.2 dichas soluciones serán eficientes

[4] $\bar{C}_\mathcal{N} = C_\mathcal{N} - C_\mathcal{B}B^{-1}N$ y $\bar{C}_\mathcal{B} = C_\mathcal{B} - C_\mathcal{B}B^{-1}B = \mathbf{0}$

Además, también hay que recordar que si el PPL (3.7) es no degenerado y $\mathcal{B}$ es una base óptima, entonces $\bar{C}_{\mathcal{N}} \geqq \mathbf{0}$.

Por lo tanto, para comenzar se necesita saber si el problema (3.7) es factible y hallar una solución factible básica de este. Para ello se encuentra una solución del problema auxiliar:

$$\begin{array}{ll} min & \mathbf{e}^T\mathbf{z} \\ s.a: & A\mathbf{x}+\mathbf{z}=\mathbf{b} \\ & \mathbf{x},\mathbf{z} \geqq \mathbf{0} \end{array} \tag{3.8}$$

el cual es siempre factible y $(\mathbf{x},\mathbf{z}) = (\mathbf{0},\mathbf{b})$ es una s.f.b de este.

Lema 3.4. *El problema* (3.7) *es factible, es decir $X \neq \emptyset$, si y solo si el problema auxiliar* (3.8) *tiene una solución óptima en $(\hat{\boldsymbol{x}},\hat{\boldsymbol{z}})$ con $\hat{\boldsymbol{z}} = \mathbf{0}$.*

Una vez que se sabe que el problema (3.7) es factible y se tiene una s.f.b de este, se supone que la base asociada a las variables básicas es una base óptima para algún $\lambda = \hat{\lambda}$ y se denota por $\hat{\mathcal{B}}$, entonces $\bar{C}(\hat{\lambda}) = \hat{\lambda}\bar{C}_1 + (1-\hat{\lambda})\bar{C}_2 \geqq \mathbf{0}$. Se necesitan distinguir los dos casos que se pueden encontrar:

1. Si $\bar{C}_2 \geq \mathbf{0}$:
   Como $\bar{C}(\hat{\lambda}) = \hat{\lambda}\bar{C}_1 + (1-\hat{\lambda})\bar{C}_2 \geqq \mathbf{0}$, entonces $\bar{C}(\lambda)$ continua siendo no negativa siempre que $\lambda \leq \hat{\lambda}$ ya que el coeficiente de $\bar{C}_2$ será mayor y el de $\bar{C}_1$ menor y así se seguirá manteniendo que es $\geqq \mathbf{0}$. Luego $\hat{\mathcal{B}}$ es base óptima $\forall \lambda \in [0,\hat{\lambda}]$.

2. Si hay al menos un $i \in \mathcal{N}$ con $\bar{C}_{2,i} < 0$:
   Existe un valor $\lambda < \hat{\lambda}$ tal que $\bar{C}(\lambda)_i = 0$, es decir $\bar{C}(\lambda)_i = \lambda\bar{C}_{1,i} + (1-\lambda)\bar{C}_{2,i} = 0$, luego

   $$\lambda(\bar{C}_{1,i} - \bar{C}_{2,i}) + \bar{C}_{2,i} = 0 \quad \Rightarrow \quad \lambda = \frac{-\bar{C}_{2,i}}{\bar{C}_{1,i} - \bar{C}_{2,i}}$$

   por debajo de ese valor $\hat{\mathcal{B}}$ ya no es óptima.
   Por lo tanto se define $I = \{i \in \mathcal{N} : \bar{C}_{2,i} < 0, \bar{C}_{1,i} \geq 0\}$ y

   $$\lambda' := max_{i \in I} \frac{-\bar{C}_{2,i}}{\bar{C}_{1,i} - \bar{C}_{2,i}}$$

   luego $\hat{\mathcal{B}}$ es base óptima $\forall \lambda \in [\lambda', \hat{\lambda}]$. En el momento que $\lambda \leq \lambda'$ una nueva base se convierte en óptima. Entonces, la variable x_s entra en la base siendo $s = i \in I$ que da valor a λ', la variable que sale es x_r tal que $r \in \hat{\mathcal{B}}$ dando valor a $min_{r \in \hat{\mathcal{B}}}\{\frac{\tilde{b}_r}{\tilde{A}_{sr}} : \tilde{A}_{sr} > 0\}$ y se toma $\hat{\lambda} = \lambda'$ en la siguiente iteración.

Para comenzar siempre se elegirá $\hat{\lambda} = 1$ y se irá iterando (encontrando nuevas variables que entran en la base y otras que la dejan) hasta que $\bar{C}_2 \geqq \mathbf{0}$, es decir, hasta que $I = \emptyset$. Hay que notar que inicialmente $I \neq \emptyset$, ya que, asumiendo que los objetivos están enfrentados, si $\mathcal{B}$ es una base óptima para $\lambda = 1$ entonces no será optima para $\lambda = 0$.

Una vez acabado este procedimiento se obtienen una secuencia de valores λ $(1 = \lambda^1 > \ldots > \lambda^l = 0)$ y de bases óptimas $(\mathcal{B}^1, \ldots, \mathcal{B}^{l-1})$ que definen las s.f.b. óptimas del problema (3.7): $\mathcal{B}^i$ es una base óptima con $\lambda \in [\lambda^{i+1}, \lambda^i]$ $\forall i = 1, \ldots, l-1$.

En el algoritmo 1 se muestra el pseudocódigo correspondiente, en el que se recogen todas las características anteriores. En la fase 1 se comprueba si el problema (3.7) es factible usando el lema 3.4; entonces en la fase 2 se resuelve el problema (3.7) eligiendo $\lambda = 1$ y considerando como base óptima la que se ha obtenido de la solución óptima en la fase 1; tras ello, en la fase 3 se realiza el proceso explicado anteriormente, en el que mientas $I = \{i \in \mathcal{N} : \bar{C}_{2,i} < 0, \bar{C}_{1,i} \geq 0\} \neq \emptyset$ se va iterando hallando

λ y averiguando la variable que entra y sale de la base.

Algorithm 1: Algoritmo del PPL Simplex bi-objetivo

Input: A,b,C para un PL bi-objetivo.

Fase 1: Resolver el problema auxiliar (3.8) usando el algoritmo Simplex uni-objetivo. Si el valor es positivo, PARAR, $X = \emptyset$. Si no $\mathcal{B}$ es una base óptima.

Fase 2: Resolver el PPL (3.7) para $\lambda = 1$ considerando como base óptima la base $\mathcal{B}$ hallada en la Fase 1. Calcular $\tilde{A}$ y $\tilde{b}$.

Fase 3:

while $I = \{i \in \mathcal{N} : \bar{C}_{2,i} < 0, \bar{C}_{1,i} \geq 0\} \neq \emptyset$ **do**

$\lambda := max_{i \in I} \dfrac{-\bar{C}_{2,i}}{\bar{C}_{1,i} - \bar{C}_{2,i}}$

$s \in argmax\{i \in I : \dfrac{-\bar{C}_{2,i}}{\bar{C}_{1,i} - \bar{C}_{2,i}}\}$

$r \in argmin\{j \in \mathcal{B} : \dfrac{\tilde{b}_j}{\tilde{A}_{sj}}, \tilde{A}_{sj} > 0\}$

$\mathcal{B} := (\mathcal{B} \backslash \{r\}) \cup \{s\}$ y actualizar $\tilde{A}$ y $\tilde{b}$.

Output: Secuencia de $\lambda - valores$ y de soluciones factibles básicas óptimas.

Tras presentar el algoritmo se muestra su funcionamiento en el siguiente problema, extraído de (Steuer (1985)) [3].

$$min \begin{cases} 3x_1 + x_2 \\ -x_1 - 2x_2 \end{cases} \qquad s.a: \begin{cases} x_2 \leq 3 \\ 3x_1 - x_2 \leq 6 \\ x_1, x_2 \geq 0 \end{cases}$$

$$\Rightarrow$$

$$min \;\; (4\lambda - 1)x_1 + (3\lambda - 2)x_2$$

$$s.a: \begin{cases} x_2 + x_3 = 3 \\ 3x_1 - x_2 + x_4 = 6 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

donde:

$$C = \begin{pmatrix} 3 & 1 \\ -1 & -2 \end{pmatrix} \qquad A = \begin{pmatrix} 0 & 1 \\ 3 & -1 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} 3 \\ 6 \end{pmatrix} \qquad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

En la figura 3.1 se puede observar el espacio de soluciones (decisiones), así como su imagen en el espacio de funciones. Se ve cómo el conjunto X_E está definido por los segmentos que unen $(0,0)$ con $(0,3)$ y $(0,3)$ con $(3,3)$, así como su imagen.

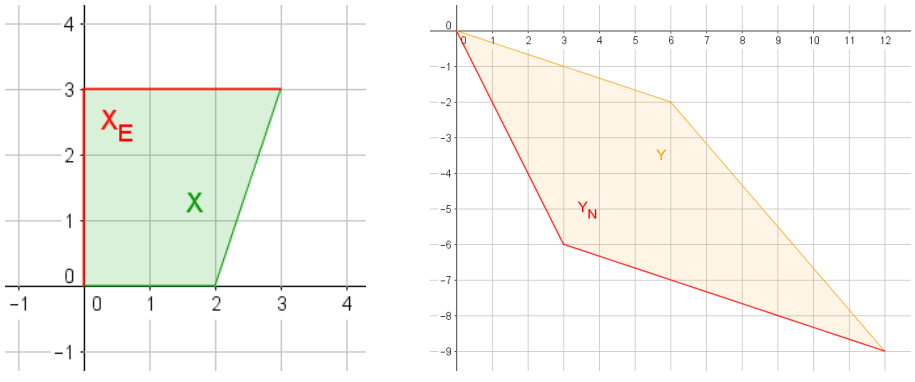

Figura 3.1: región factible con soluciones eficientes e imagen con puntos no dominados.

Para su resolución con el método Simplex se transforma el problema en uno como el (3.7):

$$C = \begin{pmatrix} 3 & 1 & 0 & 0 \\ -1 & -2 & 0 & 0 \end{pmatrix} \quad A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 3 & -1 & 0 & 1 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 3 \\ 6 \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

$$C(\lambda) = \lambda C_1 + (1-\lambda)C_2 = \lambda(3,\ 1,\ 0,\ 0) + (1-\lambda)(-1,\ -2,\ 0,\ 0) =$$
$$= (3\lambda,\ \lambda,\ 0,\ 0) + (-1,\ -2,\ 0,\ 0) + (\lambda,\ 2\lambda,\ 0,\ 0) = (4\lambda - 1,\ 3\lambda - 2,\ 0,\ 0)$$

Una vez que ya se tiene el problema preparado se comienza con el algoritmo Simplex. Como este ejemplo se va a resolver manualmente se utilizará la tabla del Simplex con los costos marginales $\bar{C}_1$ y $\bar{C}_2$.

- Iteración 1: $\lambda = 1,\ \mathcal{B} = \{3,4\}$.

Min	x_1	x_2	x_3	x_4	$B^{-1}\mathbf{b}$
x_3	0	$1^{\clubsuit}$	1	0	3
x_4	3	-1	0	1	6
$\bar{C}_1$	3	1	0	0	0
$\bar{C}_2$	-1	-2	0	0	0

  s.f.b: $\mathbf{x} = (0,\ 0,\ 3,\ 6)^T,\ \bar{C}(\lambda) = (3,\ 1,\ 0,\ 0)$
  $I = \{1,2\},\ \lambda' = max_{i\in I}\{\frac{1}{3+1}, \frac{2}{1+2}\} = \frac{2}{3},$ entra: $s = 2,$ sale: $r = 3\ \left(min_{j\in\mathcal{B}}\{\frac{3}{1}\} = 3\right).$

- Iteración 2: $\lambda = \frac{2}{3},\ \mathcal{B} = \{2,4\}$.

Min	x_1	x_2	x_3	x_4	$B^{-1}\mathbf{b}$
x_2	0	1	1	0	3
x_4	$3^{\clubsuit}$	0	1	1	9
$\bar{C}_1$	3	0	-1	0	-3
$\bar{C}_2$	-1	0	2	0	6

  s.f.b: $\mathbf{x} = (0,\ 3,\ 0,\ 9)^T,\ \bar{C}(\lambda) = (\frac{5}{3},\ 0,\ 0,\ 0)$
  $I = \{1\},\ \lambda' = max_{i\in I}\{\frac{1}{3+1}\} = \frac{1}{4},$ entra: $s = 1,$ sale: $r = 4\ \left(min_{j\in\mathcal{B}}\{\frac{9}{3}\} = 3\right).$

- Iteración 3: $\lambda = \frac{1}{4},\ \mathcal{B} = \{1,2\}$.

Min	x_1	x_2	x_3	x_4	$B^{-1}\mathbf{b}$
x_2	0	1	1	0	3
x_1	1	0	$\frac{1}{3}$	$\frac{1}{3}$	3
$\bar{C}_1$	0	0	-2	-1	-12
$\bar{C}_2$	0	0	$\frac{7}{3}$	$\frac{1}{3}$	9

  s.f.b: $\mathbf{x} = (3,\ 3,\ 0,\ 0)^T,\ \bar{C}(\lambda) = (0,\ 0,\ \frac{5}{4},\ 0)$
  $I = \emptyset\ \checkmark.$

Se han obtenido los siguientes resultados:

- La base $\mathcal{B} = \{3,4\}$ y la s.f.b $\mathbf{x} = (0,\ 0,\ 3,\ 6)^T$ son óptimas para $\lambda \in [\frac{2}{3}, 1]$, con valor del vector objetivo $C\mathbf{x} = (0,\ 0)^T$.

- La base $\mathcal{B} = \{2,4\}$ y la s.f.b $\mathbf{x} = (0,\ 3,\ 0,\ 9)^T$ son óptimas para $\lambda \in [\frac{1}{4}, \frac{2}{3}]$, con valor del vector objetivo $C\mathbf{x} = (3,\ -6)^T$.
- La base $\mathcal{B} = \{1,2\}$ y la s.f.b $\mathbf{x} = (3,\ 3,\ 0,\ 0)^T$ son óptimas para $\lambda \in [0, \frac{1}{4}]$, con valor del vector objetivo $C\mathbf{x} = (12,\ -9)^T$.

Los valores $\lambda = \frac{2}{3}$ y $\lambda = \frac{1}{4}$ corresponden a los vectores de pesos $(\frac{2}{3},\ \frac{1}{3})^T$ y $(\frac{1}{4},\ \frac{3}{4})^T$ respectivamente. Entonces de deben de minimizar las funciones:

$$\frac{2}{3}(3x_1 + x_2) + \frac{1}{3}(-x_1 - 2x_2) = \frac{5}{3}x_1$$

$$\frac{1}{4}(3x_1 + x_2) + \frac{3}{4}(-x_1 - 2x_2) = -\frac{5}{4}x_2$$

es por esto que la región de las soluciones eficientes es la que aparece en la figura 3.1, la cual ha sido encontrada con solo dos vectores de pesos.

Al igual que el algoritmo Simplex en PL uni-objetivo no encuentra todas las soluciones óptimas, en bi-objetivo no encuentra todas las soluciones eficientes. Unicamente se van a encontrar unas s.f.b. eficientes que se corresponden a todos los puntos extremos de la región de soluciones eficientes, por ello la X_E es el borde de la región factible uniendo estos puntos extremos e Y_N será la imagen de esta región.

Capítulo 4

Desarrollo del programa

Este capítulo se centra en el programa. Como ya se ha mencionado en la introducción, este trabajo es una continuación y ampliación del TFG ”Desarrollo de un Programa Informático para la Formulación de Piensos Compuestos” [1]. Por lo tanto, el programa desarrollado en el presente trabajo, se inicia a partir del código que se desarrolló en el anterior trabajo, el cual formulaba un compuesto atendiendo únicamente a un objetivo. A esta primera versión del software se le ha incorporado un menú para realizar la formulación bi-objetivo.

4.1. Antigua versión del software

Al arrancar el programa se veía el panel en el que se generaba la dieta. En este panel se podían seleccionar manualmente los nutrientes necesarios del compuesto y los ingredientes que se iban a emplear para él, ambos se podían acotar superiormente e inferiormente. Además, a cada ingrediente le aparecía su precio, el cual también estaba permitido que fuera modificado manualmente. Todo ello se podía guardar en un fichero *.die* para posteriormente cargarlo directamente sin la necesidad que volver a seleccionar todo manualmente.

Una vez que todos los requerimientos de la dieta estaban seleccionados era el momento de generar el problema atendiendo al único objetivo de minimizar el coste del compuesto. Este problema se podía escribir en un segundo panel en forma matemática. El siguiente paso era resolver el problema, la solución de este (salvo cuando era no factible) aparecía en este mismo panel. Por último, se le podía realizar un análisis de sensibilidad al problema y generar un Excel con todos los resultados.

4.2. Nueva versión del software

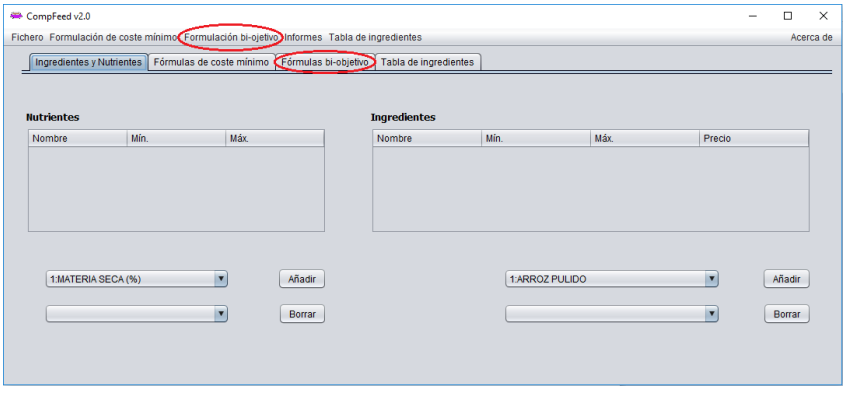

Figura 4.1: Inicio del programa

En la nueva versión, continúan apareciendo todas opciones que ofrecía la primera versión. Sin embargo, ahora también se puede ver un menú de formulación bi-objetivo y un panel donde aparecerán las soluciones de esta nueva formulación, tal y como se puede ver en la figura 4.1. Para generar la dieta se trabajará de la misma forma que en la antigua versión.

El menú *Formulación bi-objetivo* contiene dos opciones: *Generar problema* y *Optimizar*. La primera opción pasa a estar disponible cuando al menos una de las tablas de nutrientes o de ingredientes contiene algún elemento. La segunda opción se activa una vez que el problema está generado.

4.2.1. Generar problema

Al seleccionar esta opción, aparece la ventana emergente de la figura 4.2, la cual pide que se seleccione el nutriente que se va a querer maximizar en el compuesto, este será el segundo criterio del problema[1]. El primer criterio de problema será minimizar el coste.

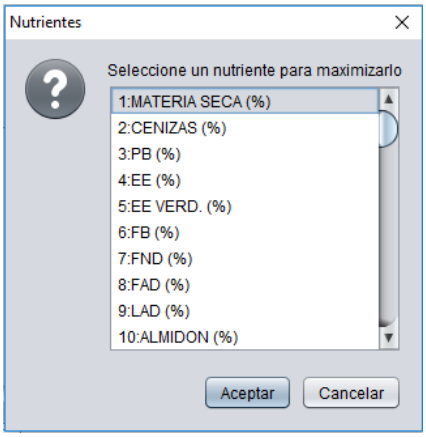

Figura 4.2: Selección del segundo criterio.

Una vez seleccionado el nutriente, aparece una nueva venta la cual pregunta si se desea escribir el problema en el panel de *fórmulas bi-objetivo*, de ser así aparecerá escrito en forma matemática. Un ejemplo de ello es lo que muestra la figura 4.3, en él, se ha formulado el sencillo problema que se formuló al principio del capítulo 2, pero añadiéndole como segundo objetivo maximizar la energía.

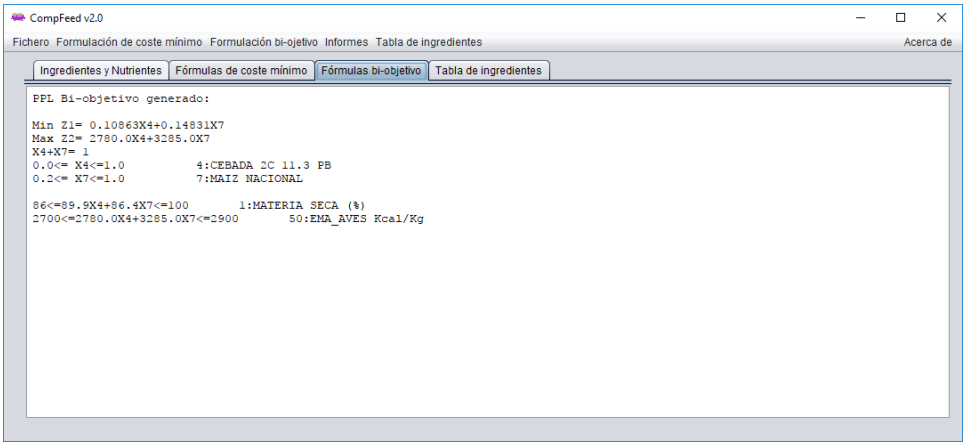

Figura 4.3: Ejemplo formulacón bi-objetivo.

[1]Como se trabajará con mínimo, se toma la función opuesta.

4.2.2. Optimizar

Esta segunda opción resuelve el problema bi-objetivo. Como ya se ha comentado, generalmente, los dos objetivos van ser de carácter conflictivo, por lo que mejorar uno afectará de forma negativa al otro. Por tanto, el programa escribirá todas las soluciones eficientes correspondientes a los puntos extremos de la región de soluciones eficientes, cada solución se consigue con un λ perteneciente a un subintervalo de $[0, 1]$. De cada solución se muestra el valor de cada uno de los dos objetivos (coste y nutriente seleccionado), la cantidad que se utiliza de cada ingrediente y la cantidad de cada nutriente que posee la dieta. En la figura 4.4 se pueden ver las soluciones del problema generado anteriormente.

```
Con lambda perteneciente a: [0.9999214319160031,1.0]

Solución de costo:    0,1166 € por Kg

Solución con:   2881,0000  del nutriente  50:EMA_AVES Kcal/Kg


Ingredientes utilizados:
4:CEBADA 2C 11.3 PB:                        80,000 %
7:MAIZ NACIONAL:                            20,000 %

Ingredientes no utilizados:

Composición lograda:
1:MATERIA SECA (%):       89,200  en (    86,000,   100,000)
50:EMA_AVES Kcal/Kg:    2881,000  en (  2700,000,  2900,000)
```

```
Con lambda perteneciente a: [0.0,0.9999214319160031]

Solución de costo:    0,1181 € por Kg

Solución con:   2900,0000  del nutriente  50:EMA_AVES Kcal/Kg


Ingredientes utilizados:
4:CEBADA 2C 11.3 PB:                        76,238 %
7:MAIZ NACIONAL:                            23,762 %

Ingredientes no utilizados:

Composición lograda:
1:MATERIA SECA (%):       89,068  en (    86,000,   100,000)
50:EMA_AVES Kcal/Kg:    2900,000  en (  2700,000,  2900,000)
```

Figura 4.4: Soluciones del ejemplo.

Además el programa crea una tabla para poder comparar de forma más sencilla todas estas soluciones, esta tabla contiene una fila con el precio de cada solución, otra con la cantidad del nutriente seleccionado y una fila con la cantidad de cada uno de los ingredientes utilizados. Por último, aparece el gráfico *Coste/Nutriente*, que ayuda a ver el incremento de cada uno de los objetivos en cada solución, para así poder elegir la que más conviene, además hay que notar que los puntos de unión entre soluciones también son soluciones eficientes. La figura 4.5 muestra la tabla y el gráfico del problema anterior.

```
TABLA RESUMEN

Coste                                0,117    0,118
Nutriente: 50:EMA_AVES Kcal/Kg    2881,000 2900,000

4:CEBADA 2C 11.3 PB                 80,000   76,238
7:MAIZ NACIONAL                     20,000   23,762
```

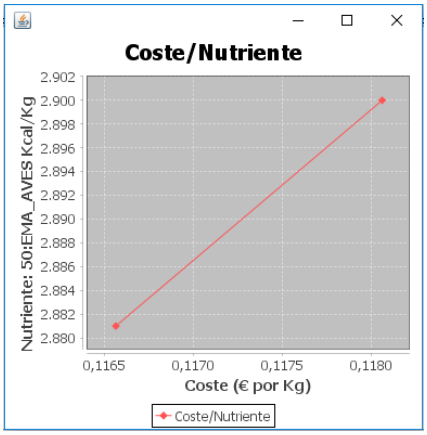

Figura 4.5: Tabla y gráfico del ejemplo.

4.3. La programación

Para la construcción del algoritmo bi-objetivo se ha utilizado la librería GLPK [5]. Con la ayuda de esta librería se ha creado una nueva clase llamada `PPLsBiObjetivo`, en ella está la función `resuelveProblema`, que es la que halla las soluciones, las escribe, construye la tabla resumen y dibuja el gráfico. El código de esta clase puede consultarse en el anexo B.

4.3.1. El constructor

Lo primero que hace el programa es crear un PPLBO, para ello el constructor de la clase `PPLsBiObjetivo` toma como argumentos la tabla de nutrientes, la tabla de ingredientes y el área de texto donde se imprimen todos los resultados. En el cuerpo del constructor se crea el problema de la siguiente manera:

- Se crea el problema con la rutina `GLPK.glp_create_prob`, el cual inicialmente está vacío.
- Se establecen las variables del problema. Para ello se utiliza la rutina `GLPK.glp_add_cols` que fija el numero de variables estructurales o columnas, en este caso es el número de ingredientes. Con `GLPK.glp_set_col_kind` se implanta el tipo de variables, en este caso se elige que son de tipo continuo. Y con `GLPK.glp_set_col_bnds` se establece que las variables están acotadas superior e inferiormente.
- Se establecen las restricciones del problema. Para ello se utiliza la rutina `GLPK.glp_add_rows` que fija el número de variables auxiliares o filas, en este caso es el número de nutrientes más uno, ya que la primera restricción del problema es que la suma de las cantidades de todos los ingredientes sea 1. Esta primera restricción es de tipo fijo, el resto están acotadas superior e inferiormente, todo ello se establece a través de `GLPK.glp_set_row_bnds`. Para asignar el valor de los coeficientes de las restricciones se utilizan dos arrays de la clase GLPK (uno de números enteros y otro de reales), estos se crean mediante `GLPK.new_intArray` y `GLPK.new_doubleArray`; una vez creados, se utilizan las rutinas `GLPK.intArray_setitem` y `GLPK.doubleArray_setitem` para asignar dichos coeficientes a los nuevos arrays, en la primera restricción se asignan todo unos y en las siguientes la cantidad de ese nutriente que posee cada ingrediente; el último paso es almacenar las restricciones guardadas en estos arrays en una matriz con `GLPK.glp_set_mat_row`.
- Se establece la función objetivo. Aunque en el caso bi-objetivo se tengan dos funciones, para empezar se toma como función objetivo únicamente la primera de ellas, ya que se comienza con $\lambda = 1$. Con la rutina `GLPK.glp_set_obj_dir` se establece que el problema es de mínimo y con `GLPK.glp_set_obj_coef` se almacenan los coeficientes en las variables estructurales, que son el coste que tiene cada uno de los ingredientes.

4.3.2. Los métodos

La clase GLPK no está preparada para resolver problemas multi-objetivo, unicamente obtiene una solución óptima en problemas uni-objetivo. Sin embargo, como se ha explicado en el capítulo anterior, con el método de las ponderaciones, el problema bi-objetivo se puede reducir a ir generando los λ's e ir resolviendo los PPL uni-objetivo. Por ello, se van a crear unos métodos en la clase `PPLsBiObjetivo`, con estos se podrá hallar los λ's manualmente.

Los sucesivos λ's se hallan con la fórmula:

$$\lambda := max_{i \in I} \frac{-\bar{C}_{2,i}}{\bar{C}_{1,i} - \bar{C}_{2,i}}$$

por lo tanto, para poder aplicar esta fórmula se va a tener que hallar $\bar{C}_1$ y $\bar{C}_2$ en cada iteración, esto se va a realizar con los métodos `costosMarginales1` y `costosMarginales2`, los cuales van a ser idénticos, salvo por el array con el que trabaja cada uno. Una vez que se tienen los costes marginales, se utiliza el método `calculaLambda`, el cual trabaja con la anterior fórmula. Y finalmente se utiliza el método de las ponderaciones con el lambda hallado, para así obtener la función objetivo que se le pasará al problema en cada iteración, este último paso se realiza en el método `modificaProblema`.

costesMarginales

Los métodos de los costes marginales utilizan las mismas rutinas que se han empleado para asignar los coeficientes de las restricciones a los arrays de la clase GLPK, en este caso asignan los coeficientes de C_1 o C_2 no nulos, dependiendo de que coste marginal se este hallando. Después, se utiliza la rutina `GLPK.glp_transform_row`, la cual realiza la operación $\bar{C}_{1,j} = C_{1,j} - C_{1,\mathcal{B}}B^{-1}A_j \ \forall j$ en la primera función, y de forma similar para los costes marginales de la segunda función objetivo. El último paso se realiza con las rutinas `GLPK.intArray_getitem` y `GLPK.doubleArray_getitem`, con las cuales se almacenan los nuevos costes marginales en arrays, que seran los que devuelven estos métodos.

calculaLambda

El método que calcula lambda comienza hallando las variables que están en I:

$$I = \{i \in \mathcal{N} : \bar{C}_{2,i} < 0, \bar{C}_{1,i} \geq 0\}$$

sin embargo, debido a que la librería GLPK transforma el problema en uno con cotas superiores e inferiores, se debe de considerar también el problema equivalente en la cota superior pues esta definición sólo afecta a las variables no básicas que están en la cota inferior, para las variables no básicas que están en la cota superior:

$$I = \{i \in \mathcal{N} : \bar{C}_{2,i} > 0, \bar{C}_{1,i} \leq 0\}$$

Utilizando las rutinas `GLPK.glp_get_row_stat` y `GLPK.glp_get_col_stat` sabemos si las variables auxiliares y las estructurales respectivamente, están en cota inferior o superior, para así emplear una u otra definición. Una vez que se sabe cuál es el conjunto I se emplea la fórmula para averiguar el valor de λ, este valor es el que devuelve el método.

modificaProblema

El método que calcula la función objetivo toma como argumento el valor de λ y emplea la fórmula:

$$C(\lambda) = \lambda C_1 + (1-\lambda)C_2$$

una vez que se ha hallado, con la ayuda de la rutina `GLPK.glp_set_obj_coef` se almacenan los coeficientes de la función objetivo.

resuelveProblema

En el momento que se tiene la función objetivo, se puede utilizar el método `resuelveProblema` para resolver un problema uni-objetivo. Este método comienza tomando $\lambda = 1$, pues como ya se ha mencionado, siempre se comienza tomando este valor.

Entonces, el método entra en un bucle hasta que $I = \emptyset$. En este bucle con rutina `GLPK.glp_simplex` se recuperan los datos del problema, se resuelve por el método simplex y se almacenan los resultados. Con la rutina `GLPK.glp_get_status` se comprueba si la solución es óptima:

- Si no es óptima se considera que el problema es no factible y se muestra un mensaje de que el problema es no factible (por construcción de este no puede ser no acotado).
- Si es óptima se imprime la solución del problema en el panel *Fórmulas bi-objetivo*, se calculan los nuevos costes marginales, se halla el lambda de la siguiente iteración y la nueva función objetivo.

Todo esto lo realiza hasta que se sale del bucle, que es tantas veces como soluciones eficientes correspondientes a los puntos extremos de la región de soluciones eficientes hay.

Una vez halladas todas las soluciones se escribe la tabla comparativa de todas las soluciones y se dibuja el gráfico *Coste/Nutriente*.

escribeSolucion

Para imprimir cada una de las soluciones el programa utiliza el método `escribeSolucion`. Para cada intervalo de lambdas (desde el lambda hallado hasta el de la siguiente solución) se escribe el precio del compuesto (primera función objetivo en la solución óptima) y la cantidad de nutriente seleccionado para maximizar (segunda función objetivo en la solución óptima). A continuación, se escribe la cantidad de cada ingrediente utilizado y después la de los no utilizados. Por último, se escribe la composición del compuesto logrado, es decir, la cantidad de cada uno de los nutrientes junto con sus cotas inferior y superior.

escribeTabla

El método `escribeTabla` escribe en el panel *Fórmulas bi-objetivo* las componentes de una matriz, la cual se ha ido creando según se iban hallando las soluciones. Esta tabla está formada según se ha explicado en la subsección 4.2.2.

dibujaGrafico

El método `dibujaGrafico`, es el encargado de crear la pantalla emergente del gráfico *Coste/Nutriente*. Para la creación de este método se ha utilizado la librería *jFreeChart* [4].

Capítulo 5

Ejemplo real y conclusiones

En este capítulo se muestra un ejemplo real de formulación de piensos resuelto con el módulo adicional creado en el programa informático con el que se ha trabajado. Posteriormente se presentan las conclusiones obtenidas con la realización de este trabajo.

5.1. Ejemplo real de formulación bi-objetivo de piensos compuestos

Para realizar una adecuada formulación de un pienso, se busca una combinación de ingredientes que pueda cubrir los requerimientos de los animales, es decir que proporcione un aporte de nutrientes el cual permita un rendimiento productivo máximo, sin que esto afecte al bienestar o a la salud de los animales.

Se considera como ejemplo la dieta del fichero `gallinas_jun17b.die`, facilitado como ya se ha dicho por el segundo director. En este pienso para gallinas que se desea formular pueden intervenir 13 ingredientes como materias primas y requiere de 16 nutrientes.

A continuación en la tabla 5.1 se muestra la limitación mínima y máxima que se ha fijado en cada uno de los ingredientes seleccionados, así como el precio por kilogramo de este. También se va a mostrar en la tabla 5.2 las especificaciones de concentración mínima y máxima de los nutrientes que se precisan.

nº	Ingrediente	Mínimo	Máximo	Precio
4	CEBADA 2C 11.3 PB	15	100	0.168
7	MAÍZ NACIONAL	0	100	0.178
12	TRIGO BLANDO 12.9 PB	0	20	0.171
36	GLUTEN FEED MAÍZ 21 %	0	100	0.190
74	HNA. GIRASOL 34	0	100	0.230
89	HNA. SOJA 47	0	100	0.382
166	AC. SOJA	0	100	0.702
171	AC. PALMA	0	100	0.697
193	CARBONATO CÁLCICO	0	100	0.135
200	FOSFATO BICÁLCICO ANH.	0	100	0.660
214	CLORURO SÓDICO MARINO 98	0	100	0.250
223	DL METIONINA	0	100	3.550
227	L-LISINA-HCL	0	100	2.000

Tabla 5.1: Tabla de ingredientes.

nº	Nutriente	Mínimo	Máximo
1	MATERIA SECA (%)	85	100
4	EE (%)	0	5
6	FB (%)	0	6
18	C18:2 (%)	1.20	100
21	Ca (%)	3.7	4.10
24	Pdisp. (%)	0.33	0.36
27	Na (%)	0.14	100
30	K (%)	0.45	0.90
38	Colina ppm	0	10000
50	EMA_AVES Kcal/Kg	2740	2800
67	LYS (%)	0.71	100
68	MET (%)	0.31	0.9
69	M+C (%)	0.61	100
70	THR (%)	0.5	100
71	TRP (%)	0.16	100
72	ILE (%)	0.59	100

Tabla 5.2: Tabla de nutrientes.

Los dos objetivos de este compuesto van a ser: minimizar el coste, siendo este un objetivo fijo en todas las dietas, y maximizar el nutriente número 68, la Metionina (nutriente esencial y limitante).

Este es un problema con un gran número variables (13 ingredientes) y de restricciones (16 nutrientes y la restricción fija), en el anexo A se puede consultar el problema escrito en forma matemática.

A continuación se pueden ver en la tabla 5.3 las tres soluciones eficientes halladas con el programa informático. De cada solución aparece el precio, la cantidad de Metionina y la cantidad de cada nutriente que posee esa solución, es decir el valor de los dos objetivos y de cada una de las variables.

Coste	0,219	0,220	0,238
68:MET (%)	0,343	0,374	0,900
CEBADA 2C 11.3 PB	15,000	15,000	15,000
MAÍZ NACIONAL	38,245	38,216	37,333
TRIGO BLANDO 12.9 PB	20,000	20,000	20,000
GLUTEN FEED MAÍZ 21 %	0,000	0,000	0,000
HNA. GIRASOL 34	0,000	0,000	0,641
HNA. SOJA 47	15,989	15,992	15,682
AC. SOJA	0,443	0,437	0,459
AC. PALMA	0,000	0,000	0,000
CARBONATO CALCICO	8,278	8,278	8,276
FOSFATO BICALCICO ANH.	1,584	1,584	1,583
CLORURO SODICO MARINO 98	0,311	0,311	0,311
DL METIONINA	0,109	0,140	0,670
L-LISINA HCL	0,041	0,041	0,046

Tabla 5.3: Tabla de soluciones.

Para finalizar la presentación de resultados de este ejemplo se muestra el gráfico de la figura 5.1, en el cual se muestra la imagen en el espacio de funciones el conjunto de soluciones eficientes encon-

tradas más todas las soluciones eficientes que se obtienen por la combinación lineal convexa de parejas adyacentes de estas (X_E).

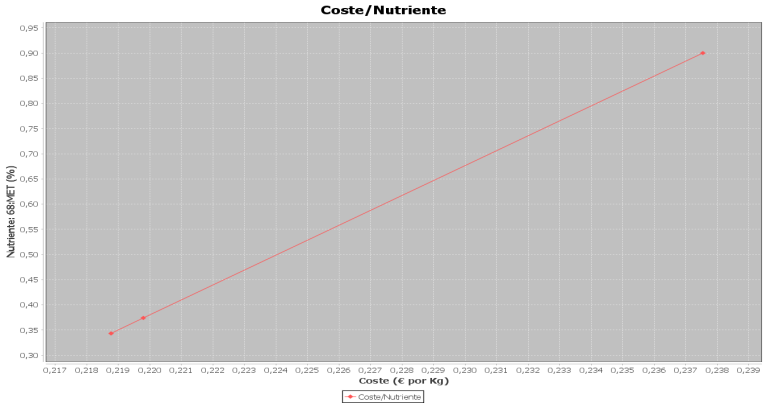

Figura 5.1: Gráfico Coste/Nutriente.

5.2. Conclusiones

Los objetivos plateados al inicio del trabajo se han logrado satisfactoriamente. Para comprobarlo se van a recordar estos objetivos y comentarlos.

En primer lugar se pretendía "*obtener y explicar los conocimientos teóricos mínimos que permitan entender, modelar y resolver este tipo de problemas multi-objetivo*". En el capítulo 2 se ha comenzado recordando en que consiste la optimización uni-objetivo para posteriormente ampliar este conocimiento a optimización multi-objetivo, se ha explicado el concepto de soluciones eficientes (las óptimas en este tipo de modelos) y finalmente se ha trabajado con estas soluciones mediante varios métodos. En el capítulo 3 se ha profundizado en el caso lineal de la optimización multi-objetivo (que es el que interesaba para su aplicación), se ha explicado la forma de obtener las soluciones eficientes con el método de los pesos y finalmente se ha trabajado con el método Simplex bi-objetivo (que es el que posteriormente se emplea para la formulación de piensos).

En segundo lugar se pretendía "*plasmar estos nuevos conocimientos en el software previamente desarrollado. Para ello se programará un módulo adicional para gestionar la incorporación de un segundo criterio de selección (este segundo objetivo puede ser un nutriente esencial y limitante como la Metionina, que es el primer aminoácido limitante en aves y segundo o tercero en porcino, aunque podría ser cualquier otro nutriente de interés para el nutricionista) y su posterior resolución y presentación visual de resultados*". Como se puede comprobar en el capítulo 4 y en el anexo B, se ha desarrollado este módulo adicional apropiadamente. La forma de utilizarlo, al igual que la anterior versión del sofware, es de fácil manejo y no requiere conocimientos matemáticos para su utilización, únicamente conocimientos sobre formulación de piensos.

A todo esto hay que añadir el hecho de enfrentarse a un problema real y obtener soluciones válidas. Al principio de este capítulo se ha mostrado un ejemplo real para gallinas (planteado en el anexo A), el cual ha sido resuelto con la aplicación creada y se han obtenido las soluciones eficientes del problema, de ellas el nutricionista podrá elegir la que más le conviene para cada momento.

En conclusión, con este Trabajo de Fin de Grado se han podido aplicar diferentes competencias adquiridas en el grado, ampliándolas con otros nuevos conocimientos y emplearlas en un problema real. Esto resulta muy atractivo y motivador, pues se ha podido comprobar el importante papel que tienen las matemáticas en la vida cotidiana y lo útiles que resultan.

Bibliografía

[1] AGUILAR A., *Desarrollo de un Programa Informático para la Formulación de Piensos Compuestos*, Trabajo de Fin de Grado, Universidad de Zaragoza, 2016.

[2] CONFEDERACIÓN ESPAÑOLA DE FABRICANTES DE ALIMENTOS COMPUESTOS PARA ANIMALES (CESFAC), *Mercados Estadística 2015*.

[3] EHRGOTT M., *Multicriteria Optimization*, 2nd edition, Springer, Heidelberg, 2005.

[4] GILBERT D., *The JFreeChart Class Library*, Developer Guide, Version 1.0.5, 2007.

[5] MAKHORIN A., *GNU Linear Programming Kit*, Reference Manual for GLPK, Version 4.58. Draft, 2016.

[6] MIETTINEN K. M., *Nonlinear Multiobjetive Optimization*, Kluwer Academic Publishers, Norwell, 35-41, 79, 88-89, 1999.

[7] MOULDER, N. D., *Investment outlook for the global animal protein industry. AFMA Forum 2016*, Animal Feed Manufacturers Association (AFMA). Sun City, South Africa. 1 to 3 March 2016.

[8] RÍOS S. Y RÍOS M.J., *Procesos de Decisión Multicriterio*, Eudema, Madrid, 108-109, 1989.

[9] SHEN, Y. B. AND WEAVER, A. C. AND KIM, S. W., *Effect of feed grade-methionine on growth performance and gut health in nursery pigs compared with conventional-methionine*, Journal of animal science, Vol 92, Nº 12, 5530-5539, 2014.

[10] SURRA, J. C., *Apuntes de formulación y tecnologías de la fabricación de piensos*, Servicio de reprografía Escuela Politécnica Superior - Universidad de Zaragoza, Huesca, 2017.

[11] SURRA, J. C., Comunicación personal, 2017.

Anexos

Anexo A

Planteamiento del ejemplo real

En este anexo se muestra cómo queda el planteamiento matemático del problema del capítulo 5. Este problema tiene como funciones objetivo minimizar el coste y maximizar la Metionina (nutriente 68). Las restricciones a las que está sujeto el problema son la cantidad de cada uno de los nutrientes que puede tener el pienso (la primera garantiza que la composición con esos ingredientes alcance el 100 %), por último se restringe la cantidad de cada uno de los ingredientes.

$$\begin{cases} \textbf{Min} \quad 0.168X_4 + 0.178X_7 + 0.171X_{12} + 0.190X_{36} + 0.230X_{74} + 0.382X_{89} + 0.702X_{166} + 0.697X_{171} \\ \qquad +0.135X_{193} + 0.660X_{200} + 0.250X_{214} + 3.550X_{223} + 2.00X_{227} \\ \\ \textbf{Max} \quad 0.19X_4 + 0.15X_7 + 0.21X_{12} + 0.37X_{36} + 0.77X_{74} + 0.67X_{89} + 99.0X_{223} \end{cases}$$

Sujeto a:

$$\begin{cases} X_4 + X_7 + X_{12} + X_{36} + X_{74} + X_{89} + X_{166} + X_{171} + X_{193} + X_{200} + X_{214} + X_{223} + X_{227} = 1 \\ \\ 85 \leq 89.9X_4 + 86.4X_7 + 89.7X_{12} + 88.7X_{36} + 89.9X_{74} + 88X_{89} + 100X_{166} + 100X_{171} + 98X_{193} \\ \qquad +99.7X_{200} + 98.3X_{214} + 99.6X_{223} + 98.5X_{227} \leq 100 \qquad 1 : MATERIA\ SECA\ (\%) \\ \\ 0 \leq 1.7X_4 + 3.3X_7 + 1.4X_{12} + 3X_{36} + 1.7X_{74} + 1.9X_{89} + 100X_{166} + 100X_{171} \leq 5 \qquad 4 : EE\ (\%) \\ \\ 0 \leq 4.7X_4 + 2.1X_7 + 2.4X_{12} + 7.7X_{36} + 20.9X_{74} + 4.1X_{89} \leq 6 \qquad 6 : FB\ (\%) \\ \\ 1.2 \leq 0.7X_4 + 1.66X_7 + 0.56X_{12} + 1.29X_{36} + 0.77X_{74} + 0.72X_{89} + 53.46X_{166} + 9.9X_{171} \leq 100 \\ \qquad 18 : C18 : 2\ (\%) \\ \\ 3.7 \leq 0.06X_4 + 0.03X_7 + 0.05X_{12} + 0.15X_{36} + 0.32X_{74} + 0.29X_{89} + 38.6X_{193} + 27X_{200} + 0.02X_{223} \\ \qquad \leq 4.10 \qquad 21 : Ca\ (\%) \\ \\ 0.33 \leq 0.11X_4 + 0.05X_7 + 0.15X_{12} + 0.29X_{36} + 0.19X_{74} + 0.2X_{89} + 14.67X_{200} \leq 0.36 \\ \qquad 24 : Pdisp.\ (\%) \\ \\ 0.14 \leq 0.02X_4 + 0.01X_7 + 0.02X_{12} + 0.2X_{36} + 0.03X_{74} + 0.02X_{89} + 0.07X_{193} + 0.03X_{200} + 38.5X_{214} \\ \qquad \leq 100 \qquad 27 : Na\ (\%) \\ \\ 0.45 \leq 0.4X_4 + 0.29X_7 + 0.32X_{12} + 0.9X_{36} + 1.55X_{74} + 2.2X_{89} + 0.07X_{193} \leq 0.90 \qquad 30 : K\ (\%) \\ \\ 0 \leq 1025X_4 + 500X_7 + 890X_{12} + 1450X_{36} + 2750X_{74} + 2740X_{89} \leq 10000 \qquad 38 : Colina\ ppm \\ \\ 2740 \leq 2780X_4 + 3285X_7 + 3215X_{12} + 1890X_{36} + 1615X_{74} + 2360X_{89} + 9000X_{166} + 8150X_{171} \\ \qquad +4500X_{223} + 3800X_{227} \leq 2800 \qquad 50 : EMA.AVES\ Kcal/Kg \end{cases}$$

$$\begin{cases} 0.71 \leq 0.41X_4 + 0.22X_7 + 0.36X_{12} + 0.66X_{36} + 1.22X_{74} + 2.88X_{89} + 78X_{227} \leq 100 & 67 : LYS\ (\%) \\ 0.31 \leq 0.19X_4 + 0.15X_7 + 0.21X_{12} + 0.37X_{36} + 0.77X_{74} + 0.67X_{89} + 99X_{223} \leq 0.9 & 68 : MET\ (\%) \\ 0.61 \leq 0.43X_4 + 0.31X_7 + 0.49X_{12} + 0.8X_{36} + 1.37X_{74} + 1.38X_{89} + 99X_{223} \leq 100 & 69 : M+C\ (\%) \\ 0.5 \leq 0.37X_4 + 0.27X_7 + 0.37X_{12} + 0.76X36 + 1.23X_{74} + 1.85X_{89} \leq 100 & 70 : THR\ (\%) \\ 0.16 \leq 0.14X_4 + 0.06X_7 + 0.15X_{12} + 0.14X_{36} + 0.44X_{74} + 0.63X_{89} \leq 100 & 71 : TRP\ (\%) \\ 0.59 \leq 0.4X_4 + 0.26X_7 + 0.45X_{12} + 0.65X_{36} + 1.39X_{74} + 2.13X_{89} \leq 100 & 72 : ILE\ (\%) \end{cases}$$

$$\begin{cases} 0.15 \leq X_4 \leq 1.0 & 4 : CEBADA\ 2C\ 11.3\ PB \\ 0.0 \leq X_7 \leq 1.0 & 7 : MAIZ\ NACIONAL \\ 0.0 \leq X_{12} \leq 0.2 & 12 : TRIGO\ BLANDO\ 12.9\ PB \\ 0.0 \leq X_{36} \leq 1.0 & 36 : GLUTEN\ FEED\ MAIZ\ 21\% \\ 0.0 \leq X_{74} \leq 1.0 & 74 : HNA.\ GIRASOL\ 34 \\ 0.0 \leq X_{89} \leq 1.0 & 89 : HNA.\ SOJA47 \\ 0.0 \leq X_{166} \leq 1.0 & 166 : AC.\ SOJA \\ 0.0 \leq X_{171} \leq 1.0 & 171 : AC.\ PALMA \\ 0.0 \leq X_{193} \leq 1.0 & 193 : CARBONATO\ CALCICO \\ 0.0 \leq X_{200} \leq 1.0 & 200 : FOSFATO\ BICALCICO\ ANH. \\ 0.0 \leq X_{214} \leq 1.0 & 214 : CLORURO\ SODICO\ MARINO\ 98 \\ 0.0 \leq X_{223} \leq 1.0 & 223 : DL\ METIONINA \\ 0.0 \leq X_{227} \leq 1.0 & 227 : L-LISINA\ HCL \end{cases}$$

Anexo B

Código del programa: clase PPLsBiObjetivo

```
1  package programadietas;
2
3  import java.text.DecimalFormat;
4  import java.util.Arrays;
5  import javax.swing.JFrame;
6  import javax.swing.JOptionPane;
7  import javax.swing.JTable;
8  import javax.swing.JTextArea;
9  import org.gnu.glpk.GLPK;
10 import org.gnu.glpk.GLPKConstants;
11 import static org.gnu.glpk.GLPKConstants.GLP_OPT;
12 import org.gnu.glpk.GlpkException;
13 import org.gnu.glpk.SWIGTYPE_p_double;
14 import org.gnu.glpk.SWIGTYPE_p_int;
15 import org.gnu.glpk.glp_prob;
16 import org.gnu.glpk.glp_smcp;
17 import org.jfree.chart.ChartFactory;
18 import org.jfree.chart.ChartPanel;
19 import org.jfree.chart.JFreeChart;
20 import org.jfree.chart.plot.PlotOrientation;
21 import org.jfree.chart.plot.XYPlot;
22 import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;
23 import org.jfree.data.xy.XYDataset;
24 import org.jfree.data.xy.XYSeries;
25 import org.jfree.data.xy.XYSeriesCollection;
26 import org.jfree.util.ShapeUtilities;
27 import static programadietas.datosGlobales.A;
28 import static programadietas.datosGlobales.bL;
29 import static programadietas.datosGlobales.bU;
30 import static programadietas.datosGlobales.c;
31 import static programadietas.datosGlobales.c1;
32 import static programadietas.datosGlobales.c2;
33 import static programadietas.datosGlobales.cm1;
34 import static programadietas.datosGlobales.cm2;
35 import static programadietas.datosGlobales.l;
36 import static programadietas.datosGlobales.u;
37 import static programadietas.datosGlobales.nutMax;
38 import static programadietas.datosGlobales.matriz;
39 import static programadietas.datosGlobales.matrizN;
40
41 /**
42 *
43 * @author Marta
44 */
```

```
45
47 public class PPLsBiObjetivo {
49
   protected glp_prob lp=null;
51 protected glp_smcp parm=null;
   protected JTable ingredientes;
53 protected JTable nutrientes;
   protected JTextArea texto;
55 SWIGTYPE_p_int indices;
   SWIGTYPE_p_double valores;
57 protected int numVar;
   protected int numRes;
59 protected double lambda;
   protected double viejolambda;
61 protected double delta=1E-10;
   protected int contador;
63
65
   PPLsBiObjetivo(JTable ingredientesArg,JTable nutrientesArg,JTextArea textoArg){
67
     ingredientes=ingredientesArg;
69   nutrientes=nutrientesArg;
     texto=textoArg;
71
     lp = GLPK.glp_create_prob();
73   GLPK.glp_set_prob_name(lp,"sinNombre") ;
75   //Variables
     GLPK.glp_add_cols(lp,ingredientes.getRowCount());
77   for (int i=0; i< ingredientes.getRowCount(); i++){
       GLPK.glp_set_col_name(lp, i+1, ""+ingredientes.getValueAt(i,0));
79     GLPK.glp_set_col_kind(lp, i+1,GLPKConstants.GLP_CV);
       GLPK.glp_set_col_bnds(lp, i+1,GLPKConstants.GLP_DB,l[i],u[i]);
81   }
     indices = GLPK.new_intArray(ingredientes.getRowCount()+1);
83   valores = GLPK.new_doubleArray(ingredientes.getRowCount()+1);
     GLPK.glp_add_rows(lp,nutrientes.getRowCount()+1);
85
     //Primera restricción suma igual a 1
87   GLPK.glp_set_row_name(lp,1,"Total");
     GLPK.glp_set_row_bnds(lp,1,GLPKConstants.GLP_FX,1.0,1.0);
89   for (int i=0; i< ingredientes.getRowCount(); i++){
       GLPK.intArray_setitem(indices, i+1, i+1);
91     GLPK.doubleArray_setitem(valores, i+1, 1.0);
     }
93   GLPK.glp_set_mat_row(lp,1,ingredientes.getRowCount(),indices,valores);
     GLPK.delete_intArray(indices);
95   GLPK.delete_doubleArray(valores);
97   //Restricciones de nutrientes
     for(int j=0;j<nutrientes.getRowCount();j++){
99     indices = GLPK.new_intArray(ingredientes.getRowCount()+1);
       valores = GLPK.new_doubleArray(ingredientes.getRowCount()+1);
101    GLPK.glp_set_row_name(lp,j+2,""+nutrientes.getValueAt(j, 0));
       GLPK.glp_set_row_bnds(lp,j+2,GLPKConstants.GLP_DB,bL[j],bU[j]);
103    for (int i=0; i< ingredientes.getRowCount(); i++){
         GLPK.intArray_setitem(indices, i+1, i+1);
105      GLPK.doubleArray_setitem(valores, i+1, A[j][i]);
       }
107    GLPK.glp_set_mat_row(lp,j+2,ingredientes.getRowCount(),indices,valores);
```

```
      GLPK.delete_intArray(indices);
109   GLPK.delete_doubleArray(valores);
    }
111

113 //Función objetivo
    GLPK.glp_set_obj_dir(lp,GLPK.GLP_MIN);
115 GLPK.glp_set_obj_name(lp, "Z");
    GLPK.glp_set_obj_coef(lp,0,0.0);
117 for (int i=0; i< ingredientes.getRowCount(); i++){
      GLPK.glp_set_obj_coef(lp,i+1,c1[i]);
119 }

121
    //Guardar el problema en formato cplex
123 GLPK.glp_write_lp(lp, null, "sinNombre.txt");
    numVar = GLPK.glp_get_num_cols(lp);
125 numRes = GLPK.glp_get_num_rows(lp);

127 parm= new glp_smcp();
    GLPK.glp_init_smcp(parm);
129 parm.setMsg_lev(0);

131 }

133 PPLsBiObjetivo() {
    throw new UnsupportedOperationException("Not supported yet.");
135 }

137


139 double [] costosMarginales1(){
    double res[]= new double[numRes+numVar];
141 for(int i=0;i<numRes+numVar;i++) res[i]=0.0;

143 indices = GLPK.new_intArray(ingredientes.getRowCount()+1);
    valores = GLPK.new_doubleArray(ingredientes.getRowCount()+1);
145
    int h=0;
147 for (int i=0; i< ingredientes.getRowCount(); i++){
      if(c1[i]!=0){
149     GLPK.intArray_setitem(indices, h+1, i+1);
        GLPK.doubleArray_setitem(valores, h+1, c1[i]);
151     h++;
      }
153 }

155 int lenp=GLPK.glp_transform_row(lp, h, indices, valores);
    for(int i=1;i<=lenp;i++){
157   res[GLPK.intArray_getitem(indices,i)-1]=GLPK.doubleArray_getitem(valores, i);
    }
159 return res;
}
161

163 double [] costosMarginales2(){
    double res[]= new double[numRes+numVar];
165 for(int i=0;i<numRes+numVar;i++) res[i]=0.0;

167 indices = GLPK.new_intArray(ingredientes.getRowCount()+1);
    valores = GLPK.new_doubleArray(ingredientes.getRowCount()+1);
169
    int h=0;
```

```
171     for (int i=0; i< ingredientes.getRowCount(); i++){
          if(c2[i]!=0){
173         GLPK.intArray_setitem(indices, h+1, i+1);
            GLPK.doubleArray_setitem(valores, h+1, c2[i]);
175         h++;
          }
177     }

179     int lenp=GLPK.glp_transform_row(lp, h, indices, valores);
        for(int i=1;i<=lenp;i++){
181       res[GLPK.intArray_getitem(indices,i)-1]=GLPK.doubleArray_getitem(valores, i);
        }
183     return res;
    }
185

187   double calculaLambda(){
        int contador = 0;
189     int temp[]= new int[numRes+numVar];

191     for (int i = 1; i <= numRes; i++) {
          if(GLPK.glp_get_row_stat(lp, i) == GLPK.GLP_NL && cm2[i - 1] < 0
193           && cm1[i - 1] >= 0){
            temp[contador++]=i-1;
195       }
          else if(GLPK.glp_get_row_stat(lp, i) == GLPK.GLP_NU && cm2[i - 1] > 0
197             && cm1[i - 1] <= 0){
            temp[contador++]=i-1;
199       }
        }
201     for (int i = 1; i <= numVar; i++) {
          if(GLPK.glp_get_col_stat(lp, i) == GLPK.GLP_NL && cm2[numRes+i - 1] < 0
203           && cm1[numRes+i - 1] >= 0){
            temp[contador++]=numRes+i-1;
205       }
          else if(GLPK.glp_get_col_stat(lp, i) == GLPK.GLP_NU && cm2[numRes+i - 1] > 0
207             && cm1[numRes+i - 1] <= 0){
            temp[contador++]=numRes+i-1;
209       }
        }
211

        double lambda=0;
213     for(int i=0;i<contador;i++){
          if(-cm2[temp[i]]/(cm1[temp[i]]-cm2[temp[i]])>lambda)
215         lambda=-cm2[temp[i]]/(cm1[temp[i]]-cm2[temp[i]]);
        }
217     return lambda;
    }
219

221   void modificaProblema(double lambda){
        lambda-=delta;
223     for(int i=0;i<numVar;i++){
          c[i]=lambda*c1[i]+(1-lambda)*c2[i];
225     }
        for(int i=0;i<numVar;i++){
227       GLPK.glp_set_obj_coef(lp, i+1, c[i]);
        }
229   }

231

      void resuelveProblema(){
233     boolean salir = false;
```

```
    lambda=1;
    contador=1;
    matriz=new String [numVar+2][contador];
    matrizN=new double [2][contador];
    matriz[0][contador-1]=String.format("%-35s","Coste");
    matriz[1][contador-1]=String.format("%-35s","Nutriente: "+nutMax);
    for(int i=2;i<matriz.length;i++){
      matriz[i][contador-1]=String.format("%-35s",GLPK.glp_get_col_name(lp, i-1));
    }

    do {
      try {
        GLPK.glp_simplex(lp, parm);

        if(GLPK.glp_get_status(lp)!=GLP_OPT){
          JOptionPane.showMessageDialog(null, "El problema es no factible.
                                               No se puede resolver.");
          return;
        }

        cm1 = costosMarginales1();
        cm2 = costosMarginales2();

        viejolambda=lambda;
        lambda=calculaLambda();
        if(lambda!=viejolambda){
          contador++;
          for(int i=0;i<matriz.length;i++){
            matriz[i]=Arrays.copyOf(matriz[i], contador);
          }
          for(int i=0;i<matrizN.length;i++){
            matrizN[i]=Arrays.copyOf(matrizN[i], contador);
          }
          if(lambda<=delta){
            salir=true;
            lambda=0;
          }
          matriz[0][contador-1]=String.format("%1$9.3f",costoProblema());
          matriz[1][contador-1]=String.format("%1$9.3f",nutrienteMaximizado());
          matrizN[0][contador-1]=costoProblema();
          matrizN[1][contador-1]=nutrienteMaximizado();
          for(int i=2;i<matriz.length;i++){
            matriz[i][contador-1]=
            String.format("%1$9.3f", 100*GLPK.glp_get_col_prim(lp, i-1));
          }
          escribeSolucion();
          delta=1E-10;
        }else{
          delta*=1.1;
        }
        if(lambda!=0){
          modificaProblema(lambda);
        }
      }catch(GlpkException ex) {
        ex.printStackTrace();
        GLPK.glp_delete_prob(lp);
      }
    }while(!salir);

    escribeTabla(matriz);
    dibujaGrafico("Coste/Nutriente");
  }
```

```
297
    double costoProblema(){
299   double costo=0;
      for (int i = 0; i < numVar; i++){
301     costo+=GLPK.glp_get_col_prim(lp, i + 1)*c1[i];
      }
303   return costo;
    }
305

307 double nutrienteMaximizado(){
      double nutrienteMax=0;
309   for (int i = 0; i < numVar; i++){
        nutrienteMax+=GLPK.glp_get_col_prim(lp, i + 1)*(-c2[i]);
311   }
      return nutrienteMax;
313 }

315 void escribeSolucion(){
      DecimalFormat decimales = new DecimalFormat("0.00");
317
      texto.append("\n\n\n\n\nCon lambda perteneciente a: ["+String.valueOf(lambda)+
319   ","+String.valueOf(viejolambda)+"]\n");

321   double costo = costoProblema();
      texto.append(String.format("\nSolución de costo:   %7.4f ? por Kg\n", costo));
323
      double nutrienteMax = nutrienteMaximizado();
325   texto.append(String.format("\nSolución con:   %7.4f  del nutriente  ",
      nutrienteMax) + nutMax +String.format("\n\n"));

327   texto.append("\nIngredientes utilizados: \n");
      for(int i=1;i<=GLPK.glp_get_num_cols(lp);i++){
329     if (GLPK.glp_get_col_prim(lp, i)!=0)
          texto.append( String.format("%1$-40s",GLPK.glp_get_col_name(lp,i)+":")+
          String.format("%10.3f %%%n",100*GLPK.glp_get_col_prim(lp, i)));
331   }

333   texto.append("\nIngredientes no utilizados: \n");
      for(int i=1;i<=GLPK.glp_get_num_cols(lp);i++){
335     if (GLPK.glp_get_col_prim(lp, i)==0)
          texto.append( String.format("%1$-40s",GLPK.glp_get_col_name(lp,i)+":")+
          String.format("%10.3f %%%n",100*GLPK.glp_get_col_prim(lp, i)));
337   }

339   texto.append("\nComposición lograda: \n");
      for (int j=1;j<GLPK.glp_get_num_rows(lp);j++){
341     texto.append(String.format("%1$-22s",GLPK.glp_get_row_name(lp, j+1)+":")+
        String.format("%10.3f ",GLPK.glp_get_row_prim(lp, j+1))+
        " en ("+ String.format("%1$10.3f",GLPK.glp_get_row_lb(lp,j+1))+","+
        String.format("%1$10.3f",GLPK.glp_get_row_ub(lp,j+1))+")\n");
343   }
    }
345

347 void dibujaGrafico(String title) {
      double minY=matrizN[1][1];
349   double maxY=matrizN[1][1];
      double maxPrec=matrizN[0][1];
351   double minPrec=matrizN[0][1];

353   XYSeries series=new XYSeries("Coste/Nutriente");
      XYDataset dataset = new XYSeriesCollection(series);
```

```
355
        for(int i=1;i<contador;i++){
357       series.add(matrizN[0][i],matrizN[1][i]);
          if(maxY<matrizN[1][i])
359         maxY=matrizN[1][i];
          if(minY>matrizN[1][i])
361         minY=matrizN[1][i];
          if(maxPrec<matrizN[0][i])
363         maxPrec=matrizN[0][i];
          if(minPrec>matrizN[0][i])
365         minPrec=matrizN[0][i];
        }
367
        if(maxY-minY!=0){
369       minY=minY-(maxY-minY)/10;
          maxY=maxY+(maxY-minY)/10;
371     }else{
          if(maxY!=0){
373         minY=minY-minY/10;
            maxY=maxY+maxY/10;
375       }else{
            maxY=0.01;
377         minY=-0.01;
        }
379
        }
381     if(minPrec-maxPrec!=0){
          maxPrec=maxPrec-(minPrec-maxPrec)/10;
383       minPrec=minPrec+(minPrec-maxPrec)/10;
          }else{
385         if(maxPrec!=0){
              minPrec*=0.9;
387           maxPrec*=1.1;
            }else{
389           maxPrec=0.01;
              minPrec=-0.01;
391       }
        }
393
        JFreeChart chart = ChartFactory.createXYLineChart("Coste/Nutriente", "Coste (?
        por Kg)", matriz[1][0], dataset, PlotOrientation.VERTICAL, true, true, true);
395
        XYPlot plot = (XYPlot) chart.getPlot();
397     plot.getRangeAxis().setRange(minY,maxY);
        plot.getDomainAxis().setRange(minPrec,maxPrec);
399
        XYLineAndShapeRenderer renderer = (XYLineAndShapeRenderer) plot.getRenderer();
401     renderer.setSeriesShapesVisible(0, Boolean.TRUE);
        renderer.setSeriesShapesFilled(0, Boolean.TRUE);
403     renderer.setSeriesShape(0, ShapeUtilities.createDiamond(3));

405     ChartPanel chartPanel = new ChartPanel(chart);
        chartPanel.setPreferredSize(new java.awt.Dimension(500, 270));
407     JFrame jframe1= new JFrame();
        jframe1.setSize(400, 400);
409     jframe1.getContentPane().add(chartPanel);
        jframe1.setVisible(true);
411   }


413
}
```