Memoria Práctica Níger

Grupo 17: Irene San Andrés Gutiérrez, Santiago Gutiérrez Taillefer
Grupo 18: Francisco Gutiérrez Molero, Carlos Mora Martin

21 abril 2022

Aclaración

En esta práctica solo hemos hecho por separado el método de las ponderaciones (Grupo 18) y el de las e-restricciones (Grupo 17). Todo lo demás se ha hecho en conjunto.
Por otro lado, hay que indicar en el código de Gams (linea 130) la ruta en la que se quiera guardar el archivo de texto en el que se escriben los resultados (con nombre incluido .txt), si no se pone no pasa nada.

1 Introducción

El problema que se nos plantea viene dado por una crisis alimentaria que tuvo lugar en Niger durante el año 2005. Frente a esta situación FARMAMUNDI y CADEV diseñaron una respuesta destinada a paliar parte de los efectos derivados de la hambruna

Se recibieron 850 toneladas de ayuda que había que enviar para ser repartidas entre 9 poblaciones de la zona, In-Gall, Aderbissinat, Tessaoua, Dakoro, Tatokou, Bakatchiraba, Mayahi, Koundoumaoua, y Sabon Kafi, desde centros de reparto que podrían estar ubicados en Tanaut, Zinder, Agadez o Maradi.
Se trata en un primer momento de decidir dónde ubicar los centros y cómo abastecer a las poblaciones desde ellos, teniendo en cuenta que:

a) Se tiene un presupuesto total de 40.000 euros.

b) Por problemas logísticos, no se pueden abrir más de 3 centros.

c) Dado el estado de las carreteras, desde cada localidad solo se puede viajar a recoger ayuda a centros que estén a menos de 200 Kilómetros

d) Hay que distribuir al menos un 60% de la demanda global.

e) Deseamos minimizar la demanda no satisfecha.

f) Deseamos minimizar el coste total de la operación.

g) Deseamos que el reparto sea lo más equitativo posible entre localidades.

2 Índices del problema

Los índices son los valores de distintos parámetros que se recorren en los bucles para generalizar las restricciones.

1. **Centros de reparto** (r): Recorre todos los centros de reparto.
2. **Poblaciones** (p): Recorre todas las poblaciones a las que se quiere proporcionar ayuda.
3. **Objetivos** (o): Recorre todos los objetivos que queremos optimizar en nuestro problema

3 Datos del problema

1. **d(p)**: Demanda que cada población pide para cubrir sus necesidades.
2. **eurton(r)**: Coste por tonelada de ayuda que el centro r proporciona.
3. **k**: Coste fijo por construir un centro.
4. **maxpresup**: Presupuesto máximo de nuestro proyecto
5. **tontot**: Toneladas totales de ayuda de las que disponemos.
6. **dist(p,r)**: Distancia que hay del poblado p al centro de reparto r.
7. **distmax(p,r)**: Matriz que nos dice cuáles son los trayectos posibles teniendo en cuenta la restricción de distancia.

4 Variables del problema

Definición de las variables del problema, el porque son así o como se utilizan esta explicado en la sección de restricciones.

1. **X(r)**: Variable binaria que se hace 1 si el centro r se construye y 0 en caso contrario.
2. **C(p,r)**:Variable que indica la cantidad de toneladas de ayuda que la población p recibe del centro r.
3. **SI_C(p,r)**: Variable binaria que se hace 1 si el centro r proporciona ayuda a la población p y 0 en caso contrario.
4. **MAXPROP**: Variable que indica la máxima proporción de demanda no satisfecha.
5. **TON(r)**: Variable que indica las toneladas que reparte el centro r.
6. **NOSAT(p)**: Variable que indica las toneladas no satisfechas de la población p.

5 Modelización del problema

Funciones Objetivo

En primer lugar veamos como, al tratarse de un problema de decisión multicriterio, tenemos más de una función objetivo. En nuestro caso tenemos 4, las cuales son:

Minimizar la demanda no satisfecha

Minimizamos la suma de la demanda no satisfecha de cada una de las poblaciones.

$$\text{MIN: } \sum_{p}(d(p) - \sum_{r} C(p,r))$$

Minimizar el coste total de la operación

Minimizamos el coste total, que será la suma del coste fijo por construcción (solo si se construye el centro r) + el gasto por tonelada de ayuda proporcionada.

$$\text{MIN: } \sum_r (k \cdot X(r) + \sum_p C(p,r) \cdot eurton(r))$$

Minimizar la máxima proporción de demanda no satisfecha

Al buscar la mayor equidad en el reparto de ayuda, seguimos el criterio de minimizar la máxima proporción de demanda no satisfecha, de manera que no haya ninguna población que se quede sin ayuda.

$$\text{MIN: } MAXPROP$$

Minimizar la distancia recorrida

Repartiremos la ayuda de manera que los viajes entre población y centro sean lo más cortos posibles.

$$\text{MIN: } \sum_{p,r} dist(p,r) \cdot SI\_C(p,r)$$

Restricciones

Una vez explicadas nuestras funciones objetivos veamos que más restricciones tiene el problema.

• No se puede superar el presupuesto máximo.

$$\sum_r (1000 \cdot X(r) + \sum_p C(p,r) \cdot eurton(r)) \leq maxpresup \tag{1}$$

• No se pueden construir más de 3 centros de reparto.

$$\sum_r X(r) \leq 3 \tag{2}$$

• No se puede dar más ayuda de la que se tiene.

$$\sum_{p,r} C(p,r) \leq tontot \tag{3}$$

• Debe cubrirse al menos un 60% de la demanda total.

$$\sum_{p,r} C(p,r) \geq 0.6 \cdot \sum_p d(p) \tag{4}$$

• Una localidad no debe recibir más ayuda de la que pide.

$$\sum_r C(p,r) \leq d(p) \quad \forall p \tag{5}$$

• Si C(p,r) mayor que 1 entonces el poblado p recibe ayuda del centro r.

$$C(p,r) \leq d(p) \cdot SI\_C(p,r) \quad \forall p \ \forall r \tag{6}$$

• Si la distancia entre un poblado y un centro es mayor de 200km, entonces ese centro no proporciona ayuda a esa población. Además, si un centro no se construye, entonces no puede proporcionar ayuda.

$$C(p,r) \leq distmax(p,r) \cdot d(p) \cdot X(r) \quad \forall p \ \forall r \tag{7}$$

• Todas las proporciones de demandas nos satisfechas deben ser menor o igual que nuestra variable MAXPROP.

$$\frac{d(p) - \sum_r C(p,r)}{d(p)} \leq MAXPROP \quad \forall p \tag{8}$$

Modelo de forma compacta

$$\text{MIN: } \sum_p (d(p) - \sum_r C(p,r)) \qquad \text{MIN: } MAXPROP$$

$$\text{MIN: } \sum_r (k \cdot X(r) + \sum_p C(p,r) \cdot eurton(r)) \qquad \text{MIN: } \sum_{p,r} dist(p,r) \cdot SI\_C(p,r)$$

Sujeto a:

$$\sum_r (1000 \cdot X(r) + \sum_p C(p,r) \cdot eurton(r)) \leq maxpresup$$

$$\sum_r X(r) \leq 3$$

$$\sum_{p,r} C(p,r) \leq tontot$$

$$\sum_{p,r} C(p,r) \geq 0.6 \cdot \sum_p d(p)$$

$$\sum_r C(p,r) \leq d(p) \quad \forall p$$

$$C(p,r) \leq d(p) \cdot SI\_C(p,r) \quad \forall p \ \forall r$$

$$C(p,r) \leq d(p) \cdot SI\_C(p,r) \quad \forall p \ \forall r$$

$$C(p,r) \leq distmax(p,r) \cdot d(p) \cdot X(r) \quad \forall p \ \forall r$$

$$\frac{d(p) - \sum_r C(p,r)}{d(p)} \leq MAXPROP \quad \forall p$$

6 Frontera de Pareto

Definimos el conjunto de todas las soluciones eficientes como frontera de Pareto. En este apartado hallaremos la Frontera de Pareto tanto del par "demanda no satisfecha-equidad", como del par "demanda no satisfecha-coste" a través de dos métodos distintos. Por el método de las ponderaciones y por el método de las ε-restricciones.

Matriz de Pagos

En primer lugar, hallaremos la matriz de pagos de los cuatro objetivos, que nos ayudará a la hora de calcular las fronteras de Pareto. La matriz de pago se obtiene de optimizar nuestro problema teniendo en cuenta un único objetivo (el cual fijamos) y después calcular el valor de los otros minimizándolos uno a uno para que no haya problemas con las variables que necesitan ser minimizadas.

De esta manera, en nuestro caso, obtenemos la siguiente matriz de pagos:

$$\begin{array}{c} \\ obj1 \\ obj2 \\ obj3 \\ obj4 \end{array} \begin{array}{c} \begin{array}{cccc} obj1 & obj2 & obj3 & obj4 \end{array} \\ \begin{pmatrix} 119.000 & 38880.000 & 0.427 & 2170.000 \\ 387.600 & 20690.000 & 1.000 & 1804.000 \\ 199.457 & 40000.000 & 0.206 & 2374.000 \\ 354.000 & 30206.000 & 1.000 & 788.000 \end{pmatrix} \end{array}$$

Método de las Ponderaciones

El método de las ponderaciones consiste en multiplicar cada objetivo por un peso o factor no negativo y agregarlos en una única función como la siguiente

$$\text{MIN: } p_1 \cdot fobj_1 + p_2 \cdot fobj_2 + p_3 \cdot fobj_3 + p_4 \cdot fobj_4$$

De esta manera, variando los pesos de cada objetivo con un paso h obtenemos todo el conjunto eficiente, es decir, la frontera de Pareto.

Es necesario mencionar, que al tratarse de magnitudes distintas en cada una de las funciones objetivos, hay que normalizar. Para ello, hemos dividido p_i entre $(maxfobj_i - minfobj_i)$, quedando entonces la función objetivo de la siguiente forma.

$$\text{MIN: } \sum_i \frac{p_i}{maxfobj_i - minfobj_i}$$

En nuestro caso obtenemos el siguiente conjunto de soluciones eficientes:

"Demanda no satisfecha-equidad"

$$\text{MIN: } \frac{p_1}{(387.6-119)} \cdot (\sum_p (d(p) - \sum_r C(p,r))) + \frac{p_2}{(1-0.206)} \cdot MAXPROB$$

Conjunto de puntos eficientes tomando $h = \frac{1}{30}$

$$\begin{array}{c} \\ Punto1 \\ Punto2 \\ Punto3 \end{array} \begin{array}{c} \begin{array}{cc} obj1 & obj3 \end{array} \\ \begin{pmatrix} 199.457 & 0.206 \\ 139.120 & 0.309 \\ 119.000 & 0.427 \end{pmatrix} \end{array}$$

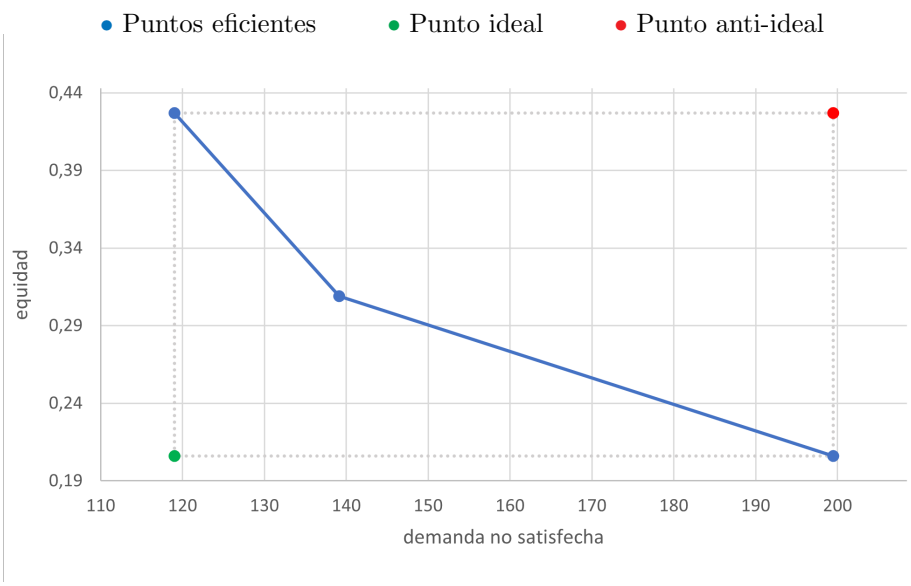

Figure 1: Frontera de Pareto *"Demanda no satisfecha-equidad"*

Podemos ver como, aun tomando un paso h muy pequeño el conjunto de soluciones eficientes solo está compuesto por 3 puntos, es decir, no existen más puntos de la región factible del problema que al mejorar uno de los objetivos no empeore el otro.

"Demanda no satisfecha-coste"

$$\text{MIN: } \frac{p_1}{(387.6-119)} \cdot (\sum_p (d(p) - \sum_r C(p,r))) + \\ \frac{p_2}{(40000-20690)} \cdot (\sum_r (k \cdot X(r) + \sum_p C(p,r) \cdot eurton(r)))$$

Conjunto de puntos eficientes tomando $h = \frac{1}{30}$

$$\begin{array}{c} \\ Punto1 \\ Punto2 \\ Punto3 \end{array} \begin{array}{c} \begin{array}{cc} obj1 & obj2 \end{array} \\ \begin{pmatrix} 387.600 & 20690.000 \\ 213.000 & 29420.000 \\ 119.000 & 38880.000 \end{pmatrix} \end{array}$$

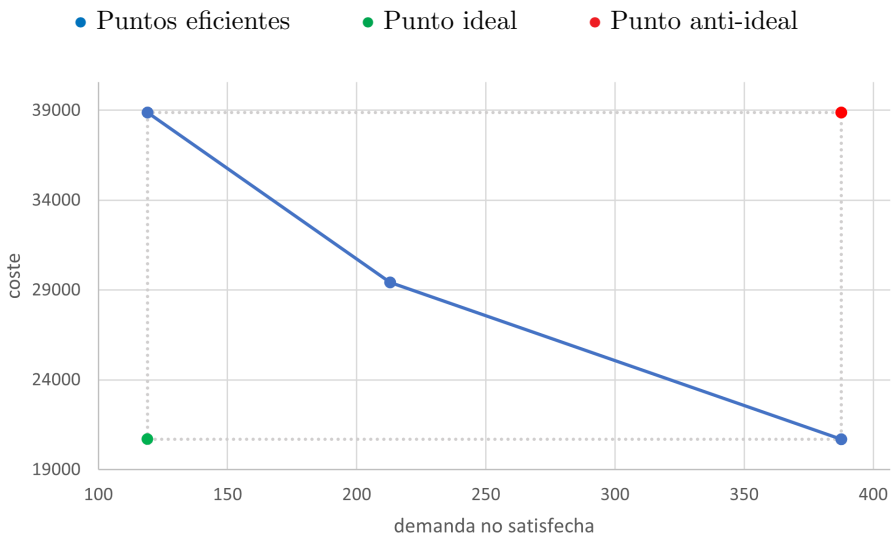

Figure 2: Frontera de Pareto *"Demanda no satisfecha-coste"*

Al igual que antes, nuestro conjunto de soluciones eficientes, está formado únicamente por tres puntos.

Método de las ε-restricciones

Este método consiste en optimizar uno de los objetivos e incorporar el resto como restricciones paramétricas de la siguiente forma.

$$\text{MIN: } Obj_1$$

$$\text{Sujeto a: } Obj_2 \leq \varepsilon_k$$

Variando el término ε_k con un paso h, se obtiene el conjunto de puntos eficientes, es decir, la frontera de Pareto. Teniendo en cuenta como se desarrolla este método, es lógico pensar que no es lo mismo minimizar $fobj_1$ y restringir $fobj_3$, que minimizar $fobj_3$ y restringir $fobj_1$. Por esta razón, obtendremos 4 fronteras de Pareto.

"Minimizar demanda no satisfecha y restringir equidad"

$$\text{MIN: } \sum_p (d(p) - \sum_r C(p,r))$$

$$\text{Sujeto a: } MAXPROB \leq \varepsilon_k$$

Conjunto de puntos eficientes tomando $h = \frac{1}{40}$

$$\begin{array}{c} \\ Obj1 \\ Obj3 \end{array} \begin{array}{c} \begin{array}{ccccccc} Punto1 & Punto2 & Punto3 & & Punto38 & Punto39 & Punto40 \end{array} \\ \begin{pmatrix} 119.000 & 119.943 & 120.886 & \cdots & 192.993 & 196.225 & 199.457 \\ 0.427 & 0.422 & 0.416 & \cdots & 0.217 & 0.211 & 0.206 \end{pmatrix} \end{array}$$

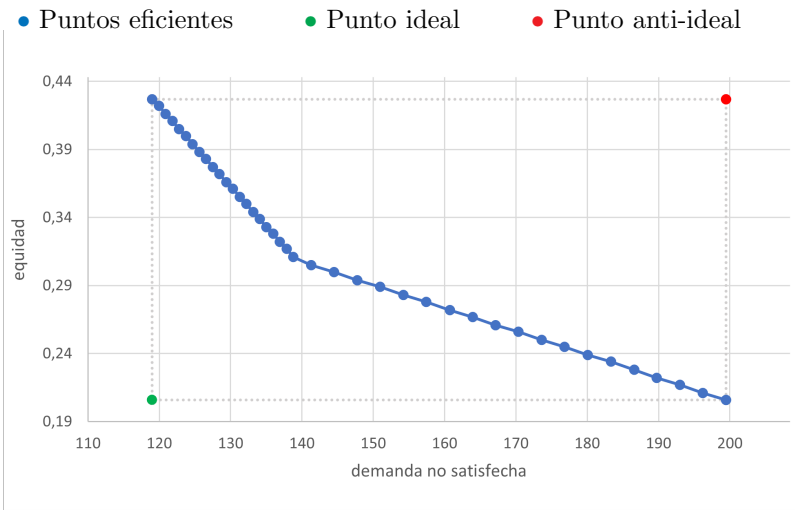

Figure 3: Frontera de Pareto *"Demanda no satisfecha-equidad"*

"Minimizamos para buscar la mayor equidad y restringir demanda no satisfecha"

$$\text{MIN: } MAXPROB$$
$$\text{Sujeto a: } \sum_p (d(p) - \sum_r C(p,r)) \leq \varepsilon_k$$

Conjunto de puntos eficientes tomando $h = \frac{1}{40}$

$$\begin{array}{c} \\ Obj1 \\ Obj3 \end{array} \begin{array}{c} \begin{array}{ccccccc} Punto1 & Punto2 & Punto3 & & Punto38 & Punto39 & Punto40 \end{array} \\ \begin{pmatrix} 199.457 & 197.446 & 195.434 & \cdots & 123.023 & 121.011 & 119.000 \\ 0.206 & 0.209 & 0.213 & \cdots & 0.404 & 0.415 & 0.427 \end{pmatrix} \end{array}$$

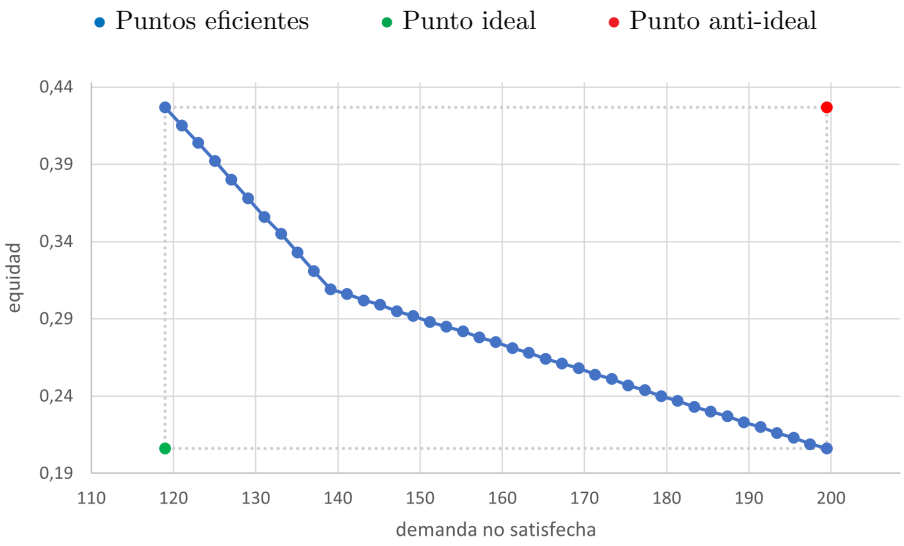

Figure 4: Frontera de Pareto *"Demanda no satisfecha-equidad"*

"Minimizar demanda no satisfecha y restringir coste"

$$\text{MIN: } \sum_p (d(p) - \sum_r C(p,r))$$
$$\text{Sujeto a: } \sum_r (k \cdot X(r) + \sum_p C(p,r) \cdot eurton(r)) \leq \varepsilon_k$$

Conjunto de puntos eficientes tomando $h = \frac{1}{40}$

$$\begin{array}{c} \\ Obj1 \\ Obj2 \end{array} \begin{array}{c} \begin{array}{cccccc} Punto1 & Punto2 & & Punto39 & Punto40 \end{array} \\ \begin{pmatrix} 119.000 & 124.053 & \cdots & 378.505 & 387.600 \\ 38880.000 & 38425.250 & \cdots & 21144.750 & 20690.000 \end{pmatrix} \end{array}$$

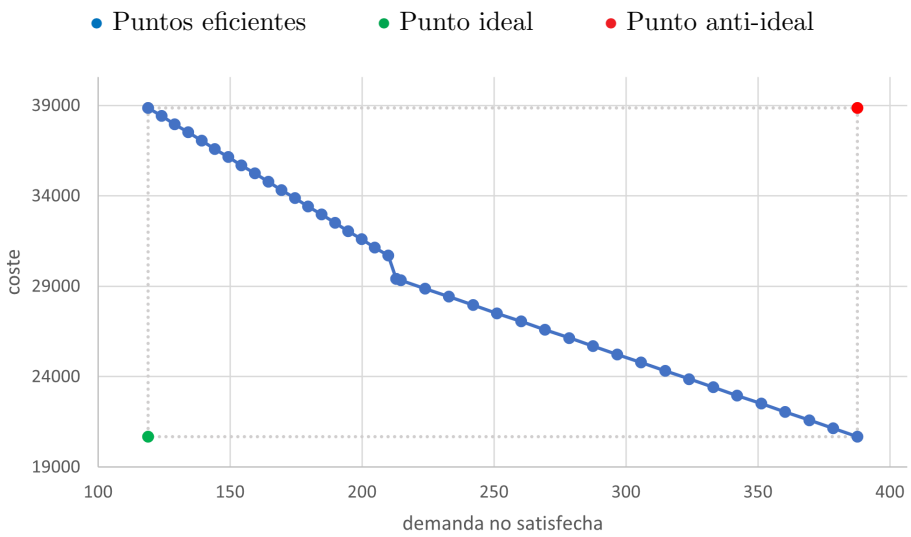

Figure 5: Frontera de Pareto *"Demanda no satisfecha-coste"*

"Minimizar coste y restringir demanda no satisfecha"

$$\text{MIN:} \quad \sum_r (k \cdot X(r) + \sum_p C(p,r) \cdot eurton(r))$$

$$\text{Sujeto a:} \quad \sum_p (d(p) - \sum_r C(p,r)) \leq \varepsilon_k$$

Conjunto de puntos eficientes tomando $h = \frac{1}{40}$

$$\begin{array}{c} \\ Obj1 \\ Obj2 \end{array} \begin{array}{c} \begin{array}{cccccc} Punto1 & Punto2 & & Punto39 & Punto40 \end{array} \\ \begin{pmatrix} 387.600 & 380.885 & \cdots & 125.715 & 119.000 \\ 20690.000 & 21025.750 & \cdots & 38275.650 & 38880.000 \end{pmatrix} \end{array}$$

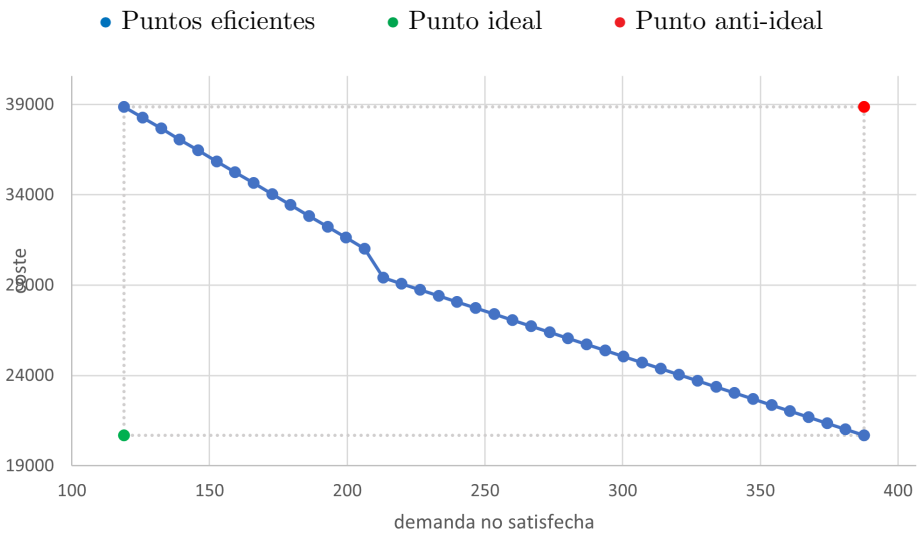

Figure 6: Frontera de Pareto *"Demanda no satisfecha-coste"*

Comparando ambos métodos, vemos como, al utilizar el método de la ε-restricciones, obtenemos muchos más puntos eficientes que conforman la frontera de Pareto.

Se puede observar que, tanto minimizando un objetivo y restringiendo el otro como al revés, obtenemos fronteras de Pareto muy similares. Además, estás son parecidas a la halladas anteriormente mediante el método de las ponderaciones, con la diferencia del número de puntos eficientes.

7 Programación Compromiso

La programación compromiso consiste en cuál de los puntos eficientes de la frontera de Pareto coger como solución óptima. El criterio que se sigue en este método es coger el punto más próximo al punto ideal.

De esta manera, definiendo z_i^* como el valor ideal del obj_i y z_{*i} como el valor anti-ideal del obj_i, tenemos que nuestra problema se reduce en minimizar la distancia al punto ideal, es decir

$$\textbf{MIN: } L = \sum_i \frac{z_i^* - obj_i}{z_i^* - z_{*i}}$$

En nuestro problema, mediante la matriz de pagos, obtenemos los siguientes z_i^*, z_{*i}.

$$\begin{array}{c} \\ z^* \\ z_* \end{array} \begin{array}{c} \begin{array}{cccc} Obj_1 & Obj_2 & Obj_3 & Obj_4 \end{array} \\ \begin{pmatrix} 119.000 & 20690.000 & 0.206 & 788.000 \\ 387.600 & 40000.000 & 1.000 & 2374.000 \end{pmatrix} \end{array}$$

Ahora, resolviendo con programación compromiso, hemos obtenido los siguientes resultados.

Resultados

En primer lugar veamos que valores toman nuestras variables objetivo.

L	OBJ_1	OBJ_2	OBJ_3	OBJ_4
2.17	381.00	22760.00	1.00	920.00

En este caso, obtenemos buenos resultados tanto en el obj_2 como en el obj_4, aunque sacrificamos el obj_1 y el obj_3, que toman, prácticamente, sus valores máximos. Todo dependerá de la importancia que le dé el decisor a cada objetivo para saber si esta es una buena solución o no.

Veamos ahora que centros se construyen a y donde proporcionan ayuda

	Maradi	Tanout	Agadez	Zinder	Dem. No Sat.
Se construye?	NO	NO	SI	SI	-
In-Gall	-	-	156.00	-	-
Aderbissinat	-	-	-	-	81.00 (100%)
Tessaoua	-	-	-	129.00	-
Dakoro	-	-	-	-	213.00 (100%)
Tatokou	-	-	-	-	39.00 (100%)
Bakatchiraba	-	-	-	-	30.00 (100%)
Mayahi	-	-	-	273.00	-
Koundoumaoua	-	-	-	30.00	-
Sabon Kafi	-	-	-	-	18.00 (100%)
Total	-	-	156.00	432.00	381.00 (39.32%)

Table 1: Tabla de resultados de la Programación Compromiso

8 Programación por metas

A la hora de resolver problemas de verdad, es muy difícil encontrar una solución que optimice todos nuestros objetivos. De ahí surge que el decisor ponga unos límites o metas para que, si no se consigue alcanzar el valor óptimo, por lo menos se cumplan las metas propuestas.

De esta manera, cuando vayamos a resolver un problema, deberemos meter la restricción $obj_i + p_i - n_i = meta_i$, donde n_i y p_i son las desviaciones negativas y positivas de la meta fijada por el decisor. Además, nuestro objetivo pasará a ser minimizar la suma de las desviaciones no deseadas.

Para resolver nuestro problema utilizaremos como metas valores que sean un 10% peores que el valor óptimo de cada objetivo por separado. Además para este método, solo tendremos en cuenta los tres primero objetivos.

Meta $Obj_1 \rightarrow 145.86$
Meta $Obj_2 \rightarrow 22621.00$
Meta $Obj_3 \rightarrow 0.285$

Table 2: Metas que utilizaremos en todos los problemas

Dentro de la programación por metas, existen diferentes procedimientos. Nosotros utilizaremos programación por metas ponderadas y por metas lexicográficas.

Programación por metas ponderadas

La programación por metas ponderadas sigue la idea de minimizar las desviaciones no deseadas de las metas fijadas. A esto se le añade la inclusión de pesos en la función a optimizar en función de la importancia que tenga cada objetivo para el decisor. Veamos un par de ejemplos.

• **Mismos pesos para todos los objetivos**

Vamos a resolver el problema tomando $pes_i = 1 \quad \forall i$, en resumen:

$$\textbf{MIN: } SUMADESV = \sum_i \frac{n_i}{meta_i}$$

$$\begin{aligned} \textbf{Sujeto a: } & obj_1 + p_1 - n_1 = meta_1 \\ & obj_2 + p_2 - n_2 = meta_2 \\ & obj_3 + p_3 - n_3 = meta_3 \end{aligned}$$

Resolviendo el problema obtenemos los siguientes resultados

SUMADESV	OBJ_1	OBJ_2	OBJ_3
0.812	145.86	40000.00	0.298

	Maradi	Tanout	Agadez	Zinder	Dem. No Sat.
Se construye?	SI	NO	SI	SI	-
In-Gall	-	-	109.57	-	46.43 (29.76%)
Aderbissinat	-	-	56.89	-	24.11 (29.76%)
Tessaoua	-	-	-	129.00	-
Dakoro	149.61	-	-	-	63.39 (29.76%)
Tatokou	-	-	-	27.39	11.61 (29.76%)
Bakatchiraba	-	-	-	29.67	0.33 (1.10%)
Mayahi	-	-	-	273.00	-
Koundoumaoua	-	-	-	30.00	-
Sabon Kafi	-	-	-	18.00	-
Total	149.61	-	166.47	507.07	145.86 (15.05%)

Table 3: Tabla de resultados de la Programación por metas ponderadas(1)

• **Distintos pesos para cada objetivo**

Supongamos que para nosotros, lo más importante en este problema, es que nos gastemos el menor dinero posible, y si podemos, cumplir el resto de objetivos. Podríamos tomas pesos $pes_1 = 0.1$, $pes_2 = 0.8$ y $pes_3 = 0.1$.

$$\textbf{MIN: } SUMADESV = \sum_i pes_i \cdot \frac{n_i}{meta_i}$$

$$\begin{aligned} \textbf{Sujeto a: } & obj_1 + p_1 - n_1 = meta_1 \\ & obj_2 + p_2 - n_2 = meta_2 \\ & obj_3 + p_3 - n_3 = meta_3 \end{aligned}$$

Resolviendo el problema obtenemos los siguientes resultados

SUMADESV	OBJ_1	OBJ_2	OBJ_3
0.377	387.60	22871.44	0.861

	Maradi	Tanout	Agadez	Zinder	Dem. No Sat.
Se construye?	SI	NO	SI	SI	-
In-Gall	-	-	21.63	-	134.37 (86.13%)
Aderbissinat	-	-	11.23	-	69.77 (86.13%)
Tessaoua	-	-	-	129.00	-
Dakoro	29.54	-	-	-	183.46 (86.13%)
Tatokou	-	-	-	39	-
Bakatchiraba	-	-	-	30.00	-
Mayahi	-	-	-	273.00	-
Koundoumaoua	-	-	-	30.00	-
Sabon Kafi	-	-	-	18.00	-
Total	29.59	-	32.86	519.00	387.6 (40%)

Table 4: Tabla de resultados de la Programación por metas ponderadas(2)

Veamos ahora que pasa si lo que queremos conseguir es la mayor equidad posible en el reparto de ayuda. Tomando $pes_1 = 0.2$, $pes_2 = 0.1$ y $pes_3 = 0.7$ obtenemos los siguientes resultados.

SUMADESV	OBJ_1	OBJ_2	OBJ_3
0.087	153.08	40000.00	0.285

	Maradi	Tanout	Agadez	Zinder	Dem. No Sat.
Se construye?	SI	NO	SI	SI	-
In-Gall	-	-	111.50	-	44.50 (28.53%)
Aderbissinat	-	-	57.89	-	23.11 (28.53%)
Tessaoua	-	-	-	129.00	-
Dakoro	152.24	-	-	-	60.76 (28.53%)
Tatokou	-	-	-	27.88	11.12 (28.51%)
Bakatchiraba	-	-	-	21.44	8.56 (28.53%)
Mayahi	-	-	-	273.00	-
Koundoumaoua	-	-	-	30.00	-
Sabon Kafi	-	-	-	12.97	5.03 (27.94%)
Total	152.24	-	169.39	494.29	153.08 (15.79%)

Table 5: Tabla de resultados de la Programación por metas ponderadas(3)

Vemos como las soluciones varían mucho en función de los pesos que les demos a los objetivos. Una vez sepamos la importancia que le dé el decisor a cada uno de ellos, sabremos decidir que solución es la mejor.

Programación por metas lexicográficas

En la programación por metas lexicográficas debemos establecer unos niveles de prioridad en los que se encuentren las metas. Este método consiste en resolver los objetivos que se encuentren en el primer nivel. Una vez resuelto, fijar ese valor y resolver el siguiente nivel, y así sucesivamente hasta llegar al último nivel con todos los objetivos previos fijados.

En nuestro problema hemos decidido diferenciar tres niveles. El primero, donde se encuentra el objetivo de la demanda no satisfecha, dado que al tratarse de un proyecto para cubrir las necesidades de las personas de Níger tras una crisis alimentaria, lo más lógico es dar la máxima ayuda que podamos. En segundo lugar, queremos que el reparto sea lo más equitativo posible, de esta manera, llegará a un mayor número de personas. Por último, teniendo en cuenta que existe un presupuesto fijado del que no nos pasaremos, creemos que lo menos importante es el coste del proyecto.

Una vez establecidos los niveles de prioridad nuestro problema se resume en

$$\textbf{MIN:}\ [obj_1(n_1), obj_3(n_3), obj_2(n_2)]$$

en ese orden.

Minimizando n_1 obtenemos la siguiente solución

SUMADESV	OBJ_1	OBJ_2	OBJ_3
0	145.86	40000.00	1.00

A continuación, fijamos el valor de obj_1 introduciendo en el siguiente problema la restricción $obj_1 = 145.86$. Minimizando obj_3 con la restricción añadida la solución pasa a ser

SUMADESV	OBJ_1	OBJ_2	OBJ_3
0.043	145.86	40000.00	0.298

Podemos ver una gran mejora en el obj_3. Veamos si es posible mejorar la solución aun más. Minimazando obj_2 teniendo en cuenta las restricciones $obj_1 = 145.86$ y $obj_3 = 0.298$ obtenemos

SUMADESV	OBJ_1	OBJ_2	OBJ_3
0.7683	145.86	40000.00	0.298

Y por otro lado,

	Maradi	Tanout	Agadez	Zinder	Dem. No Sat.
Se construye?	SI	NO	SI	SI	-
In-Gall	-	-	109.57	-	46.43 (29.76%)
Aderbissinat	-	-	56.89	-	24.11 (29.76%)
Tessaoua	-	-	-	129.00	-
Dakoro	149.61	-	-	-	63.39 (29.76%)
Tatokou	-	-	-	39.00	-
Bakatchiraba	-	-	-	30.00	-
Mayahi	-	-	-	273.00	-
Koundoumaoua	-	-	-	21.07	8.93 (29.76%)
Sabon Kafi	-	-	-	14.99	3.01 (16.72%)
Total	149.61	-	166.47	507.07	145.86 (15.05%)

Table 6: Tabla de resultados de la Programación por metas lexicográficas

Esta sería nuestra solución óptima obtenida mediante programación por metas lexicográficas. Vemos como, los objetivos que se encontraban en un nivel de prioridad mayor, se acercan mucho a su valor óptimo, mientras que los otros se adaptan a estos.

9 Conclusiones

Este problema tiene distintas maneras de resolverse mediante muchos criterios. Sabremos la bondad de cada solución cuando sepamos que prioridad tienen unos objetivos sobre otros.

Sin embargo, analizando las soluciones obtenidas mediante los distintos métodos, podemos decir que, si nuestra prioridad es minimizar el coste, entonces tendremos que sacrificar tanto el obj_1 como el obj_3, y que si queremos minimizar la demanda no satisfecha, seremos capaces de conseguir equidad en el reparto de ayuda, pero no de abaratar los costes. Lo mismo pasa cuando nuestro principal objetivo es conseguir la mayor equidad posible, conseguiremos obtener un buen valor en el obj_1, pero no en el obj_2.

En nuestra opinión, por la clase de problema que estamos trantando, creemos que la mejor solución que podemos escoger es tanto la obtenidada mediante programción por metas ponderadas (tomando todos los pesos iguales), como la obtenida mediante programación por metas lexicográficas, pues ambas consiguen minimizar la demanda no satisfecha mientras consiguen un reparto equitativo de la ayuda.

Anexo: Códigos utilizados para la resolución de los problemas

```
$title NIGER G17 y G18
*-----------------------------------------------------------------------

SETS
r centros de reparto /M,Tt,A,Z/
p poblaciones / In,Ad,Te,Da,Ta,Ba,Ma,Ko,Sa/
o funciones objetivo /f1*f4/
alias(o,o2);

PARAMETERS
d(p) demanda de p /In 156, Ad 81, Te 129,Da 213,
Ta 39, Ba 30, Ma 273, Ko 30, Sa 18/
eurton(r) precio por tonelada en r /M 90, Tt 70, A 50, Z 30/
k precio fijo por costruccion de r /1000/
maxpresup presupuesto maximo del que se dispone /40000/
tontot toneladas totales disponibles /850/
*Para sacar la matriz de pagos (EXTRA)
fpeso(o) se quiere hacer que todas valgan 0 menos 1
func_fijada el valor que obligamos a la funcion que tenga peso 1;

TABLE dist(p,r) distancia de pueblitos p a centros r
M Tt A Z
In 625 396 119 541
Ad 454 147 161 286
Te 122 191 492 114
Da 124 244 562 300
Ta 355 43 258 188
Ba 320 8 306 152
Ma 95 252 553 156
Ko 183 197 490 71
Sa 269 43 336 102;

*(EXTRA)
TABLE mpagos(o,o) matriz de pagos del modelo;
*matriz que obliga a C a ser 0 si dist ¿ 200
TABLE distmax(p,r) creamos todos los posibles escenarios que podemos usar;
distmax(p,r)$(dist(p,r)<=200) = 1;

VARIABLES
X(r) construccion (1) o no (0) del centro de reparto r
C(p,r) cantidad que da r a p
SI_C(p,r) si r da a p (1) o no (0)
*minimizamos el maximo (cuidado si hay mucha desviacion)
MAXPROP maxima proporcion de demanda no satisfecha
*VARIABLES PARA SIMPLIFICAR SUMATORIOS Y SACAR DATOS
TON(r) toneladas a repartir en r
```

NOSAT(p) toneladas no satifechas en p
OBJ1 valor objetivo de cada funcion objetivo 1
OBJ2 valor objetivo de cada funcion objetivo 2
OBJ3 valor objetivo de cada funcion objetivo 3
OBJ4 valor objetivo de cada funcion objetivo 4;

POSITIVE VARIABLES C,MAXPORC,TON,NOSAT;
BINARY VARIABLES X,SI_C;

EQUATIONS
func1 func. obj. modelo1 min demanda no satisfecha
func2 func. obj. modelo2 min coste total de operacion
func3 func. obj. modelo3 no satiafechas equivalentes (min MAXPROP)
func4 func. obj. modelo4 min distancia recorrida
maxton no superar las toneladas recibidas
presupuesto no superar el maximo presupuesto
maxconst construccion maxima de centros de reparto
demanda(p) la demanda insatisfecha tiene que ser positiva
*obligamos a las binarias
cantidad(p,r) si dist>200 C=0 si X=0 C=0
obligSI_C1(p,r) si C>0 SI_C=1
minC satisfacer 60% demanda
proporc(p) obtener la maxima proporcion de demanda no satisfecha
*Hay problemas con la equidad en la frontera de pareto
func_pesos Funcion con pesos para obligar a las funciones a un
valor
*Para sacar la informacion
centros(r) cantidad que recibe cada centro de reparto
dinsat demanda insatisfecha en cada poblado;

func1 .. OBJ1 =E= sum(p, d(p)-sum(r,C(p,r)));
func2 .. OBJ2 =E= sum(r, k*X(r) + sum(p, C(p,r)*eurton(r)));
func3 .. OBJ3 =E= MAXPROP;
func4 .. OBJ4 =E= 2*sum((p,r), dist(p,r)*SI_C(p,r));
maxton .. sum((p,r), C(p,r)) =L= tontot;
presupuesto .. sum(r, 1000*X(r) + sum(p, C(p,r)*eurton(r))) =L= maxpresup;
maxconst .. sum(r, X(r)) =L= 3;
demanda(p) .. sum(r, C(p,r)) =L= d(p);
cantidad(p,r) .. C(p,r) =L= distmax(p,r)*d(p)*X(r);
obligSI_C1(p,r) .. C(p,r) =L= d(p)*SI_C(p,r);
minC .. sum((p,r), C(p,r)) =G= 0.6*sum(p,d(p));
proporc(p) .. (d(p)-sum(r,C(p,r)))/d(p) =L= MAXPROP;
*Para la matriz de pagos
func_pesos .. OBJ1*fpeso('f1')+OBJ2*fpeso('f2')+OBJ3*fpeso('f3')
+OBJ4*fpeso('f4') =E= func_fijada;
*Informativos
centros(r) .. TON(r) =E= sum(p,C(p,r));
dinsat(p) .. NOSAT(p) =E= d(p) - sum(r,C(p,r));

*OPCIONES PARA LA RESOLUCION DE LOS MODELOS
OPTION OPTCR = 0;

MODEL modelo /func1, func2, func3, func4, maxton, presupuesto, maxconst, demanda, cantidad, obligSI_C1, minC, proporc,centros,dinsat/;

*Creamos la diagonal de la matriz de pagos, que es la solución optima
solve modelo **using MIP minimazing** OBJ1;
mpagos('f1','f1') = OBJ1.l;

solve modelo **using MIP minimazing** OBJ2;
mpagos('f2','f2') = OBJ2.l;

solve modelo **using MIP minimazing** OBJ3;
mpagos('f3','f3') = OBJ3.l;

solve modelo **using MIP minimazing** OBJ4;
mpagos('f4','f4') = OBJ4.l;

*abrimos fichero de escritura
file resultados_NIGER /ruta donde quieras crear txt/resultados_NIGER.txt/;
put resultados_NIGER 'RESULTADOS PLANIFICACION DE EMERGENCIA NIGER'////;
put "FUNCIONES OBJETIVO:"//"f1 - minimiza demanda no satisfecha"/
"f2 - minimiza el coste total de la operacion"/
"f3 - consigue equidad en demandas no satifechas"/
"f4 - minimiza distancia recorrida"////"MATRIZ DE PAGOS:"//
@13 "f1" @25 "f2" @37 "f3" @49 "f4"/;

*Modelo con la función objetivo con pesos
MODEL modelo_pagos /func1, func2, func3, func4, maxton, presupuesto, maxconst, demanda, cantidad, obligSI_C1, minC, proporc, func_pesos, centros, dinsat/;

*Para funcion objetivo 1
loop(o$(ord(o)<>1),
*cambio del peso de la funcion que fijamos
fpeso(o) = 1;
func_fijada = mpagos(o,o);
solve modelo_pagos **using MIP minimazing** OBJ1;

*actualizar matriz de pagos
mpagos(o,'f1') = OBJ1.l;
*El peso vuelve a valer 0
fpeso(o) = 0;
);

```
*Para funcion objetivo 2
loop(o$(ord(o)<>2),
*cambio del peso de la funcion que fijamos
fpeso(o) = 1;
func_fijada = mpagos(o,o);
solve modelo_pagos using MIP minimazing OBJ2;

*actualizar matriz de pagos
mpagos(o,'f2') = OBJ2.l;
*El peso vuelve a valer 0
fpeso(o) = 0;
);

*Para funcion objetivo 3
loop(o$(ord(o)<>3),
*cambio del peso de la funcion que fijamos
fpeso(o) = 1;
func_fijada = mpagos(o,o);
solve modelo_pagos using MIP minimazing OBJ3;

*actualizar matriz de pagos
mpagos(o,'f3') = OBJ3.l;
*El peso vuelve a valer 0
fpeso(o) = 0;
);

*Para funcion objetivo 4
loop(o$(ord(o)<>4),
*cambio del peso de la funcion que fijamos
fpeso(o) = 1;
func_fijada = mpagos(o,o);
solve modelo_pagos using MIP minimazing OBJ4;

*actualizar matriz de pagos
mpagos(o,'f4') = OBJ4.l;
*El peso vuelve a valer 0
fpeso(o) = 0;
);

display mpagos;

*escritura matriz de pagos
loop(o, put o.tl @3;
    loop(o2, put mpagos(o,o2);); put /;
);
```

*_____-
*CREACION FRONTERA DE PARETO POR PONDERACIONES (G18)

SETS
h intervalo en que dividimos max y min /1*31/
j indice de la frontera de pareto/1*2/;

SCALAR
p1 proporcion de la funcion objetivo 1
p2 proporcion de la funcion objetivo 2;

*Matrices donde guardaremos los pares de cada frontera de pareto

TABLE res_par1(h,j);
TABLE res_par2(h,j);

VARIABLES
PAR1 objetivo de la funcion objetivo para la frontera de pareto 1
PAR2 objetivo de la funcion objetivo para la frontera de pareto 2;

EQUATIONS
par_dem_eq función objetivo 'demanda no satifecha-equidad'
par_dem_cos función objetivo 'demanda no satifecha-coste';

*Funciones objetivos con los pesos correspondientes

par_dem_eq.. PAR1 =E= (p1/(387.6-119))*OBJ1+(p2/(1-0.206))*OBJ3;
par_dem_cos.. PAR2 =E= (p1/(387.6-119))*OBJ1+(p2/(40000-20690))*OBJ2;

*Modelo para hallar la frontera de Pareto 'demanda no satifecha-equidad'

MODEL FRONT_PAR1 /par_dem_eq,func1,func3,maxton,presupuesto,maxconst,
demanda,cantidad,obligSI_C1,minC,proporc,centros,dinsat/;

*Modelo para hallar la frontera de Pareto 'demanda no satifecha-coste'

MODEL FRONT_PAR2 /par_dem_cos,func1,func2,maxton,presupuesto,maxconst,
demanda,cantidad,obligSI_C1,minC,proporc,centros,dinsat/;

*escritura frontera de pareto por ponderados
put ///"FRONTERA DE PARETO POR PONDERACIONES:"//
"dem. no satif. contra equidad", @45 "dem. no satif. contra coste"/ @11 "f1"
@23 "f3" @55 "f1" @67 "f2"/;

loop(h,
*Damos valores a las ponderaciones de la función objetivo
p1 = (ord(h)-1)/card(h);
p2 = 1-p1;
*Resolvemos ambos modelos y guardamos las soluciones
*en las matrices correspondientes
solve FRONT_PAR1 **using MIP minimizing** PAR1;
res_par1(h,'1') = OBJ1.l;
res_par1(h,'2') = OBJ3.l;
solve FRONT_PAR2 **using MIP minimizing** PAR2;
res_par2(h,'1') = OBJ1.l;
res_par2(h,'2') = OBJ2.l;
*escritura
put res_par1(h,'1') res_par1(h,'2') @37 "–" @45 res_par2(h,'1')
@57 res_par2(h,'2')/;);

display res_par1,res_par2;

*_____-
*CREACION FRONTERA DE PARETO POR E-RESTRICCIONES (G17)

SETS
i indice para guardar datos /1,2/
e intervalo en que dividimos max y min /1*41/;

PARAMETERS
epsilon valor de epsilon que variara con las e-restricciones
min minimo de la funcion con epsilon
max maximo de la funcion con epsilon
*creacion de matriz para guardar datos
minf1_f3(i,e) guardar min f1 y f3<E
minf3_f1(i,e) guardar min f3 y f1<E
minf1_f2(i,e) guardar min f1 y f2<E
minf2_f1(i,e) guardar min f2 y f1<E;

EQUATIONS
epsilon1 epsilon para f1
epsilon2 epsilon para f2
epsilon3 epsilon para f3;

epsilon1 .. OBJ1 =L= epsilon;
epsilon2 .. OBJ2 =L= epsilon;
epsilon3 .. OBJ3 =L= epsilon;

*creacion de un modelo para cada nueva E-restriccion
MODEL modelo_E1 /func1,func2,func3,func4,maxton, presupuesto, maxconst,
demanda, cantidad, obligSI_C1, minC, proporc,epsilon1,centros,dinsat/;

MODEL modelo_E2 /func1,func2,func3,func4,maxton, presupuesto, maxconst, demanda, cantidad, obligSI_C1, minC, proporc,epsilon2/;

MODEL modelo_E3 /func1,func2,func3,func4,maxton, presupuesto, maxconst, demanda, cantidad, obligSI_C1, minC, proporc,epsilon3,centros,dinsat/;

*minimizar f1, f3<=E

max = mpagos('f1','f3');
min = mpagos('f3','f3');
loop(e,
epsilon = max - ((max-min)*(ord(e)-1))/40;
solve modelo_E3 **using MIP minimazing** OBJ1;
minf1_f3('1',e) = OBJ1.l;
minf1_f3('2',e) = OBJ3.l;
);
display minf1_f3;

*minimizar f3, f1<=E
max = mpagos('f3','f1');
min = mpagos('f1','f1');
loop(e,
epsilon = max - ((max-min)*(ord(e)-1))/40;
solve modelo_E1 **using MIP minimazing** OBJ3;
minf3_f1('1',e) = OBJ3.l;
minf3_f1('2',e) = OBJ1.l;
);
display minf3_f1;

*minimizar f1, f2<=E
max = mpagos('f1','f2');
min = mpagos('f2','f2');
loop(e,
epsilon = max - ((max-min)*(ord(e)-1))/40;
solve modelo_E2 **using MIP minimazing** OBJ1;
minf1_f2('1',e) = OBJ1.l;
minf1_f2('2',e) = OBJ2.l;
);
display minf1_f2;
*minimizar f2, f1<=E
max = mpagos('f2','f1');
min = mpagos('f1','f1');
loop(e,
epsilon = max - ((max-min)*(ord(e)-1))/40;
solve modelo_E1 **using MIP minimazing** OBJ2;
minf2_f1('1',e) = OBJ2.l;
minf2_f1('2',e) = OBJ1.l;
);
display minf2_f1;

*escritura para frontera de pareto por E-restricciones
put ///"FRONTERA DE PARETO POR E-RESTRICCIONES:"//
@16 "dem. no satif. contra equidad" @77 "dem. no satif. contra coste"/
@7 "min f1" @19 "f3 < E " @36 "min f3" @48 "f1 < E" @68 "min f1" @80
"f2 < E" @97 "min f2" @109 "f1 ¡< E"/;
loop(e, put minf1_f3('1',e),minf1_f3('2',e) @29 "-" minf3_f1('1',e),
minf3_f1('2',e) @60 "–", minf1_f2('1',e),minf1_f2('2',e) @90 "-",
minf2_f1('1',e),minf2_f1('2',e)/;);

*_____-
*PROGRAMACION COMPROMISO (G17)

PARAMETERS
maxf1 maximo en f1 con todas las funciones objetivo
maxf2 maximo en f2 con todas las funciones objetivo
maxf3 maximo en f3 con todas las funciones objetivo
maxf4 maximo en f4 con todas las funciones objetivo
minf1 minimo en f1 con todas las funciones objetivo
minf2 minimo en f2 con todas las funciones objetivo
minf3 minimo en f3 con todas las funciones objetivo
minf4 minimo en f4 con todas las funciones objetivo;

*Los minimos y maximos ya se han calculado al hacer la matriz de pagos
minf1 = mpagos('f1','f1');
minf2 = mpagos('f2','f2');
minf3 = mpagos('f3','f3');
minf4 = mpagos('f4','f4');
maxf1 = mpagos('f2','f1');
maxf2 = mpagos('f3','f2');
maxf3 = mpagos('f2','f3');
maxf4 = mpagos('f3','f4');

display maxf1,maxf2,maxf3,maxf4,minf1,minf2,minf3,minf4,mpagos;

VARIABLES
COMP variable en la que se guarda el compromiso;

EQUATIONS
compromiso funcion objetivo para programacion compromiso;

*Ojo el punto es el de min, entonces minimizar la distancia a minf

compromiso.. COMP =E= ((OBJ1-minf1)/(maxf1-minf1)) +
((OBJ2-minf2)/(maxf2-minf2))+((OBJ3-minf3)/(maxf3-minf3))+
((OBJ4-minf4)/(maxf4-minf4));

MODEL modelo_comp /func1,func2,func3,func4,maxton,presupuesto,maxconst,
demanda,cantidad,obligSI_C1,minC,proporc,compromiso,centros,dinsat/;

solve modelo_comp **using MIP minimazing** COMP;

display COMP.l, OBJ1.l, OBJ2.l, OBJ3.l, OBJ4.l, X.l , C.l, TON.l, NOSAT.l;

*escritura resultados progamacion compromiso
put ///"PROGRAMACION COMPROMISO:"//"COMP:" COMP.l/"OBJ1:" OBJ1.l/"OBJ2:" OBJ2.l/"OBJ3" @6 OBJ3.l/"OBJ4:"OBJ4.l;
put // "Centros reparto:" @28 "M" @39 "Tt" @52 "A" @64 "Z"/ "Construidos (X):";
loop(r, put X.l(r);); put /"A repartir (TON):" @17; loop(r, put TON.l(r););
put // "Poblaciones:" @34 "In" @46 "Ad" @58 "Te" @70 "Da" @82 "Ta" @94 "Ba" @106 "Ma" @118 "Ko" @130 "Sa"/"No satisfechas (NOSAT):";
loop(p, put NOSAT.l(p););
put //"Cantidad a repartir de r a p:"//@23 "In" @35 "Ad" @47 "Te" @59 "Da" @71 "Ta" @83 "Ba" @95 "Ma" @107 "Ko" @119 "Sa"/;
loop(r, put r.tl; loop(p, put C.l(p,r);); put/;);

*_____-
*PROGRAMACIÓN POR METAS (G17 y G18)

SETS
z diferentes configuraciones de pesos /1,2,3/;

PARAMETERS
metf(o) meta que ponemos para fo
pesof(o) peso para las funciones;

VARIABLES
Pf1 desviacion positiva de f1
Nf1 desviacion negativa de f1
Pf2 desviacion positiva de f2
Nf2 desviacion negativa de f2
Pf3 desviacion positiva de f3
Nf3 desviacion negativa de f3
SUMADESV suma de todas las desviaciones que vamos a minimizar;

POSITIVE VARIABLES Pf1,Pf2,Pf3,Nf1,Nf2,Nf3;

EQUATIONS
funcmeta funcion obejtivo para minimizar la desviaciones de las metas
meta1 restriccion para saatisfacer la meta de la func 1
meta2 restriccion para saatisfacer la meta de la func 2
meta3 restriccion para saatisfacer la meta de la func 3;

*Debemos normalizar los valores, por eso se divide entre la meta
*Como todas las funciones son minimizar, entonces f1 +P - N ¡= M
*Entonces hay que minimizar la N que es lo que no hemos llegado

funcmeta .. SUMADESV =E= pesof('f1')*Nf1/metf('f1') + pesof('f2')*Nf2/metf('f2')+ pesof('f3')*Nf3/metf('f3');

```
meta1 .. OBJ1 + Pf1 - Nf1 =E= metf('f1');
meta2 .. OBJ2 + Pf2 - Nf2 =E= metf('f2');
meta3 .. OBJ3 + Pf3 - Nf3 =E= metf('f3');

MODEL modelo_metas /func1,func2,func3,maxton,presupuesto,maxconst,
demanda,cantidad,obligSI_C1,minC,proporc,funcmeta,meta1,meta2,meta3,
centros,dinsat/;

*Para las metas, utilizamos valores que sean un 10% peores que
*el valor optimo de cada objetivo por separado

metf('f1') = minf1 + (maxf1-minf1)/10;
metf('f2') = minf2 + (maxf2-minf2)/10;
metf('f3') = minf3 + (maxf3-minf3)/10;

display metf;

loop(z,
*Todos los objetivos con el mismo peso
pesof(o)$(ord(z)=1)=1;
*Más importante minimizar coste
pesof('f1')$(ord(z)=2) = 0.1;
pesof('f2')$(ord(z)=2) = 0.8;
pesof('f3')$(ord(z)=2) = 0.1;
*Más importante buscar la equidad en el reparto
pesof('f1')$(ord(z)=3) = 0.2;
pesof('f2')$(ord(z)=3) = 0.1;
pesof('f3')$(ord(z)=3) = 0.7;
*Resolvemos
solve modelo_metas using MIP minimazing SUMADESV;
*Escritura resultados progamacion por metas
put ///"PROGRAMACION POR METAS PONDERADAS:"/
"(mismas metas que antes)"//
"pesof1:" pesof('f1')/"pesof2:" pesof('f2')/ "pesof3:" pesof('f3')/
"OBJ1:" @8 OBJ1.l/"OBJ2:" @8 OBJ2.l/"OBJ3" @8 OBJ3.l;
put // "Centros reparto:" @28 "M" @39 "Tt" @52 "A" @64 "Z"/
"Construidos (X):";
loop(r, put X.l(r);); put /"A repartir (TON):" @17; loop(r, put TON.l(r););
put // "Poblaciones:" @34 "In" @46 "Ad" @58 "Te" @70 "Da" @82 "Ta" @94
"Ba" @106 "Ma" @118 "Ko" @130 "Sa"/"No satisfechas (NOSAT):";
loop(p, put NOSAT.l(p););
put //"Cantidad a repartir de r a p:"//@23 "In" @35 "Ad" @47 "Te" @59 "Da"
@71 "Ta" @83 "Ba" @95 "Ma" @107 "Ko" @119 "Sa"/;
loop(r, put r.tl; loop(p, put C.l(p,r);); put/;);
);
```

```
*_____-
*PROGRAMACIÓN POR METAS LEXICOGRÁFICAS (G18)

*Antes de empezar debemos ver en que nivel de prioridad
*se encuentra cada una de nuestras metas
*Para nosotros la meta 1 es nuestra mayor prioridad, despues
*la meta 3, y por último, la meta 2.

PARAMETERS
val_fijo1
val_fijo2;

*1º nivel de prioridad
pesof('f1') = 1;
pesof('f2') = 0;
pesof('f3') = 0;

MODEL LEX1 /func1,func2,func3,maxton,presupuesto,maxconst, demanda,
cantidad,obligSI_C1,minC,proporc,funcmeta, meta1,meta2,meta3,centros,dinsat/;
solve LEX1 using MIP minimazing SUMADESV;
*Guardamos el resultado para fijarlo en el siguiente nivel val_fijo1=OBJ1.l;
display val_fijo1;

*2º nivel de prioridad
pesof('f1') = 0;
pesof('f2') = 0;
pesof('f3') = 1;

EQUATIONS
desv1_fija fijamos el valor de la desviacion del nivel 1;

desv1_fija.. OBJ1 =E= val_fijo1;

MODEL LEX2 /func1,func2,func3,maxton,presupuesto,maxconst, demanda,
cantidad,obligSI_C1,minC,proporc,funcmeta, meta1,meta2,meta3,desv1_fija,
centros,dinsat/;
solve LEX2 using MIP minimazing SUMADESV;
*Guardamos el resultado para fijarlo en el siguiente nivel val_fijo2=OBJ3.l;

*3º nivel de prioridad
pesof('f1') = 0;
pesof('f2') = 1;
pesof('f3') = 0;

EQUATIONS
desv3_fija fijamos el valor de la desviacion del nivel 2;

desv3_fija.. OBJ3 =E= val_fijo2;
```

MODEL LEX3 /func1,func2,func3,maxton,presupuesto,maxconst, demanda, cantidad,obligSI_C1,minC,proporc,funcmeta, meta1,meta2,meta3,desv1_fija, desv3_fija,centros,dinsat/;
solve LEX3 **using MIP minimazing** SUMADESV;

*escritura resultados progamacion por metas lexicográficas
put ///"PROGRAMACION POR METAS LEXICOGRAFICAS:"/
"(misma metas, priorizando func1 y despues func3)"//
"OBJ1:" OBJ1.l/ "OBJ2:" OBJ2.l/ "OBJ3" OBJ3.l;
put // "Centros reparto:" @28 "M" @39 "Tt" @52 "A" @64 "Z"/
"Construidos (X):";
loop(r, put X.l(r);); put /"A repartir (TON):" @17; loop(r, put TON.l(r););
put // "Poblaciones:" @34 "In" @46 "Ad" @58 "Te" @70 "Da" @82 "Ta" @94 "Ba" @106 "Ma" @118 "Ko" @130 "Sa"/"No satisfechas (NOSAT):";
loop(p, put NOSAT.l(p););
put //"Cantidad a repartir de r a p:"//@23 "In" @35 "Ad" @47 "Te" @59 "Da" @71 "Ta" @83 "Ba" @95 "Ma" @107 "Ko" @119 "Sa"/;
loop(r, put r.tl; loop(p, put C.l(p,r);); put/;);