Universidad de Valladolid

Facultad de Ciencias

Trabajo Fin de Grado

Grado en Matemáticas

Relajación y heurística Lagrangiana. Aplicación a un problema de optimización combinatoria.

Autora: Elena Fernández Serna

Tutores:

Jesús Sáez Aguado

Pedro César Álvarez Esteban

A todas las personas que han hecho posible que haya llegado hasta aquí,
en especial a mi tutor Jesús Sáez Aguado, que me ha ayudado
a escribir el punto final de esta etapa.

Aguilar de Campoo, Junio 2021.

Resumen

La **optimización combinatoria** es un campo de las matemáticas en el que se trabaja con un problema modelizado de modo que el objetivo que se tiene es minimizar o maximizar una función. Las variables de ese problema están sujetas a una serie de restricciones que nos complican su resolución, entre las que se encuentran las restricciones de variables enteras.

Realizaremos una pequeña introducción a los problemas de optimización relajados, cuyo nombre proviene de ser más "permisivos" con las restricciones, profundizando concretamente en la denominada **relajación Lagrangiana**. Gracias a las características de la función Lagrangiana, es posible emplear el **algoritmo subgradiente** para buscar un óptimo de esta función iteración tras iteración. Detallaremos los pasos y parámetros necesarios para este método, describiendo detalladamente una de las reglas más famosas, la de Held-Wolfe-Crowder.

Se presentarán técnicas heurísticas con las que podremos obtener soluciones para la primera iteración del algoritmo, además de su posible mejora con una determinada frecuencia. La técnica de combinar la relajación y la **heurística Lagrangiana** nos permitirá obtener simultáneamente buenas cotas inferiores y superiores del óptimo del problema original.

Por último, se ha procedido a aplicar todo lo recogido en la memoria para la resolución de un problema de optimización combinatoria en particular, el **problema de cubrimiento máximo**. Para ello se ha empleado el programa informático de *Xpress IVE* y se presentan los resultados y conclusiones obtenidas del análisis de un conjunto de datos.

Palabras clave: Optimización, Programación entera, Relajación Lagrangiana, Heurística Lagrangiana, Algoritmo subgradiente, Cubrimiento máximo

Índice general

Resumen III

0. Introducción **1**

0.1. Conceptos básicos 2

0.2. Problema relajado 4

0.3. Propiedades del problema relajado 4

0.4. Tipos de relajación 6

1. Relajación Langrangiana **9**

1.1. Subproblema Lagrangiano P(λ) 10

1.1.1. Relajación Lagrangiana 10

1.1.2. Propiedad de acotación 11

1.2. Problema dual Lagrangiano (DL) 12

1.2.1. Ejemplo. Empresa mayorista de patatas. 13

1.3. Propiedades de la función dual 16

1.3.1. Concavidad de L(λ) 16

1.3.2. Subgradiente de una función en un punto 18

1.3.3. Conjunto de soluciones óptimas de (DL) 19

1.4. Caso (PE) de maximización 20

1.4.1. Estructura del problema 20

1.4.2. Ejemplo problema de maximización. Fábrica de galletas 21

1.4.3. Restricciones de igualdad 24

1.5. Teorema de convexificación y dualización 25

1.5.1. Teorema de convexificación y dualización 27

1.5.2. Consecuencias del teorema 29

2. Algoritmo subgradiente **31**

2.1. Optimización no diferenciable 32

2.2. Método subgradiente 33

2.2.1. Algoritmos ascendentes 33

2.2.2. Lógica del algoritmo subgradiente 34

2.2.3. Parámetros generales del algoritmo 35

2.2.4. Propiedad de los algoritmos subgradientes 37

2.2.5. Diagrama de flujo 39

2.3. Tamaño de paso del algoritmo 40

2.3.1. Condiciones de convergencia 40

2.3.2. Algunas reglas para el tamaño de paso 42

2.4. Métodos heurísticos 45

2.4.1. Método Greedy 46

2.5. Regla de Held-Wolfe-Crowder 49

2.5.1. Pseudocódigo del algoritmo subgradiente 50

3. Problema de cubrimiento máximo **53**

3.1. Problemas de localización 54

3.1.1. Problema de cubrimiento total (*set covering*) 55

3.2. Problema de cubrimiento máximo 59

3.2.1. Ejemplo de problema de cubrimiento máximo 60

3.2.2. Algoritmo Greedy para problemas de cubrimiento máximo . . . 63

3.2.3. Relajación Lagrangiana para el problema de cubrimiento máximo 64

3.2.4. Algoritmo subgradiente para el problema de cubrimiento máximo 66

3.3. Resolución de problemas de cubrimiento máximo 69

3.4. Conclusiones 76

A. Algunos resultados de análisis convexo **79**

Bibliografía **83**

Capítulo 0

Introducción

Comenzaremos esta memoria haciendo una introducción a la teoría de la relajación en problemas de optimización, una técnica empleada particularmente en el campo de la optimización entera, donde las variables de decisión deben tomar valores enteros. Veremos varios tipos de relajación para centrarnos posteriormente en el capítulo 1 en la denominada relajación Lagrangiana. Este proceso consiste en eliminar las restricciones "difíciles" incluyéndolas en la función objetivo de tal forma que quede definido un problema dual asociado al problema original.

Estudiaremos las propiedades de esta función dual, tratando teoremas como el de convexificación y dualización. Además, daremos la definición de subgradiente de la función en un punto, lo que nos permitirá en el capítulo 2 presentar el algoritmo subgradiente. Con este método seremos capaces de obtener buenas cotas inferiores o superiores del óptimo, dependiendo si nos encontramos ante un problema de minimización o de maximización. Cuanto menor sea la diferencia entre la cota superior y la inferior, mejor será nuestra aproximación al óptimo.

En el último capítulo se pretende ilustrar todo lo visto anteriormente específicamente para el problema de cubrimiento máximo, en el cual nuestro objetivo a *grosso modo* es maximizar la demanda cubierta sujetos a un número prefijado de servicios. Se combinará la relajación con la heurística Lagrangiana para obtener unos resultados numéricos a partir del uso de lenguajes de programación.

0.1. Conceptos básicos

Hoy en día nos encontramos muchas ocasiones en las que necesitamos tomar una decisión que nos permita maximizar unas ganancias o minimizar el esfuerzo requerido para una tarea. En matemáticas nos encargamos de modelizar estas situaciones y convertirlas en problemas de optimización.

En estos problemas, el propósito es maximizar o minimizar una función objetivo Z a raíz de dar valores a las variables de decisión $\{X_1, X_2, ... , X_n\}$, las cuales están sujetas a una serie de restricciones funcionales, de no negatividad y de variables enteras.

El conjunto de soluciones que cumplen todas las restricciones constituye una región factible, en donde buscaremos la solución que haga tener a Z su valor máximo o mínimo, dependiendo del problema.

Se ha seguido la notación del libro *Integer Programming* de Wolsey [18], en el cual se puede consultar ampliamente los temas tratados en esta breve introducción.

Definición 0.1. Se define problema de programación lineal entera **(PE)** como aquel problema de optimización de una función lineal, con restricciones lineales y en el que al menos una de sus variables de decisión X_i está restringida a tomar un valor entero.

Un problema de **programación lineal entera** en el que nuestro objetivo sea minimizar, puede modelizarse de la siguiente manera:

(PE)

$$\begin{gathered}
\text{Min } Z = c_1X_1 + c_2X_2 + ... + c_nX_n \\
\text{sujeto a:} \\
a_{11}X_1 + a_{12}X_2 + ... + a_{1n}X_n \leq b_1 \\
a_{21}X_1 + a_{22}X_2 + ... + a_{2n}X_n \leq b_2 \\
\vdots \\
a_{m1}X_1 + a_{m2}X_2 + ... + a_{mn}X_n \leq b_m
\end{gathered}$$

$$X_i \geq 0, \qquad X_i \in \mathbb{N} \tag{1}$$

Definición 0.2. La **región factible** de un problema **(PE)** es el conjunto de las soluciones que verifican todas las restricciones del problema. Lo denotamos por S.

Definición 0.3. El **valor óptimo** del problema, el cual lo denotamos por Z_{pe}, es el que además de cumplir todas las restricciones, alcanza el valor más favorable de la función objetivo.

La modelización anterior (1) no es una estructura cerrada, pues el problema a tratar puede tener como objetivo maximizar Z y alguna restricción puede aparecer con los signos de igualdad o desigualdad mayor o igual, veremos en otra sección como tratar estos casos.
En algunos problemas de gran escala se presentan funciones objetivo complejas y puede dificultarse su resolución, dando lugar a problemas en los que el costo computacional o el tiempo de resolución sea muy grande. De ser así, nuestro objetivo es aproximarnos al valor óptimo mediante cotas en un tiempo razonable.
Denotaremos una cota inferior de la solución óptima como **LB** (del inglés *lower bound*) y **UB** (del inglés *upper bound*) a una cota superior.

$$LB \leq Z_{pe} \leq UB$$

Definición 0.4. Llamamos **hueco absoluto** al valor de la diferencia $(UB - LB)$. Siendo UB la cota superior y LB la cota inferior del óptimo.

El problema a tratar puede tener unidades muy pequeñas o muy grandes y con el valor del hueco absoluto podríamos no hacernos una idea del error que se puede llegar a cometer. Debido a esto, se suele trabajar con **hueco relativo**:

$$\text{hueco relativo } (\%) = \frac{UB-LB}{UB} 100$$

0.2. Problema relajado

Partimos de un problema de programación lineal entera **(PE)**, si eliminásemos algunas de las restricciones, el problema resultante sería más fácil de resolver. Este problema a priori más sencillo, va a ser lo que conozcamos como problema relajado.

Un problema modelizado como en (1) podemos escribirlo de forma equivalente como:
(PE)

$$\text{Minimizar } cx$$
$$\text{sujeto a } x \in S$$

Una relajación, o **problema relajado** del problema **(PE)**, es un problema de optimización de la siguiente forma:
(PR)

$$\text{Minimizar } Z_R(x)$$
$$\text{sujeto a } x \in S_R$$

que cumple dos condiciones:

1. $S \subset S_R$ es decir, el conjunto factible del problema relajado contiene al conjunto factible del problema original.

2. $Z_R(x) \leq cx \qquad \forall x \in S$. La función objetivo del problema relajado es menor o igual que la del problema original para todo $x \in S$.

0.3. Propiedades del problema relajado

Una vez que resolvemos el problema relajado **(PR)**, cuyo óptimo denotamos por Z_{pr}, podemos sacar información útil sobre el problema original **(PE)** y su valor óptimo Z_{pe}.

Definición 0.5. Un problema es **no factible**, si la intersección de los conjuntos en los que se cumplen todas las restricciones es el conjunto vacío.

EJEMPLO. Vemos en [Figura 1] un ejemplo sencillo en dos variables de un problema no factible, pues las regiones donde se cumplen las condiciones tienen intersección vacía.

$$\min X + Y$$
$$\text{sujeto a}$$
$$X + Y \geq 6$$
$$X + Y \leq 4$$
$$X, Y \geq 0$$

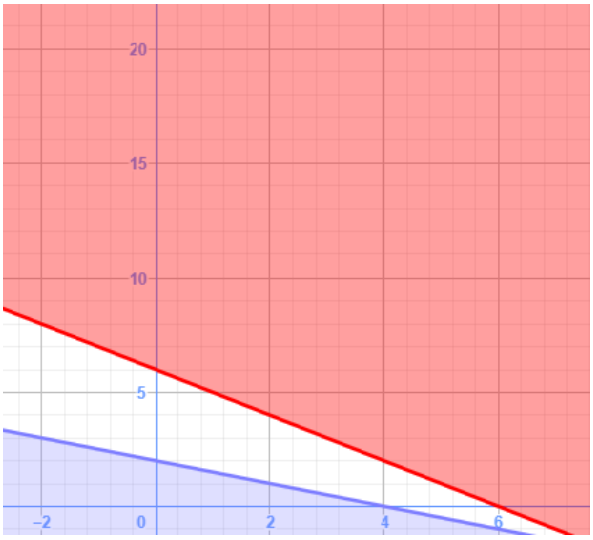


Figura 1: Ejemplo de problema no factible

Proposición 0.1. Si el problema relajado **(PR)** no es factible, el problema original **(PE)** tampoco lo será.

Demostración. La demostración es sencilla, pues teniendo en cuenta la primera condición que debe de cumplir un problema relajado, $S \subset S_R$, si $S_R = \emptyset$, entonces $S = \emptyset$. □

Proposición 0.2. Si la solución óptima del problema relajado (Z_{pr}) es factible para el problema original, es decir, si se ajusta a todas las restricciones, entonces será la solución óptima del problema original $Z_{pe} = Z_{pr}$.

Demostración. Tomamos $X_R \in S_R$, como $S \subset S_R$, el problema "grande"será el problema relajado, por tanto si X_R es el óptimo del problema "grande"lo será también del "pequeño" y $Z_{pe} = Z_{pr}$. □

0.4. Tipos de relajación

A continuación vamos a detallar los tipos de relajación más comunes en programación entera.

Relajación de restricciones

Cuando tenemos la igualdad $Z_R(x) = cx$ para todo $x \in S$, se denomina relajación de restricciones. Al problema relajado sólo le hacemos cumplir la primera condición, $S \subset S_R$, pues en la segunda tenemos la igualdad para todo $x \in S$.
Se obtiene eliminando alguna restricción del problema original y tiene la forma general:
(PR)

$$\text{Minimizar } cx$$
$$\text{sujeto a } x \in S_R$$

Relajación lineal simple

Es un tipo de relajación de restricciones que consiste en eliminar las restricciones de valores enteros. Si en un principio $X_i \in \{0, ..., N\}$, tras la relajación X_i pasará a tomar cualquier valor del intervalo $0 \leq X_i \leq N$.
En el caso de programación entera pura, los valores de X_i son 0 o 1, es decir, o se encuentra presente la variable i (valor 1) o no se encuentra (valor 0). Si lo relajamos linealmente, X_i podrá tomar valores en el intervalo $0 \leq X_i \leq 1$.
Una relajación lineal simple tiene forma:

(PL)

$$\text{Minimizar } \sum_{j=1}^{n} c_j x_j$$
$$\text{sujeto a}$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad i = 1, ..., m$$
$$0 \leq x_j \leq N \qquad j = 1, ..., n$$

Programas como *Xpress-Mosel* me permiten resolver de una manera eficiente relajaciones lineales mediante comandos ya incorporados. Es más costoso encontrar una solución que tenga que tomar valores enteros que una dentro de un intervalo.

Relajación lineal fuerte

Consiste en añadir a la relajación lineal simple restricciones de planos de corte. Necesitamos que el conjunto de puntos que constituyen las soluciones factibles sea un conjunto acotado, pues el algoritmo de planos de corte se mueve por esta envoltura convexa convergiendo en la solución óptima. Este procedimiento puede ser consultado en el capítulo 9 de las notas de Krumke [12].
Los paquetes modernos que incluyen programas como *Cplex*, *Xpress* o *Lindo* requieren mucho esfuerzo computacional para realizar relajaciones fuertes, además el resultado al que se llega no es tan bueno si lo miramos de forma proporcional respecto a la cantidad de esfuerzo empleado.

Relajación Lagrangiana

Es otro tipo muy utilizado en programación entera, ya no es una relajación de restricciones. Dedicaremos el capítulo 1 a profundizar sobre ella.

Capítulo 1

Relajación Langrangiana

En este capítulo introduciremos las bases de la relajación Lagrangiana, técnica muy empleada en problemas de programación entera para encontrar cotas del óptimo.
A partir de ahora, **trataremos siempre con un problema de minimización**, a no ser que se especifique lo contrario. Dedicaremos una sección entera dentro del capítulo al caso de maximización.
Como detallaremos a continuación, el procedimiento de la relajación Lagrangiana consiste en introducir las restricciones difíciles del problema en la función objetivo mediante unos multiplicadores de Lagrange y definir así el problema dual. Las ventajas que esto nos ofrece es disponer de una función cóncava, en la que nos será más sencillo encontrar el óptimo del dual. Veremos algunos resultados más teóricos como el teorema de convexificación y dualización con su correspondiente demostración.
La técnica de la relajación Lagrangiana en problemas de minimización nos proporcionará una buena cota inferior de Z_{pe}.
Para este capítulo, las principales referencias que se han seguido han sido los libros de Shapiro [14] y Wolsey [18], además de Hiriart-Urruty [10] para los resultados de análisis convexo.

1.1. Subproblema Lagrangiano P(λ)

Vamos a suponer un problema de optimización entera que lo podamos modelizar de la siguiente forma:

$$\textbf{(PE)} \qquad \text{Minimizar } f(x) = \sum cx$$

$$\text{sujeto a}$$

$$Ax \leq b$$

$$Dx \leq d$$

$$x \;\; entero, \;\; x \geq 0 \tag{1.1}$$

Donde la función objetivo $f(x)$ que voy a minimizar es continua y el vector x es el vector de variables, que tendrá dimensión n. Es decir, $x = (x_1, x_2, ..., x_n) \in \mathbb{R}^n$.
Las restricciones de la forma $Ax - b \leq 0$, donde $A \in \mathbb{M}_{mxn}$ y $b \in \mathbb{R}^m$, son restricciones complicantes o difíciles. Denotaremos por $g(x) = Ax - b \leq 0 \in \mathbb{R}^m$.
El conjunto $X = \{Dx \leq d\}$, con $x \geq 0$ y x entero, es el conjunto de soluciones factibles de $Dx \leq d$. La optimización sobre X tiene una forma especial y es un problema más sencillo de resolver, por lo que si eliminásemos las restricciones complicantes nos quedaría un problema a priori "fácil".
El inconveniente con el que nos encontramos al relajar los problemas de esta manera es que perdemos una cantidad importante de información, para tratar de evitar esto surge la relajación Lagrangiana, cuyo fundamento es incluir las restricciones complicantes $g(x) \leq 0$ en la función objetivo.

1.1.1. Relajación Lagrangiana

Definición 1.1. Una **relajación Lagrangiana** de un problema **(PE)** consiste en la introducción de las restricciones complicantes en la función objetivo mediante los multiplicadores de Lagrange $\lambda_i \geq 0$ dando lugar a un subproblema Lagrangiano $\mathbf{P(\lambda)}$.

Sea $\lambda \in \mathbb{R}^m$ el vector de multiplicadores. El subproblema Lagrangiano de (1.1) es:

$$\mathbf{P}(\lambda) \qquad \begin{array}{c} \text{Minimizar } f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) \\ \text{sujeto a} \end{array}$$

$$x \in X \subset \mathbb{R}^n \tag{1.2}$$

Definición 1.2. Dado un problema de programación entera **(PE)** y su subproblema Lagrangiano $\mathbf{P}(\lambda)$, llamamos **Lagrangiano** de $\mathbf{P}(\lambda)$ a $L(x, \lambda) = f(x) + \lambda g(x)$

Definición 1.3. Dado un problema de programación entera **(PE)** y su subproblema Lagrangiano $\mathbf{P}(\lambda)$, se conoce como **función Lagrangiana** a la función objetivo del subproblema Lagrangiano, dada por $L(\lambda) = min_{x \in X}\{f(x) + \lambda g(x)\}$ con $x \in X$ y $\lambda \geq 0$

NOTA: Hemos definido con abuso de notación a la función Lagrangiana como $L(\lambda)$ después de definir al Lagrangiano como $L(x, \lambda)$ para no dificultar la comprensión con múltiples notaciones.

1.1.2. Propiedad de acotación

Proposición 1.1. Para cualquier $\lambda \geq 0$, se tiene que el valor la función Lagrangiana evaluada en λ es menor o igual que el valor óptimo del problema original. $L(\lambda) \leq Z_{pe}$.

Demostración. Si el problema **(PE)** no tiene soluciones factibles, tomamos por convenio que $Z_{pe} = \infty$, luego es trivial que $L(\lambda) \leq Z_{pe}$ para cualquier λ.
Si **(PE)** sí tiene soluciones factibles, tomamos una de estas soluciones, $\hat{x} \in X$. Por ser solución factible, cumple las restricciones complicantes, por tanto $g(\hat{x}) \leq 0$ y $\lambda \geq 0$. Luego tenemos que la función Lagrangiana:

$$L(\lambda) = min_{x \in X}\{f(x) + \lambda g(x)\} \leq f(\hat{x}) + \lambda g(\hat{x}) \leq f(\hat{x})$$

Esta desigualdad es válida para toda solución factible del problema **(PE)**, en concreto para el óptimo, $\hat{\hat{x}} \in X$, y concluimos que $L(\lambda) \leq f(\hat{\hat{x}}) = Z_{pe}$ para cualquier λ. □

Corolario 1.1. El valor $L(\lambda)$ es una cota inferior de la solución óptima del problema original Z_{pe}.

1.2. Problema dual Lagrangiano (DL)

En el subproblema Lagrangiano, se penaliza el no cumplimiento de las restricciones complicantes llevando estas a la función Lagrangiana mediante los coeficientes de Lagrange $\lambda_i \geq 0$.
Acabamos de ver por la proposición 1.1 que $L(\lambda)$ es una cota inferior de Z_{pe}. Lo que nos interesa es encontrar la mejor de esas cotas, y aquí es donde nace el problema dual, pues para encontrar la mejor de las cotas tendremos que resolver un problema de maximización.

Definición 1.4. Dado un problema de programación entera **(PE)** y su subproblema Lagrangiano **P(λ)**, definimos el **problema dual Lagrangiano** como el problema de maximización que permite encontrar la mejor cota inferior (LB) de la solución óptima del problema original Z_{pe}.

El problema dual Lagrangiano de (1.2) viene dado por:

$$\begin{aligned} \textbf{(DL)} \qquad & \text{Maximizar } L(\lambda) \\ & \text{sujeto a} \\ & \lambda \geq 0 \end{aligned} \tag{1.3}$$

Denotamos por Z_d al valor optimo del problema (1.3).

Definición 1.5. Llamamos **hueco de dualidad** a la diferencia dada por los valores $Z_{pe} - Z_d$. Cuanto mejor encontremos la cota, menor será el hueco de dualidad.

Una vez que hemos encontrado una cota inferior mediante relajación Lagrangiana, se procede a la búsqueda de la cota superior (UB) con técnicas heurísticas, lo que trataremos en capítulos posteriores.
La combinación de estos dos métodos se conoce como "Heurística Lagrangiana".

1.2.1. Ejemplo. Empresa mayorista de patatas.

Para ilustrar lo dicho hasta ahora, vamos a proceder a realizar un ejemplo con un problema de minimización sencillo para 3 variables de decisión.
Supongamos que una empresa mayorista de patatas quiere minimizar sus gastos. Ellos compran las patatas a un agricultor y las venden en un supermercado. Esta empresa puede escoger comprar o no comprar 3 tipos de patatas, por tanto las variables de decisión tomarán el valor $x_i = 0$ si el tipo de patata i no se va a comprar y $x_i = 1$ si el tipo de patata i sí se va a comprar.
El precio de compra que pone el agricultor por cada tipo de patata es el siguiente:
Tipo 1 = 3€ Tipo 2= 9€ Tipo 3= 14€
Las condiciones que pone la empresa es que obtengan al menos 9€por la reventa. Como no es muy buen negociante, el precio de venta al supermercado es:
Tipo 1 = 1€ Tipo 2= 4€ Tipo 3= 7€
El problema inicial sería:
(PE)

$$\text{minimizar } Z = 3x_1 + 9x_2 + 14x_3$$
$$\text{sujeto a}$$
$$x_1 + 4x_2 + 7x_3 \geq 9$$
$$x_i \in \{0, 1\}$$

Introducimos las restricciones complicantes en la función objetivo mediante el multiplicador de Lagrange $\lambda \geq 0$. El subproblema lagrangiano viene dado por:
P(λ)

$$\text{minimizar } 3x_1 + 9x_2 + 14x_3 + \lambda(-x_1 - 4x_2 - 7x_3 + 9)$$
$$\text{sujeto a}$$
$$x_i \in \{0, 1\}$$

La función Lagrangiana del problema es:

$$\mathbf{L}(\lambda) = 9\lambda + min\{(3 - \lambda)x_1 + (9 - 4\lambda)x_2 + (14 - 7\lambda)x_3\}$$

Y el problema dual por tanto, será de la forma:

(DL)

$$\text{maximizar } \mathbf{L}(\lambda) = 9\lambda + min\{(3-\lambda)x_1 + (9-4\lambda)x_2 + (14-7\lambda)x_3\}$$
$$\text{sujeto a}$$
$$\lambda \geq 0$$

Para encontrar el máximo en **(DL)**, impondremos $x_i = 0$ en los tramos de λ donde el coeficiente de x_i es positivo y $x_i = 1$ en los tramos donde el coeficiente x_i es negativo.

Intervalo	x_1	x_2	x_3
$0 \leq \lambda \leq 2$	0	0	0
$2 \leq \lambda \leq 2.25$	0	0	1
$2.25 \leq \lambda \leq 3$	0	1	1
$3 \leq \lambda$	1	1	1

Con los datos de la tabla construímos $L(\lambda)$ lineal a trozos en cada intervalo:

$$L(\lambda) = \begin{cases} 9\lambda & si \quad 0 \leq \lambda \leq 2 \\ 2\lambda + 14 & si \quad 2 \leq \lambda \leq 2.25 \\ -2\lambda + 23 & si \quad 2.25 \leq \lambda \leq 3 \\ -3\lambda + 26 & si \quad 3 \leq \lambda \end{cases}$$

Representamos la función $L(\lambda)$ en el plano X-Y, véase [Figura 1.1]. Cada intervalo definido a trozos lo graficaremos de un color.

Como podemos observar, el máximo de $L(\lambda)$ se alcanza en el punto (2.25,18.5), lo que podíamos deducir pues la función es creciente hasta λ =2.25 y decrece a partir de este punto.

Para λ=2.25, $L(\lambda) = 18.5$ y se presentan dos soluciones óptimas de $P(\lambda)$, correspondientes a los valores:

$$\textbf{S1}.\ x_1, x_2 = 0,\ \ x_3 = 1$$
$$\textbf{S2}.\ x_1 = 0,\ \ x_2, x_3 = 1.$$

S	x_1	x_2	x_3	$L(\lambda)$	$P(\lambda)$	(PE)
S1	0	0	1	18.5	18.5	14
S2	0	1	1	18.5	18.5	23

S1 presenta una solución menor para el problema original **(PE)**, sin embargo, no cumple la restricción complicante del problema **(PE)**, pues $x_1 + 4x_2 + 7x_3 = 7 \ngeq 9$. El agricultor minizaría sus gastos, pero no llegaría a los ingresos establecidos.
Por tanto, **S2** es la única factible. El vendedor de patatas deberá comprar al agricultor tipo 2 y tipo 3 para minimizar sus gastos y conseguir su objetivo de recaudar 9€.

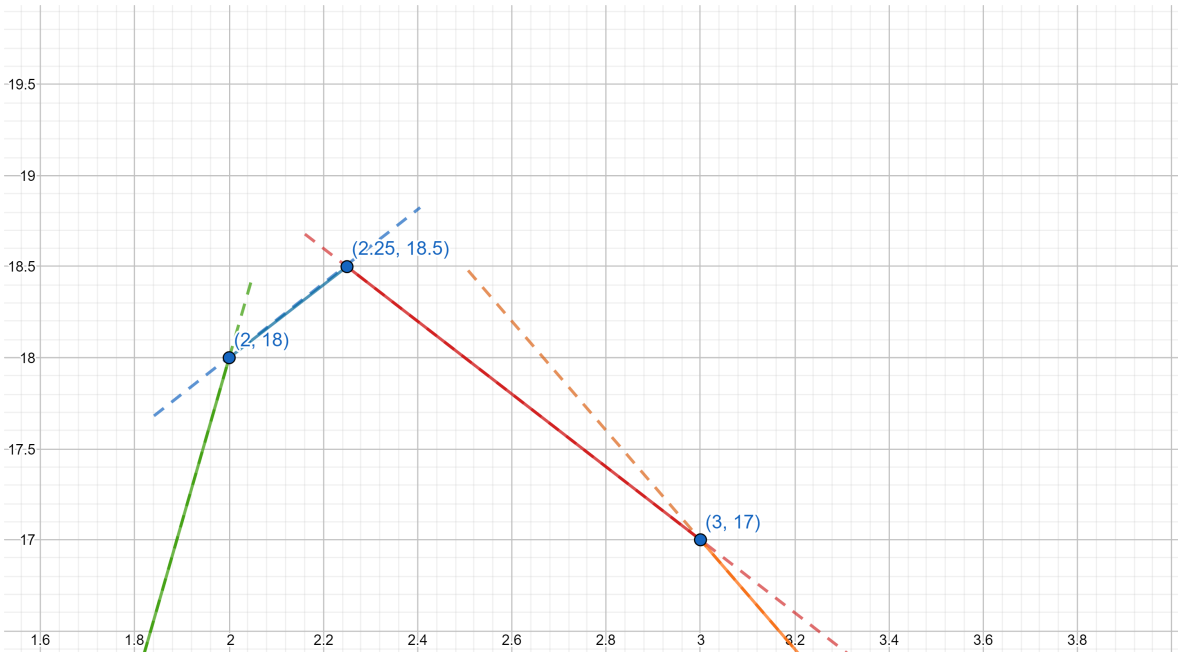


Figura 1.1: Solución gráfica del problema dual al ejemplo de las patatas

1.3. Propiedades de la función dual

El problema dual Lagrangiano **(DL)** tiene características importantes que implican que su solución sea factible. Acabamos de ver en [Figura 1.1] la forma gráfica que presentaría $L(\lambda)$ para un problema primal de minimización.
$L(\lambda)$ es una función cóncava. Además, es una función lineal a trozos, por tanto, no es diferenciable en los vértices, donde se presentan la soluciones óptimas del problema.
No obstante, es subdiferenciable en todo el dominio, pues existe subgradiente en todo punto, a continuación veremos lo que esto significa.
En esta sección se emplearán algunos resultados de análisis convexo que pueden ser consultados en el libro de Hiriat-Urruty [10] y en el apéndice A.

1.3.1. Concavidad de L(λ)

Definición 1.6. Una función $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ es **cóncava** si para todo par de vectores $x, y \in \mathbb{R}^n$ y para todo $\alpha \in \mathbb{R}$ con $0 \leq \alpha \leq 1$ se verifica que:

$$f(\alpha x + (1-\alpha)y) \geq \alpha f(x) + (1-\alpha)f(y)$$

En una función cóncava, si tomamos cualquier cuerda que una dos puntos de esta, la gráfica va a estar siempre por encima de esta cuerda. [Figura 1.2]

Definición 1.7. Una función $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ es **convexa** si para todo par de vectores $x, y \in \mathbb{R}^n$ y para todo $\alpha \in \mathbb{R}$ con $0 \leq \alpha \leq 1$ se verifica que:

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$$

En una función convexa, si tomamos cualquier cuerda que una dos puntos de esta, la gráfica va a estar siempre por debajo de esta cuerda. [Figura 1.3]

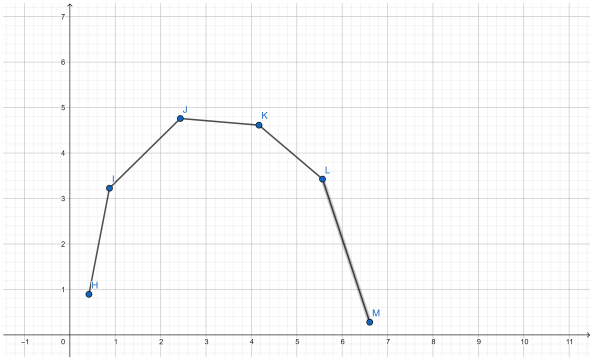


Figura 1.2: Función cóncava

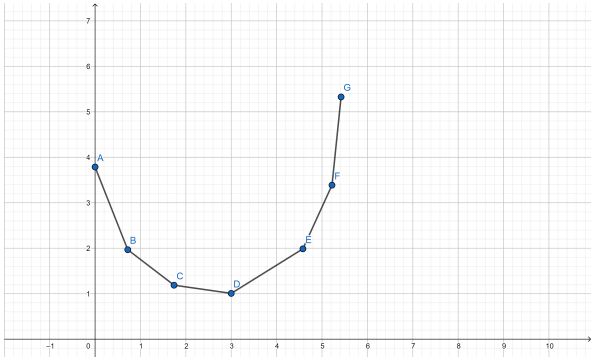


Figura 1.3: Función convexa

Proposición 1.2. La función Lagrangiana $L(\lambda)$ para el caso de minimización siempre es finita y cóncava.

Demostración. $L(\lambda)$ es finita en $\mathbb{R}^n$ por hipótesis de que $f(x)$ y $g(x)$ son funciones continuas en un espacio X compacto.
Veamos que $L(\lambda)$ es cóncava. Para ello, dado un α arbitrario con $0 \leq \alpha \leq 1$, tomamos dos puntos λ_1 y λ_2 y veamos que satisface la condición de concavidad. Por definición de función Lagrangiana, existirá un x_0 que cumpla:

$$L(\alpha\lambda_1 + (1-\alpha)\lambda_2) = f(x_0) + \{\alpha\lambda_1 + (1-\alpha)\lambda_2\}g(x_0)$$

Además, para ese x_0 se tiene que:

$$(1.)\ L(\lambda_1) \leq f(x_0) + \lambda_1 g(x_0)$$
$$(2.)\ L(\lambda_2) \leq f(x_0) + \lambda_2 g(x_0)$$

Multiplicando (1.) por α y (2.) por $(1-\alpha)$ y sumándolas se sigue:

$$\alpha L(\lambda_1) + (1-\alpha)L(\lambda_2) \leq f(x_0) + \{\alpha\lambda_1 + (1-\alpha)\lambda_2\}g(x_0) = L(\alpha\lambda_1 + (1-\alpha)\lambda_2)$$

Que es la condición de concavidad que habíamos dado en la definición 1.6. □

Corolario 1.2. Para problemas de maximización $L(\lambda)$ siempre es finita y convexa.

1.3.2. Subgradiente de una función en un punto

Definición 1.8. Subgradiente de una función. Sea una función $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ e $\hat{y} \in \mathbb{R}^n$. Un vector $\gamma \in \mathbb{R}^n$ se dice que es un subgradiente de f en el punto $\hat{y}$ si cumple que:

$$f(y) \leq f(\hat{y}) + \gamma^T(y - \hat{y})$$

para todo $y \in \mathbb{R}^n$ con $y \neq \hat{y}$.

El conjunto de todos los subgradientes de la función $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ en el punto $\hat{y}$ se representa por $\partial f(\hat{y})$ y se conoce como el subdiferencial.

En el problema dual tratábamos un problema de maximización en donde queríamos encontrar el óptimo de $L(\lambda)$ en todo $\mathbb{R}^n$. Si la función $L(\lambda)$ alcanza el óptimo en λ_0, entonces $\mathbf{0} \in \partial L(\lambda_0)$.

Proposición 1.3. Si $\hat{x} \in X$ es solución del subproblema Lagrangiano P(λ) para un λ^*, entonces el vector $\gamma^T = g(\hat{x})$ es un subgradiente de la función Lagrangiana en λ^*.

Demostración. Como $\hat{x}$ es el óptimo de la función Lagrangiana en λ^*, se tiene que:

$$L(\lambda^*) = f(\hat{x}) + \lambda^* g(\hat{x}) = min_{x \in X}\{f(x) + \lambda^* g(x)\}$$

Para cualquier λ, tenemos por definición de $L(\lambda)$ y tomando $\gamma^T = g(\hat{x})$:

$$L(\lambda) = min_{x \in X}\{f(x) + \lambda g(x)\} \leq f(\hat{x}) + \lambda g(\hat{x}) = L(\lambda^*) - \lambda^* g(\hat{x}) + \lambda g(\hat{x})$$

$$L(\lambda) \leq L(\lambda^*) + \gamma^T(\lambda - \lambda^*)$$

□

Este resultado indica que con la resolución del subproblema Lagrangiano se tiene ya un subgradiente sin necesidad de cálculos adicionales.
El algoritmo subgradiente, método para la búsqueda del óptimo en la función Lagran-

giana, el cual trataremos a fondo en el siguiente capítulo, se aprovecha del resultado de esta proposición.

1.3.3. Conjunto de soluciones óptimas de (DL)

En el apéndice A, dedicado a ampliar algunos aspectos sobre funciones y conjuntos convexos, se prueba que una función cóncava tiene subgradiente en todo punto. Como $L(\lambda)$ es una función cóncava, podemos considerar γ un subgradiente de L en λ_0.

Lema 1.1. Sea γ un subgradiente de L en λ_0. Entonces, el conjunto $\{\lambda \,/\gamma^T(\lambda - \lambda_0) \geq 0\}$ contiene todas las soluciones óptimas del problema dual DL.

Demostración. Como γ es subgradiente de L en λ_0, tenemos que por definición, para todo $\lambda \in \mathbb{R}^n$ se cumple:

$$\gamma^T(\lambda - \lambda_0) \geq L(\lambda) - L(\lambda_0)$$

Si λ es óptimo de (DL), entonces se tiene que $L(\lambda) - L(\lambda_0) \geq 0$, obteniendo el resultado deseado. □

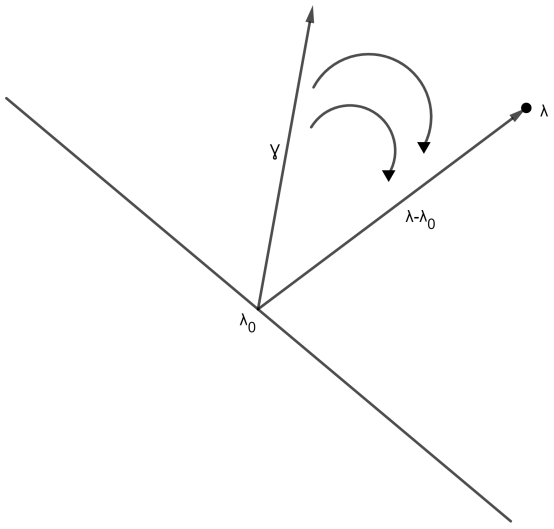


Figura 1.4: γ subgradiente de L en λ_0

1.4. Caso (PE) de maximización

1.4.1. Estructura del problema

Hasta ahora hemos generalizado el caso para un problema de optimización en el que originalmente teníamos que minimizar. Esta no tiene por qué ser una estructura cerrada, pues podemos encontrarnos ante un problema de maximización:

(PE) Maximizar $f(x)$

sujeto a

$$g_i(x) \leq 0 \quad i \in \{1, ..., m\}$$

$$x \in X \subset \mathbb{R}^n \tag{1.4}$$

Cuyo subproblema Lagrangiano queda de la forma:

P(λ) Maximizar $f(x) - \sum_{i=1}^{m} \lambda_i g_i(x)$

sujeto a

$$x \in X \subset \mathbb{R}^n \tag{1.5}$$

La función Lagrangiana por tanto es:

$$L(\lambda) = max_{x \in X}\{f(x) - \lambda g(x)\}$$

Y el problema dual consiste en minimizar $L(\lambda)$ como sigue:

(DL) Minimizar $L(\lambda)$

sujeto a

$$\lambda \geq 0 \tag{1.6}$$

Utilizando la propiedad de acotación para $\hat{x}$ una solución óptima del original, si tenemos en cuenta que $g(\hat{x}) \leq 0$:

$$L(\lambda) = max_{x \in X}\{f(x) - \lambda g(x)\} \geq f(\hat{x}) - \lambda g(\hat{x}) \geq f(\hat{x}) = Z_{pe}$$

La diferencia con el problema de minimización radica en que la cota que obtenemos es una cota superior del problema original (UB) y buscaremos la cota inferior mediante técnicas heurísticas.

1.4.2. Ejemplo problema de maximización. Fábrica de galletas

Veamos un caso práctico para ilustrar el modelo de maximización. Supongamos que una fábrica de galletas está dispuesta a renovarse produciendo 4 tipos nuevos de galleta. Sus opciones son producir el tipo $i = 1, 2, 3, 4$ donde $x_i = 1$ o no producirlo $x_i = 0$. Cada tipo de galleta genera un beneficio y produce un gasto por su fabricación a la empresa:

.	Tipo 1	tipo 2	Tipo 3	Tipo 4
Beneficio	3	5	6	8
Gasto	1	2	4	1

Su objetivo es maximizar las ganancias, condicionado a que el gasto total sea menor o igual que 5. El problema inicial quedaría modelado:
(PE)

$$\text{maximizar } Z = 3x_1 + 5x_2 + 6x_3 + 8x_4$$
$$\text{sujeto a}$$
$$x_1 + 2x_2 + 4x_3 + x_4 \leq 5$$
$$x_i \in \{0, 1\}$$

Introducimos las restricciones complicantes en la función objetivo mediante el multiplicador de Lagrange $\lambda \geq 0$. El subproblema lagrangiano viene dado por:
P(λ)

$$\text{maximizar } 3x_1 + 5x_2 + 6x_3 + 8x_4 - \lambda(x_1 + 2x_2 + 4x_3 + x_4 - 5)$$
$$\text{sujeto a}$$
$$x_i \in \{0, 1\}$$

La función Lagrangiana del problema es:

$$\mathbf{L}(\lambda) = 5\lambda + max\{(3-\lambda)x_1 + (5-2\lambda)x_2 + (6-4\lambda)x_3 + (8-\lambda)x_4\}$$

Y el problema dual por tanto, será de la forma:
(DL)

$$\text{minimizar } \mathbf{L}(\lambda) = 5\lambda + max\{(3-\lambda)x_1 + (5-2\lambda)x_2 + (6-4\lambda)x_3 + (8-\lambda)x_4\}$$
$$\text{sujeto a}$$
$$\lambda \geq 0$$

Para encontrar el máximo de **(DL)**, impondremos $x_i = 0$ en los tramos de λ donde el coeficiente de x_i es negativo y $x_i = 1$ en los tramos donde el coefiente x_i es positivo. Nos queda una familia de soluciones de la forma:

Familia de soluciones				
Intervalo	x_1	x_2	x_3	x_4
$0 \leq \lambda \leq 1.5$	1	1	1	1
$1.5 \leq \lambda \leq 2.5$	1	1	0	1
$2.5 \leq \lambda \leq 3$	1	0	0	1
$3 \leq \lambda \leq 8$	0	0	0	1
$8 \leq \lambda$	0	0	0	0

Con los datos de la tabla construímos $L(\lambda)$ lineal a trozos en cada intervalo:

$$L(\lambda) = \begin{cases} -3\lambda + 22 & si \quad 0 \leq \lambda \leq 1.5 \\ \lambda + 16 & si \quad 1.5 \leq \lambda \leq 2.5 \\ 3\lambda + 11 & si \quad 2.5 \leq \lambda \leq 3 \\ 4\lambda + 8 & si \quad 3 \leq \lambda \leq 8 \\ 5\lambda & si \quad 8 \leq \lambda \end{cases}$$

Representamos la función $L(\lambda)$ en el plano $X-Y$, donde el eje de las X corresponde a los valores de λ y eje de las Y a los valores de $L(\lambda)$, véase [Figura 1.5].
Es una función decreciente hasta el punto $\lambda = 1.5$ y creciente a partir de ese punto, por lo que ya podemos intuir que ahí será donde alcance su mínimo.

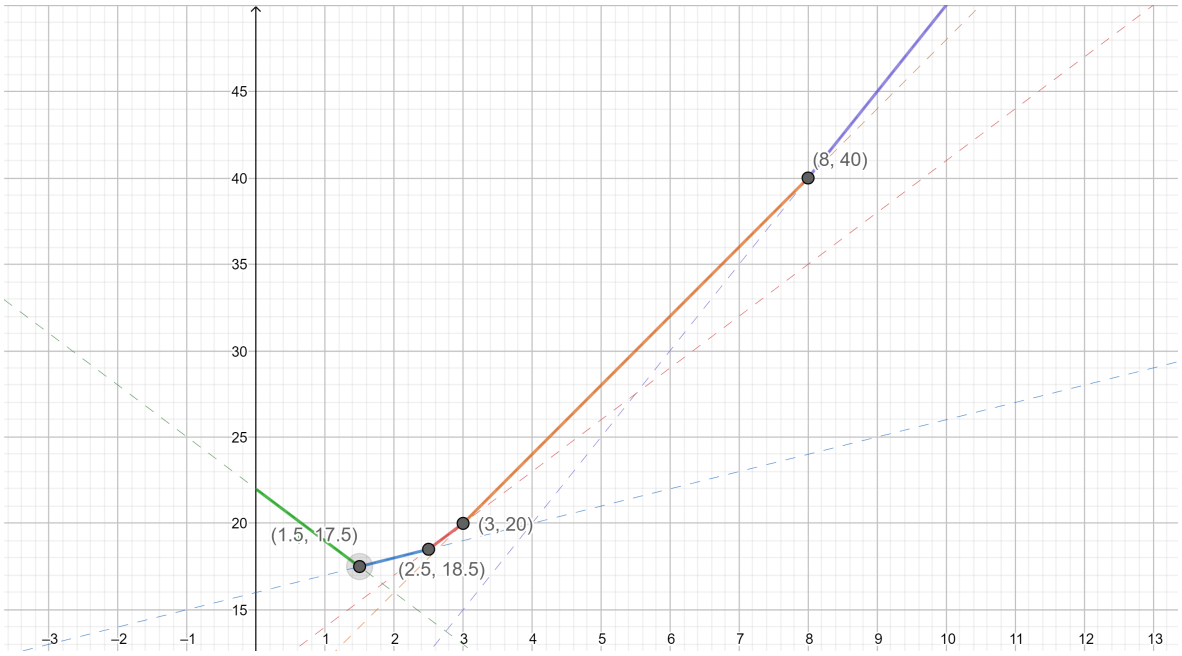


Figura 1.5: Solución gráfica del problema dual al ejemplo de las galletas

La función $L(\lambda)$ es convexa y alcanza su mínimo en el punto (1.5,17.5), es decir, para $\lambda = 1.5$. En ese punto se presentan dos soluciones óptimas para los valores:

$$\textbf{S1}\ x_1, x_2, x_3, x_4 = 1$$
$$\textbf{S2}\ x_1, x_2, x_4 = 1 \text{ y } x_3 = 0$$

Evaluamos $L(\lambda)$, $P(\lambda)$ y (PE) para estos casos:

S	x_1	x_2	x_3	x_4	$L(\lambda)$	$P(\lambda)$	(PE)
S1	1	1	1	1	17.5	17.5	22
S2	1	1	0	1	17.5	17.5	16

El valor en el problema original para S1 es mayor, pero no cumple la restricción $x_1 + 2x_2 + 4x_3 + x_4 = 8 \nleq 5$, por tanto no es una solución factible para el problema original.

En consecuencia, nos quedamos con S2. Si la empresa fabrica galletas de tipo 1,3 y 4 maximizará sus ganancias cumpliendo las condiciones pedidas.

1.4.3. Restricciones de igualdad

Hasta ahora las restricciones complicantes han venido dadas por una desigualdad, presentándose como $Ax \leq b$ o $Ax \geq b$ y haciendo que la función $g(x)$ fuese negativa. Podemos encontrarnos en el caso de que aparezcan de la forma $Ax = b$.
La restricción complicante de igualdad, $Ax - b = 0$, puede ser remplazada por dos restricciones con desigualdades:

$$Ax - b \leq 0 \qquad \text{y} \qquad -Ax + b \leq 0$$

Por lo que se necesitan dos multiplicadores Lagrangianos $\lambda \geq 0$ y $\mu \geq 0$ para construir la relajación Lagrangiana del problema, siendo esta:

$$L(\lambda, \mu) = \{f(x) + \lambda(Ax - b) + \mu(-Ax + b)\} \qquad \text{con } x \in X$$

Sacando factor común a $g(x) = Ax - b$ me queda:

$$L(\lambda, \mu) = \{f(x) + g(x)(\lambda - \mu)\} \qquad \text{con } x \in X$$

Y finalmente, si definimos la variable $\rho = \lambda - \mu$, la relajación Lagrangiana resulta ser de una variable, la cual puede tomar valores tanto negativos como positivos.

$$L(\rho) = \{f(x) + g(x)\rho\} \qquad \text{con } x \in X$$

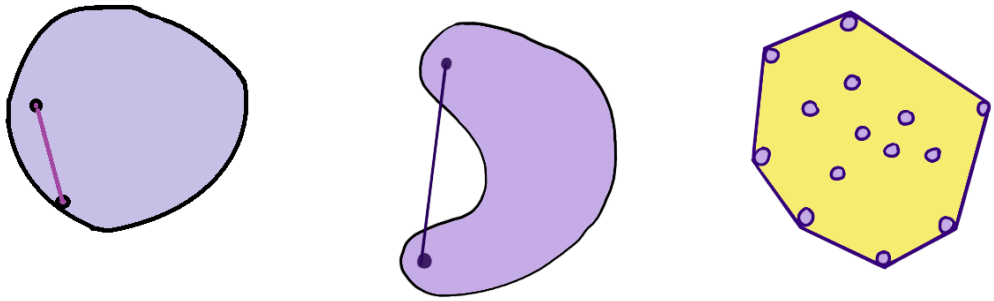
1.5. Teorema de convexificación y dualización

En esta sección vamos a tratar un resultado teórico de gran importancia para problemas de optimización que nos relaciona la convexificación y la dualización de funciones. En el Apéndice A se tratan algunos resultados de análisis convexo que utilizaremos en la sección.

Definición 1.9. Un **conjunto convexo**, $S \subseteq \mathbb{R}^n$, se define como aquel que para cada par de valores $x, y \in S$ y para todo α con $0 \leq \alpha \leq 1$, verifica que:

$$z = \alpha x + (1 - \alpha)y \qquad \in S$$

Definición 1.10. La **envoltura convexa** de un conjunto cualquiera C se define como el mínimo conjunto convexo que contiene a C. Ejemplo, [Figura 1.6 (c)]



(a) Conjunto convexo (b) Conjunto no convexo (c) Envoltura convexa del conjunto de puntos

Figura 1.6:

Cuando en el problema primal (PE) tratamos con funciones $f(x)$ y $g(x)$ convexas y el conjunto X es también convexo, el mínimo es alcanzado y este será igual al valor del máximo del problema dual (DL), dejando un hueco de dualidad nulo.
Si el problema primal no es convexo, como sucede en programación entera, el hueco de dualidad normalmente es positivo y se busca una aproximación al valor óptimo con un algoritmo.

Vamos a comenzar representando en el plano el conjunto $[f(X), g(X)]$ en color lila. Sea $[f, g]$ el conjunto dado por:

$$[f, g] = \bigcup_{x \in X} \{(\eta, \xi) / \ \eta \geq f(x), \ \ \xi \geq g(x)\}$$

Denotaremos por $[f, g]^c$ a la envoltura convexa de $[f, g]$ y $L(\lambda) = y_0 + \lambda y$

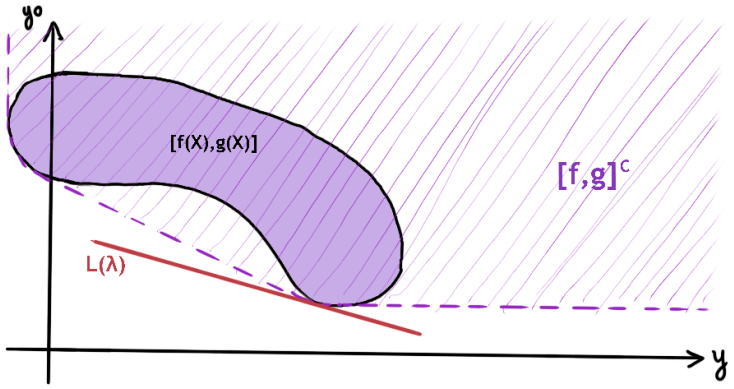


Figura 1.7: Representación de $[f(X), g(X)]$, $[f, g]^c$ y $L(\lambda)$

Lema 1.2. Para todo $\lambda \geq 0$, el hiperplano $L(\lambda) = y_0 + \lambda y$ es hiperplano de soporte del conjunto $[f, g]^c$. Además, se tiene que $L(\lambda) \leq \eta + \lambda\xi$ para todo $(\eta, \xi) \in [f, g]^c$.

La demostración a este lema que no se ha incluido, puede ser consultada en el libro de Shapiro [14].

Definimos ahora el valor:

$$v^c(\xi) = inf\{\eta / (\eta, \xi) \in [f, g]^c\}$$

En caso de que no exista ningún $(\eta, \xi) \in [f, g]^c$, por convenio será $+\infty$.

Definición 1.11. El valor $v^c = v^c(0)$ se denomina valor del problema primal convexificado.

1.5.1. Teorema de convexificación y dualización

Teorema 1.1. Teorema de convexificación y dualización. El valor óptimo del problema dual (DL) es igual al valor del problema primal convexificado.

$$v^c = Z_d$$

Demostración. Para el caso general en $\mathbb{R}^n$.

Veamos primero la desigualdad $Z_d \leq v^c$

Si $v^c = +\infty$, es trivial.

Supongamos que $v^c \neq \infty$, tomamos un $(\eta, 0) \in [f, g]^c$ arbitrario. Por el lema anterior, para cualquier $\lambda \geq 0$ se tiene que:

$$L(\lambda) \leq \eta + \lambda 0 = \eta$$

En concreto se cumple para el λ donde se alcanza el óptimo y $Z_d = sup\ L(\lambda) \leq \eta$

Como $(\eta, 0)$ fue tomado de forma arbitraria, esto significa que se cumple esa condición para todos los $\eta \in [f, g]^c$ y podemos concluir que $Z_d \leq v^c$.

Veamos ahora la otra desigualdad, $v^c \leq Z_d$

Si $v^c = -\infty$, vuelve a ser trivial.

Supongamos que $v^c \neq -\infty$, de esta manera, existe un valor $r \in \mathbb{R}$ arbitrario tal que $r < v^c$. Entonces, el valor $(r, 0) \notin [f, g]^c$. Como $[f, g]^c$ es un conjunto cerrado y convexo, existe un hiperplano $\beta = \lambda_0 y_0 + \lambda y$ que separa estrictamente $(r, 0)$ del conjunto $[f, g]^c$. El vector (λ_0, λ) y β satisfacen:

$$\lambda_0 r + \lambda 0 < \beta \leq \lambda_0 \eta + \lambda \xi \qquad \text{para todo } (\eta, \xi) \in [f, g]^c$$

Como η y cada componente $\xi_1, ..., \xi_n$ de ξ están acotadas inferiormente por $[f, g]$, la segunda desigualdad implica que los $\lambda_i \geq 0$ para $i = \{1, ..., n\}$.

Para completar la prueba tenemos que distinguir dos casos:

1. Existe algún punto $(\eta, 0) \in [f, g]^c$ para algún $\eta \in \mathbb{R}$.

Se sigue por la desigualdad anterior que $\lambda_0 \neq 0$, pues si no, llegaríamos a que $0 < \beta \leq 0$, absurdo.

Mediante escalado de λ, λ_0 y β, asumimos que $\lambda_0 = 1$. El conjunto $[f(x), g(x)]$ está contenido en $[f, g]$ para cada $x \in X$, lo que implica:

$$L(\lambda) = min_{x \in X}\{f(x) + \lambda g(x)\} \geq \beta > r$$

Como r era un valor arbitrario con $r < v^c$, tenemos que $v^c \leq Z_d = sup\ L(\lambda)$.

2. No existe un punto $(\eta, 0) \in [f, g]^c$.

Lo que implica que $v^c = \infty$. Entonces, el conjunto $\{(r, 0), r \in \mathbb{R}\}$ y $[f, g]^c$ son disjuntos, cerrados y convexos, y por tanto existe un hiperplano tal que:

$$\lambda_0 r + \lambda 0 < \beta \leq \lambda_0 \eta + \lambda \xi$$

para todo $(\eta, \xi) \in [f, g]$ y todo $r \in \mathbb{R}$. Al igual que antes, tenemos que $(\lambda_0, \lambda) \geq 0$. Tomamos $r \longrightarrow \infty$, lo que fuerza que para que se cumpla la primera desigualdad, $\lambda_0 = 0$ y entonces β será positivo. Además $\beta \leq \lambda \xi$ y para todo $(\eta, \xi) \in [f, g]$ y cualquier $K > 0$ se tiene que:

$$K\beta \leq (K\lambda)\xi$$

Tomando $\xi = g(x)$, esta desigualdad implica que:

$$K\beta \leq min_{x \in X}[(K\lambda)g(x)]$$

Entonces, si $l = min_{x \in X}[f(x)]$ se tiene que:

$$K\beta + l \leq min_{x \in X}[f(x) + (K\lambda)g(x)] = L(K\lambda)$$

Dejando que K vaya hacia ∞ se demuestra que:

$$v^c = Z_d = supL(\lambda) = \infty$$

Lo que completa la prueba del teorema de convexificación y dualización. $\square$

1.5.2. Consecuencias del teorema

El teorema de convexificación y dualización es un resultado teórico importante. Nos asegura que el óptimo de (DL), Z_d, va a ser igual al valor del problema primal convexificado v^c, esto provoca un par de resultados importantes que hemos de tener en cuenta.

Observación 1.1. La cota obtenida con el dual Lagrangiano, Z_d va a ser mejor que la cota que da la relajación lineal simple, Z_{RL}, esto es:

$$Z_{RL} \leq Z_d \leq Z_{pe}$$

Esto se debe a que la relajación lineal ya es un problema convexo.
Partimos de un problema original donde se trabaja con el conjunto X. En la relajación Lagrangiana se trata con el conjunto $Conv(X)$, mientras que para la relajación lineal se trabaja con $Conv(\bar{X})$, siendo $\bar{X}$ un conjunto ya convexo que contiene a X, luego por las propiedades de los problemas relajados:

$$Conv(X) \subseteq Conv(\bar{X}) = \bar{X}$$

$$Z_{RL} = min_{x \in \bar{X}}\ cx \leq min_{x \in Conv(X)}\ cx = v^c = Z_d$$

Además se cumple la desigualdad completa de la observación, pues conocemos que el óptimo del dual es cota inferior del óptimo del problema original por la proposición 1.1. y finalmente:

$$Z_{RL} \leq Z_d \leq Z_{pe}$$

Observación 1.2. Si los subproblemas Lagrangianos tienen la propiedad de integralidad, es decir, que obtenemos soluciones enteras pese a aplicar la relajación lineal, entonces el valor del dual no será mejor que el de la relajación lineal.

Se visualizará esta observación claramente en los resultados prácticos del capítulo 3.

Capítulo 2

Algoritmo subgradiente

Una vez que tenemos la función Lagrangiana $L(\lambda)$ definida, nos interesa encontrar un valor $\hat{\lambda}$ para el cual se alcance el óptimo de esta función. Se pueden emplear varias técnicas en la resolución del problema de optimización dual, destacando entre las más populares el método *Bundle*, algunas versiones del método *Símplex* o el método subgradiente [14], [10], [11].

Esta última es una de las más aplicadas, pues consiste en un algoritmo sencillo de programar que nos proporcionará tras una serie de iteraciones una buena cota del óptimo. Todos los métodos subgradiente siguen una estructura similar que veremos en el diagrama de flujo del algoritmo, pero la distinta forma de tomar alguno de sus parámetros da lugar a las diferentes reglas, entre las cuales destacaremos la Regla de Held-Wolfe-Crowder [9].

Se introducirá además el método Greedy, una técnica heurística con la que se consigue una primera solución inicial para arrancar el algoritmo.

Las referencias principales en este capítulo han sido el libro de Shor [15] y el Shapiro [14], donde aparece toda la información relacionada con el método subgradiente.

2.1. Optimización no diferenciable

Comenzaremos dando la definición de función diferenciable. Tanto para optimización diferenciable como para no diferenciable, la idea que emplearemos en la búsqueda del óptimo es partir de un punto y seguir una dirección adecuada hacia el óptimo para ir aproximándonos a este mediante sucesivas iteraciones.

Definición 2.1. Sea f una función definida $f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$, se dice que f es diferenciable en el punto $x0 \in \mathbb{R}^n$ si existe una aplicación lineal $T : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ que cumpla:

$$f(x0 + h) = f(x0) + T(h) + \theta(h)$$

donde $\theta(h)$ verifica que $\lim_{t \to 0} \frac{||\theta(h)||}{||h||} = 0$

Definición 2.2. Se denomina gradiente de una función $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ en el punto $x \in \mathbb{R}^n$ al vector:

$$\nabla f(x) = (\frac{\partial f(x)}{\partial x1}, \frac{\partial f(x)}{\partial x2}, ..., \frac{\partial f(x)}{\partial xn})$$

Los problemas que presentan funciones diferenciables pueden ser resueltos mediante el uso de una técnica denominada método gradiente. Partiendo de un punto inicial, la dirección de movimiento que se toma en cada iteración es la del valor del gradiente en ese punto.

En la práctica, muchas de las funciones que aparecen son no diferenciables, como es en nuestro caso la función dual (1.3) y no podemos emplear la técnica del gradiente ya que no es seguro que exista en todo punto. Estos problemas los resolveremos a partir del método subgradiente, que tiene ciertas similitudes con el gradiente, pues el subgradiente apunta hacia el óptimo, resultado que nos ofrece el libro de Kiwiel [11].

Definición 2.3. En optimización no diferenciable, definimos **método subgradiente** a una agrupación particular de métodos iterativos en los que partiendo de un punto inicial λ_0, con una dirección de movimiento en cada punto dada por el subgradiente y un escalar que llamaremos tamaño de paso, se va creando una sucesión de puntos $\{\lambda_j\}$ que se aproximan al óptimo del problema.

A continuación realizamos un pequeño esbozo de lo que sería el comportamiento de estos métodos [Figura 2.1]. Partimos desde un punto λ_k y tomando una dirección de movimiento γ_k, correspondiente al valor de un subgradiente de la función en ese punto, y un valor de tamaño de paso θ_k, que nos marca cuánto nos movemos en esa dirección, llegamos al siguiente punto λ_{k+1}.

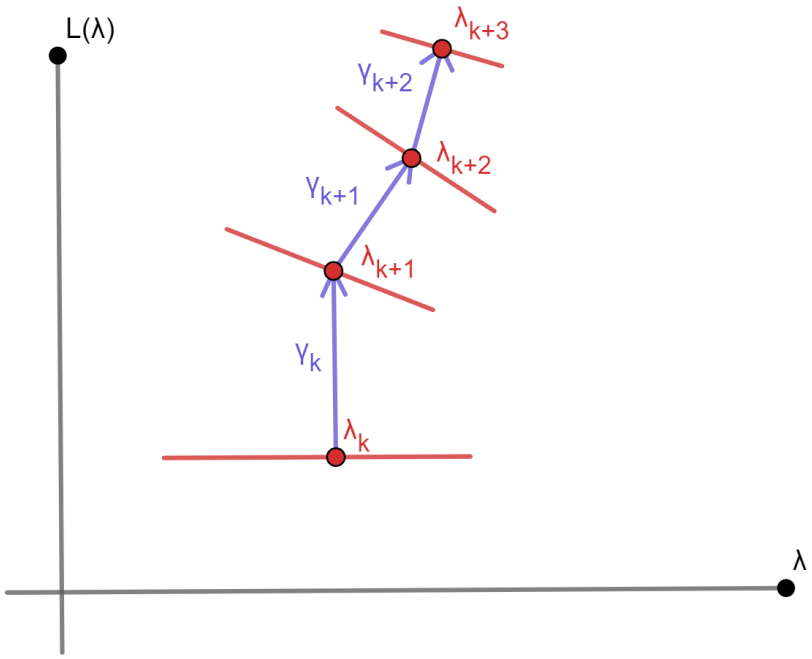


Figura 2.1: Comportamiento de las iteraciones subgradientes

La [Figura 2.1] nos muestra el proceso en zig-zag de ir ascendiendo hacia el óptimo de la función que en este caso sería un máximo. Veremos a lo largo de este capítulo como describir un algoritmo que lleve a cabo este proceso de una forma eficiente.

2.2. Método subgradiente

2.2.1. Algoritmos ascendentes

Muchas de las técnicas empleadas para la búsqueda del óptimo de la función dual se aprovechan de las propiedades que esta tiene, particularmente de la concavidad y la existencia de subgradiente en todo punto.

La forma cóncava de la función dual para problemas de minimización permite que partiendo de un punto que puede ser $\lambda_0 = 0$ y mediante una serie de iteraciones, vayamos "ascendiendo"hacia el óptimo.
El algoritmo subgradiente se considera algoritmo ascendente por la forma en que se dirige al óptimo de (DL). Al principio le damos unos parámetros entre los que se incluyen el punto inicial, el tamaño de paso y una serie de tolerancias con lo que este va mejorando la dirección hasta que consigue llegar al óptimo exacto o aproximarse hasta donde los parámetros de tolerancia le indiquen.

2.2.2. Lógica del algoritmo subgradiente

Habíamos visto en la proposición 1.3 que si x es solución del subproblema Lagrangiano P(λ), se tiene que $\gamma = g(x) = Ax - b$ es un subgradiente de la función dual L en el punto λ.
Tomamos $\hat{\lambda}$ un valor óptimo del problema dual (DL) y como vimos en [Figura 1.4], la dirección que forma γ con la dirección que va desde λ hasta $\hat{\lambda}$ es un ángulo agudo.
Nos planteamos de qué forma podemos tomar otro punto que se encuentre más cerca del óptimo que λ. Sea λ' el punto definido por:

$$\lambda' = \lambda + \theta\gamma$$

con $\theta \in \mathbb{R}$ y $\theta > 0$ lo suficientemente pequeña. Si se tiene que $\lambda' \ngeq 0$, proyectamos, es decir nos quedamos con:

$$\lambda' = max\{0,\ \lambda + \theta\gamma\}$$

Al escalar θ se le conoce como tamaño de paso o de etapa.
De una manera general por tanto, tomando un $\lambda_0 \geq 0$ inicial cualquiera, el funcionamiento del algoritmo para calcular el λ_{j+1} sería:

1. Encontrar un subgradiente γ_j de L en λ_j
2. Elegir el tamaño de paso $\theta_j > 0$
3. Calcular $\lambda_{j+1} = max\{0, \lambda_j + \theta_j\gamma_j\}$
4. Si no se cumplen las condiciones de parada repetir hasta que ocurra.

Este esquema es una pequeña introducción, más adelante profundizaremos en los pasos y las iteraciones de este algoritmo, definiendo rigurosamente cada etapa a seguir. Además, como lo que importa del subgradiente es la dirección, se suele normalizar de la siguiente manera:

$$\lambda'_{j+1} = \lambda_j + \theta_j \frac{\gamma_j}{||\gamma_j||^2}$$

2.2.3. Parámetros generales del algoritmo

Acabamos de ver un pequeño esquema de cómo trabaja el algoritmo subgradiente, para darle inicio tenemos que introducir una serie de parámetros.

Punto inicial λ_0

En la etapa inicial debemos escoger un valor de $\lambda_0 \in \mathbb{R}^n$. Como λ_0 tiene n componentes, se suele añadir un superíndice para indicar el número de componente que estamos tratanto. Es decir $\lambda_0 = (\lambda_0^1, \lambda_0^2, ..., \lambda_0^n)$.
Es habitual tomar $\lambda_0 = 0$ porque la función $L(\lambda)$ es cóncava y por tanto iremos ascendiendo por ella iteración tras iteración.

Subgradiente γ_0

Para encontrar un subgradiente γ_0 de L en el punto λ_0, tomando un $\hat{x}$ que satisfaga el subproblema Lagrangiano para ese λ_0, ya sabemos que $\gamma_0 = g(\hat{x}) = A\hat{x} - b$ será un subgradiente de la función en ese punto.
Retomando la idea de que $L(\lambda)$ es cóncava y que el algoritmo actúa de forma ascendente, podemos hacernos una idea de cómo mejorará la dirección hacia el óptimo iteración tras iteración. Se desplazará en forma de zig-zag como vimos en [Figura 2.1] y la dirección de movimiento que se toma en λ_j es la de γ_j.

Tamaño de paso θ_0

La cuestión más importante y a la que dedicaremos la próxima sección es a la toma de la sucesión de tamaños de paso, pues es la clave para que el algoritmo subgradiente converja al óptimo.
En cada iteración, además de una dirección de movimiento hay que determinar cuánto nos movemos en esa dirección y eso es lo que me marca el tamaño de paso θ_j.
Inicialmente se toma un tamaño de paso θ_0 para la primera iteración y a partir de esta se puede multiplicar o dividir dando lugar al siguiente tamaño de paso.
Las distintas formas de ir tomando esta sucesión de tamaños de paso condicionarán el uso de una regla u otra.

Tolerancias

El algoritmo debe de parar en algún momento, esto puede ser porque se ha alcanzado el óptimo en algún paso o porque se ha traspasado alguno de los límites que marcamos en un principio.
Si se llega al óptimo, la norma del subgradiente será igual a cero, por lo que hemos de establecer como primer criterio de parada $||\gamma_j|| = 0$ para algún $j = 0, 1, 2, 3...$
Para alcanzar el óptimo pueden ser necesarias demasiadas iteraciones y quizás sea mejor conformarnos con una aproximación a este que requiera menos iteraciones y que sea tan buena como nosotros establezcamos. Es por eso que se suelen fijar tres parámetros de tolerancia:
tol1: controla la norma del subgradiente. Parará si $tol1 > ||\gamma_j||^2$.
tol2: controla el tamaño de etapa. Parará si $tol2 > \theta_j$.
tol3 o itermax: número máximo de iteraciones. Suele fijarse entre 300 y 500.

Los valores de $tol1$ y $tol2$ toman valores pequeños, usualmente entre $1.e^{-4}$ y $1.e^{-5}$.
Si alcanzamos el valor de alguna de estas tolerancias el algoritmo parará, dándonos una aproximación al óptimo con un margen de error conocido.

Cada regla del método subgradiente tiene algo particular, por lo que puede requerir algún parámetro extra. La mayoría requieren una cota superior (UB) de la solución óptima, se puede optar por dar un valor muy alto o por buscarla con alguna técnica heurística como puede ser el método Greedy el cual se presentará más adelante.
En la sección siguiente nombraremos las reglas más populares con sus características que las diferencian entre sí.

2.2.4. Propiedad de los algoritmos subgradientes

No podemos asegurarnos que la función L crezca en cada iteración, es decir, no es seguro que $L(\lambda_j) < L(\lambda_{j+1})$, por ello es conveniente ir guardando estos valores y quedarnos con el mejor que encontremos mientras vayamos aplicando el algoritmo.
A pesar de que no podemos garantizar el crecimiento de L, sí que se verifica una propiedad importante sobre la norma.

Teorema 2.1. Sea L la función Lagrangiana de un problema de optimización, supongamos que existe el máximo de L que denotaremos por λ^*. Suponemos además que aplicamos el método subgradiente de tal forma que obtenemos una sucesión $\{\lambda_j\}$ con $j = 0, 1, 2, ...$ dada por la recurrencia $\lambda_{j+1} = \lambda_j + \theta_j \gamma_j$ donde γ_j es un subgradiente de L en λ_j y θ_j el tamaño de paso. Entonces para todo $j = 0, 1, 2...$ se tiene la siguiente desigualdad:

$$||\lambda^* - \lambda_{j+1}|| \leq ||\lambda^* - \lambda_j||$$

Demostración. Demostrar la desigualdad del teorema es equivalente a probar que la sucesión $\{||\lambda^* - \lambda_j||\}$ con $j = 0, 1, 2, 3...$ es monótona decreciente. Utilizando la definición de λ_{j+1} se obtiene (2.1) y desarrollando nos queda (2.2)

$$\begin{aligned}
||\lambda^* - \lambda_{j+1}||^2 =& ||\lambda^* - \lambda_j - \theta_j \gamma_j||^2 && (2.1)\\
=& ||\lambda^* - \lambda_j||^2 + \theta_j^2 ||\gamma_j||^2 - 2\theta_j(\lambda^* - \lambda_j)\gamma_j && (2.2)\\
\leq& ||\lambda^* - \lambda_j||^2 + \theta_j^2 ||\gamma_j||^2 - 2\theta_j(L(\lambda^*) - L(\lambda_j)) && (2.3)\\
=& ||\lambda^* - \lambda_j||^2 + \frac{(\lambda_j^2 - 2\lambda_j)[L(\lambda^*) - L(\lambda_j)]^2}{||\gamma_j||^2} && (2.4)
\end{aligned}$$

La desigualdad (2.3) viene dada porque λ_j es un subgradiente y la igualdad (2.4) de tomar el parámetro tamaño de paso como $\theta_j = \frac{\lambda_j[L(\lambda^*)-L(\lambda_j)]}{||\gamma_j||^2}$.

Sea $\Delta = \frac{[L(\lambda^*)-L(\lambda_j)]^2}{||\gamma_j||^2} \geq 0$ se tiene que

$$(\lambda_j^2 - 2\lambda_j)\Delta = \lambda_j(\lambda_j - 2)\Delta \leq -\epsilon_1\epsilon_2\Delta$$

con $0 < \epsilon_1 \leq \lambda_j \leq 2 - \epsilon_2$ y $\epsilon_2 > 0$. Llevando esta expresión a (2.4) me queda la desigualdad siguiente

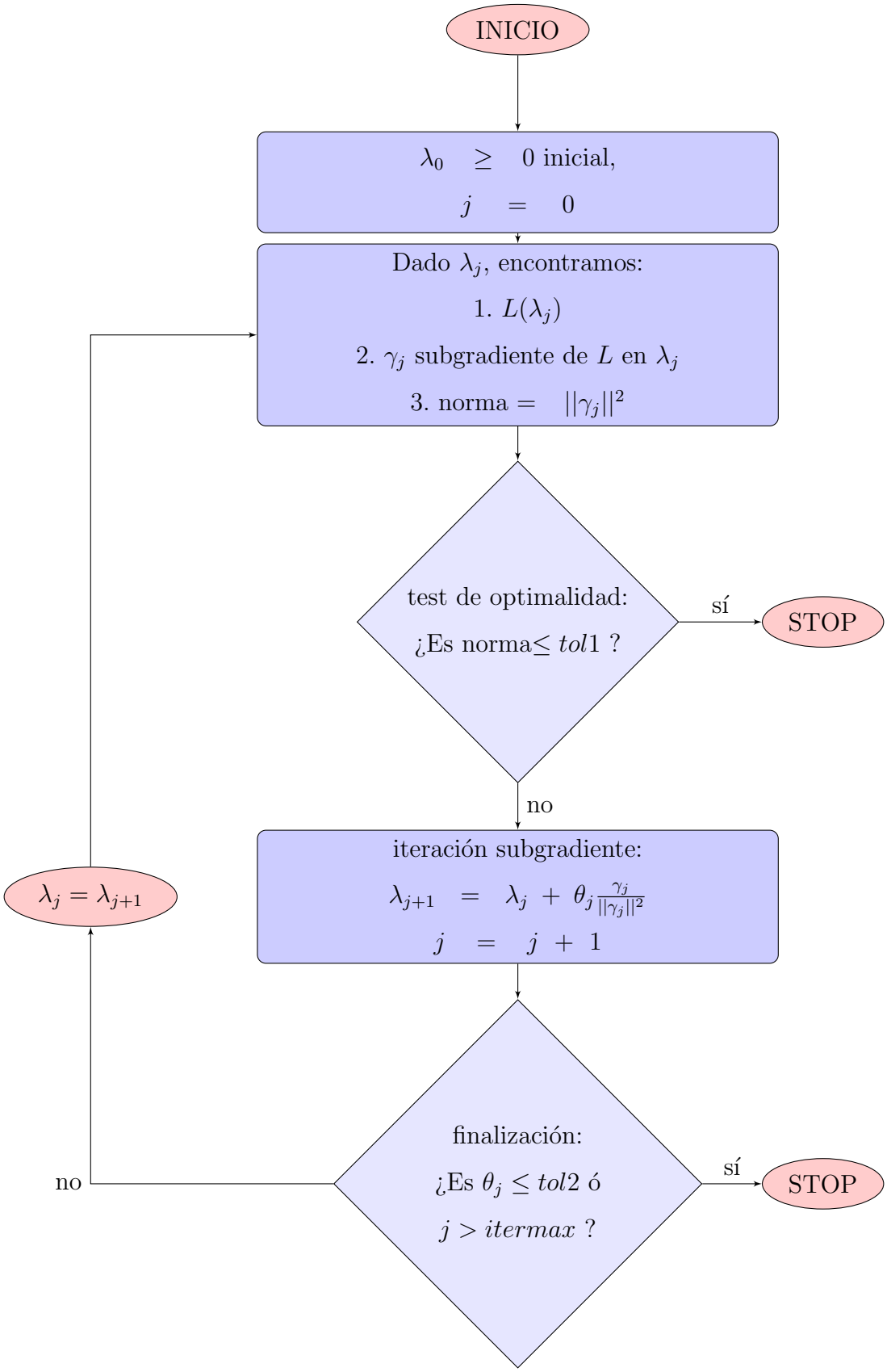
$$||\lambda^* - \lambda_{j+1}||^2 \leq ||\lambda^* - \lambda_j||^2 - \frac{\epsilon_1\epsilon_2[L(\lambda^*) - L(\lambda_j)]^2}{||\gamma_j||^2}$$

De donde deducimos finalmente que la sucesión $\{||\lambda_j - \lambda^*||\}$ con $j = 0, 1, 2, 3...$ es monótona decreciente.

□

En la práctica, obviamente no conoceremos el valor del óptimo $L(\lambda^*)$ pues es lo que pretendemos encontrar, es por eso que tomaremos en su lugar un valor UB que será cota superior de $L(\lambda^*)$.
Se suele comenzar con un valor de UB muy alto o con una primera solución obtenida con el método heurístico Greedy. Esta cota se va mejorando con la aplicación de alguna heurística Lagrangiana adecuada para cada problema. Si conseguimos alguna solución que mejore el valor actual de UB, entonces se sustituiría este por el valor heurístico recién encontrado.
Es frecuente al programar, añadir otro parámetro que nos determine con qué frecuencia se realiza la técnica heurística para la mejora de la cota UB.

2.2.5. Diagrama de flujo



2.3. Tamaño de paso del algoritmo

2.3.1. Condiciones de convergencia

El matemático ruso Boris Polyak publicó en el año 1966 un resultado [13] que nos aseguraba que si la sucesión de tamaños de paso $\{\theta_j\}$ con $j = 0, 1, 2...$ cumplía dos condiciones, entonces la sucesión de $\{\lambda_j\}$ con $j = 0, 1, 2...$ que generaba el algoritmo subgradiente convergía hacia el óptimo del problema dual Z_d.

Teorema 2.2. Sea L la función Lagrangiana cóncava para el problema dual, definida en un conjunto D y que tiene un conjunto acotado de puntos máximos M^*. Si $\{\theta_k\}$ con $k = 0, 1, 2...$ es la sucesión de tamaños de paso positivos que cumple las condiciones:

$$\lim_{k\to\infty} \theta_k = 0$$

$$\sum_{k=1}^{\infty} \theta_k = \infty$$

Entonces para cualquier $\lambda_0 \in D$, se tiene que la sucesión $\{\lambda_k\}$ con $k = 0, 1, 2...$ que viene dada por la recurrencia del algoritmo subgradiente con $\lambda_{k+1} = \lambda_k + \theta_k \frac{\gamma_k}{||\gamma_k||^2}$ verifica o bien que existe un índice $\hat{k}$ tal que $\lambda_{\hat{k}} \in M^*$ o que:

$$\lim_{k\to\infty} L(\lambda_k) = \max_{\lambda\in D}\{L(\lambda)\} = Z_d$$

Siendo Z_d el óptimo del problema dual.

Demostración. Sea $\lambda^* \in M^*$. Por el [Teorema 2.1] se tiene que dado un $\lambda_k \notin M^*$ se cumple

$$||\lambda^* - \lambda_{k+1}||^2 \leq ||\lambda^* - \lambda_k||^2 + \theta_j^2||\gamma_j||^2 - 2\theta_j(\lambda^* - \lambda_j)\gamma_j$$

Fijado un $a > 0$ consideramos el conjunto $U = \{\lambda : L(\lambda) \geq L(\lambda^*) - a\}$ y su clausura denotada por $\bar{U}$.

Por hipótesis, M^* es cerrado y acotado, luego $\bar{U}$ es compacto.
Más aún, el conjunto $M^* \cap \bar{U}$ es vacío. Esto quiere decir que existe un número que depende de a que denotaremos $b(a)$ definido como

$$b(a) = \max_{\lambda \in \bar{U}, y \in M^*} ||y - \lambda|| > 0$$

Como $\lim_{k \to \infty} \theta_k = 0$, podemos encontrar un índice N_ϵ tal que $\theta_k < \epsilon$ para todo $k > N_\epsilon$. Tomando $\epsilon = b(a)$, tendremos que para todo $k > N_{b(a)}$ se tiene que $\theta_k < b(a)$.
Si $L(\lambda_k) \leq L(\lambda^*) - a$, entonces

$$||\lambda^* - \lambda_k||^2 - ||\lambda^* - \lambda_{k+1}||^2 \geq b(a)\theta_k$$

Como $\sum_{k=0}^{\infty} \theta_k = +\infty$, debe existir un índice aún mayor $N_a^1 > N_{b(a)}$ que cumple que $L(\lambda_{N_a^1}) \geq L(\lambda^*) - a$. Definimos otro número que depende de a:

$$d(a) = \min_{y \in \bar{U}} \max_{\lambda \in M^*} ||\lambda - y||$$

Si $L(\lambda_k) \geq L(\lambda^*) - a$,
entonces $\max_{\lambda \in M^*} ||\lambda - \lambda_k|| \leq d(a)$ y $\max_{\lambda \in M^*} ||\lambda - \lambda_{k+1}|| \leq d(a) + \theta_k$.

Por otra parte, si $L(\lambda_k) < L(\lambda^*) - a$ y $k > N_{b(a)}$,
entonces $\max_{\lambda \in M^*} ||\lambda - \lambda_{k+1}|| \leq \max_{\lambda \in M^*} ||\lambda - \lambda_k||$

Por tanto, para todo $k > N_a^1$ se tiene que

$$\max_{\lambda \in M^*} ||\lambda - \lambda_k|| \leq d(a) + \max_{k > N_a^1} \theta_k$$

Como por definición $d(a) \longrightarrow 0$ si $a \longrightarrow 0$, para cada $\delta > 0$ existe $a(\delta)$ tal que $d(a(\delta)) \leq \frac{\delta}{2}$.

Además, podemos encontrar un índice N_δ tal que

$$L(\lambda_{N_\delta}) \leq L(\lambda^*) - a(\delta) \qquad\qquad \theta_k \leq \frac{\delta}{2}$$

para todo $k > N_\delta$. Así que para todo $k > N_\delta$ se tendrá

$$\max_{y \in M^*} ||\lambda_k - y|| \leq \delta$$

Esto prueba que $\lim_{k\to\infty} \max_{y\in M^*} ||\lambda_k - y|| = 0$.
Y por la continuidad de la función L se tiene finalmente que $\lim_{k\to\infty} L(\lambda_k) = L(\lambda^*)$ □

Se considera un resultado teórico pues sí es verdad que si se cumplen esas dos condiciones el algoritmo converge, pero en la práctica tenemos una convergencia bastante lenta.

2.3.2. Algunas reglas para el tamaño de paso

Dentro del algoritmo subgradiente, a lo largo de la historia de las matemáticas se han desarrollado distintas reglas para la toma del parámetro tamaño de paso. La más practicada es la de Held-Wolfe-Crowder. Todas siguen el diagrama de flujo indicado en la sección 2.2.5, la diferencia entre ellas radica en la toma del tamaño de paso.

1. Regla clásica de Held-Wolfe-Crowder

Data del año 1978 [9] y es la regla más utilizada. Hemos de disponer de una cota superior del óptimo Z_d desconocido. Normalmente se toma esta cota UB con alguna técnica heurística y puede ser el valor de cualquier solución x_0 factible conocida. Evaluando x_0 en la función principal se tiene que $UB = cx_0$.

$$L(\lambda) \leq Z_d \leq UB = cx_0$$

Como ya comentamos en el capítulo 1, la diferencia $UB - L(\lambda)$ se denomina hueco relativo. Definiremos T el tamaño de paso de que depende de θ como:

$$T_j = \theta_j \frac{UB - L(\lambda_j)}{||\gamma_j||^2}$$

Por tanto, la fórmula para las iteraciones de Held-Wolfe-Crowder se sigue:

$$\lambda_{j+1} = \lambda_j + T_j \gamma_j$$

Para este método fijamos un valor de K, que corresponde al número de iteraciones que se realizarán con el mismo valor de θ. Comenzamos con $\theta = 2$ y si tras K iteraciones no se ha alcanzado el óptimo o mejorado el valor del Lagrangiano dentro de una tolerancia marcada, se divide θ entre 2.
Dedicaremos la sección siguiente a profundizar sobre esta regla, ya que como hemos recalcado, es la más empleada por su facilidad de programar.

2. Regla de Daskin, Beasley

Los trabajos de Beasley [3] y Daskin [7] sobre heurística Lagrangiana recogieron una nueva forma de tomar el tamaño de paso. Esta regla es bastante parecida a la anterior, también se establece un parámetro K como en la de Held-Wolfe-Crowder y partiendo de $\theta = 2$, si el valor de $L(\lambda_j)$ no mejora en K iteraciones, se divide θ entre 2.
Beasley introdujo una modificación que consistía en multiplicar el valor de UB por 1.05, quedando la fórmula por tanto:

$$\lambda_{j+1} = \lambda_j + \theta \frac{1.05UB - L(\lambda_j)}{||\gamma_j||^2} \gamma_j$$

3. Regla de Avella-Boccia-Sforza-Vasil'ev

En 2009 se publicó el artículo en donde los investigadores daban a conocer esta regla [1]. Es un tanto diferente, pues se reduce el valor del tamaño de paso θ en cada iteración.

Inicialmente partimos de $\theta = 2$ y cada iteración subgradiente vamos dividiéndolo por 1.005. Se realiza esta operación independientemente de si se ha mejorado el valor de $L(\lambda)$ o no.
La fórmula de Avella-Boccia-Sforza-Vasil'ev por tanto será:

$$\lambda_{j+1} = \lambda_j + \frac{\theta}{1,005^j} \frac{UB - L(\lambda_j)}{||\gamma_j||^2}$$

4. Regla de Caprara-Fischetti-Toth

Esta regla publicada en el año 2000 por los matemáticos Caprara, Fishetti y Toth [4] comienza con un valor de θ inferior, el cual se va multiplicando o dividiendo.
Comenzamos con $\theta = 0.1$. Cada $K = 20$ iteraciones subgradientes con ese valor de θ se comparan la mejor y la peor cota inferiores calculadas en total.
Si la diferencia entre estos dos valores es más del 1 %, entonces dividimos θ entre 2.
Si la diferencia entre estos dos valores es menos del 1 %, entonces multiplicamos θ por 1.5.

2.4. Métodos heurísticos

La 23ª edición del diccionario de la lengua española de la Real Academia Española recoge el término *heurístico* como:

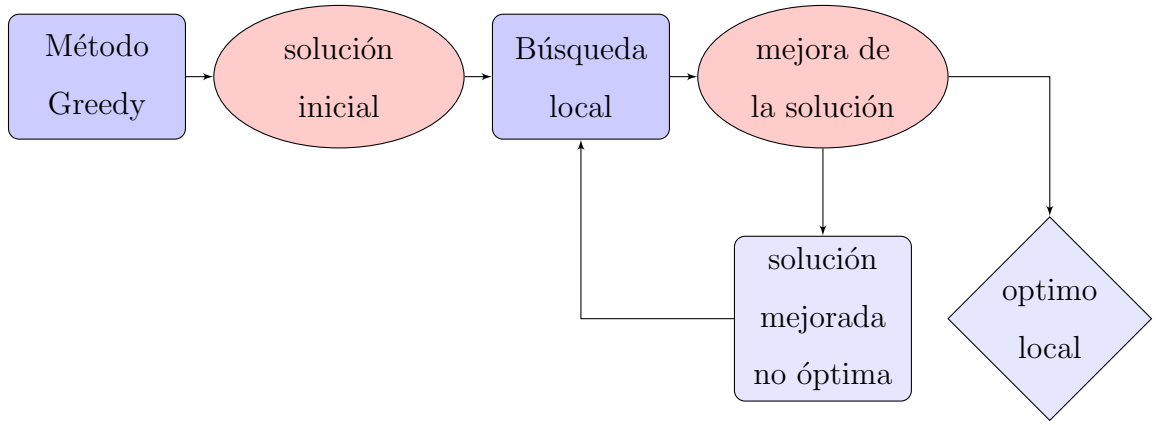
> Del griego *εύρίσκειν*, heurískein ('hallar', 'inventar') y -tíco.
> 1. adj. Perteneciente o relativo a la heurística.
> 2. f. Técnica de la indagación y del descubrimiento.
> 3. f. Búsqueda o investigación de documentos o fuentes históricas.
> 4. f. En algunas ciencias, manera de buscar la solución de un problema mediante métodos no rigurosos, como por tanteo, reglas empíricas, etc.

Nuestra definición de heurística corresponde con la acepción 4 de la RAE. Por tanto dejamos claro que los métodos heurísticos no son una técnica rigurosa que siga siempre los mismos pasos, es más una técnica que vamos ajustando a medida que tratamos el problema. Es por esto que si dos personas se encuentran aplicando la misma técnica heurística al mismo problema quizás haya alguna diferencia en el resultado numérico obtenido.

Los algoritmos heurísticos surgieron ante la dificultad para resolver de forma exacta problemas de optimización combinatoria, pues para encontrar esta solución en ocasiones es requerida mucha memoria de la máquina y un tiempo excesivo y por tanto, nos interesa más llegar a una aproximación del valor exacto en un tiempo razonable.
Las soluciones obtenidas con heurística pueden ser empleadas como punto de partida de algoritmos de tipo iterativo como es nuestro caso.

Se pueden clasificar los métodos heurísticos de muchas formas, pero la distinción mas importante es la que podemos hacer entre heurísticas de construcción y heurísticas de mejora.

Una **heurística de construcción** como su propio nombre indica, construye una solución del problema paso a paso. El método más implementado es el algoritmo Greedy, el cual profundizaremos a continuación.
Las **heurísticas de mejora** parten de una solución factible ya dada y tratan de mejorarla. Dentro de estos, los métodos más usados están basados en búsqueda local.
No son excluyentes, pues podemos realizar un método heurístico que contenga ambos tipos. Primero extraeríamos una **solución inicial** con el método Greedy y procederíamos a mejorarla con búsqueda local.



Un inconveniente que tiene la búsqueda local es que aunque mejora las soluciones obtenidas con Greedy, es una técnica que se estanca cuando llega a un óptimo local debido a su planteamiento. Esto ya se soluciona con métodos más complejos conocidos como **metaheurísticos**. No entraremos a profundizar sobre este tema complejo pero podemos citar algunos como GRASP o las redes neuronales.

2.4.1. Método Greedy

Una de las heurísticas que emplearemos en este trabajo es el denominado *método Greedy*, cuya traducción literal del inglés viene a ser *método avaricioso*. Este nombre le viene dado porque su funcionamiento a *grosso modo* es hacer lo mejor en cada paso sin tener en cuenta cómo afectarán esas decisiones en pasos siguientes. Otro nombre que se le da es *método miope* pues parece que 'no ve más allá' del paso en el que se encuentra.

Como ya adelantamos anteriormente, es una heurística de construcción que me devolverá una solución factible la cual no se garantiza que sea el óptimo y que se emplea como solución inicial en otros métodos.

Funcionamiento

El funcionamiento del método Greedy es muy sencillo y se basa en la definición de un **criterio Greedy** dado a partir de la función objetivo del problema.
El algoritmo trata de encontrar el **elemento** que minimizará o maximizará ese criterio dentro de una **lista de candidatos**. Una vez que lo tenemos, **redefinimos** el criterio Greedy y buscamos un nuevo elemento candidato que cumpla el nuevo criterio Greedy. Así hasta que tenemos una **solución completa** x.
Para tomar el elemento que optimiza el criterio Greedy se hace un barrido por la lista de candidatos buscando el índice o índices de los que minimizan o maximizan el criterio. Cómo seleccionamos el criterio es clave en el rendimiento del método y dependerá del tipo de problema, no hay una fórmula prefijada para ello.

Elementos del algoritmo Greedy

`C`: lista de elementos `e` candidatos a incluir en la solución.
`x`: solución factible que nos devuelve el algoritmo.
`criterio(e)`: array con el valor de cada elemento de C evaluado en el criterio Greedy.
`cmin`: valor mínimo del criterio. Comenzamos con valor infinito y vamos actualizando.
`emin`: elemento de C donde se alcanza el mínimo del criterio Greedy.

NOTA: En caso de maximización, se cambiaría `cmin` por `cmax` que comenzaría con un valor muy pequeño (menos infinito) y buscaríamos el elemento `emax`.

Pseudocódigo en el caso de minimización

```
x:= emptyset;
C:= todos los candidatos;
while x no sea una solución Y C ≠ emptyset do
        cmin:= infinito
        for all (e in C) do
             calcular el criterio greedy: criterio(e)
             if (criterio(e) <cmin) then
                  cmin := criterio(e)
                  emin:= e
                  C := C - emin
             end-if
        end-for
x:=x+emin
actualizar el conjunto de candidatos C;
actualizar el criterio greedy;
end-while

return x;
```

Al salir del bucle `for` tenemos guardado que `emin` es el elemento de los candidatos de `C` donde se alcanza el mínimo del criterio Greedy `(cmin)` para esa etapa, por tanto el que incluiríamos en la solución `x`.
Luego actualizaríamos la lista de candidatos `C`, redefiniríamos el criterio Greedy para la siguiente etapa y repetiríamos el bucle hasta obtener una solución completa.
La solución `x` es una solución factible del problema original, la cual no nos aseguramos que sea óptima, pues en la mayoría de los casos no lo será y la emplearemos como solución inicial para otros métodos más complejos como los procedimientos de búsqueda local. En nuestro caso utilizaremos la solución Greedy en el algoritmo subgradiente como cota inferior si minimizamos o superior si maximizamos.

2.5. Regla de Held-Wolfe-Crowder

Held, Wolfe y Crowder [9] dieron dos versiones de su regla clásica, vamos a dedicar esta sección a describir el algoritmo para la primera versión.
Como ya adelantamos anteriormente, para comenzar tomamos $\theta = 2$. Si durante K iteraciones subgradiente, no hemos mejorado el valor de $L(\lambda)$ más de una tolerancia prefijada, entonces dividiremos θ por 2.
Es usual añadir otro parámetro adicional que tiene que ver con la frecuencia heurística. La finalidad de este es que cada *heu* iteraciones se realice una heurística Lagrangiana, y si se mejora la cota superior UB que teníamos hasta ahora, actualizamos ese valor.
Normalmente antes de ejecutar el algoritmo se hace un calibrado de este para probar los valores más adecuados a nuestro problema.

Parámetros

Introducidos por el usuario al principio del programa y que pueden variar en función del problema. Suelen ser pedidos por la máquina.

tol1 #controla la norma del subgradiente.

tol2 #controla el tamaño de paso θ.

itermax #número máximo de iteraciones totales.

K #número máximo de iteraciones con el mismo tamaño de paso.

heu #frecuencia heurística

UB_0 #valor inicial de la cota superior. Obtenido con Greedy

UB #valor de la cota superior. Se va actualizando.

LB #valor de la cota inferior. Inicialmente muy pequeño para mejorarlo.

theta #valor de θ. Inicialmente 2.

iter #número de iteraciones, inicialmente cero.

lambda #valor del multiplicador de Lagrange λ. Inicialmente cero.

norma #valor de la norma del subgradiente.

2.5.1. Pseudocódigo del algoritmo subgradiente

```
Inicializamos
    iter:= 0;
    lambda:= (0,...,0);
    theta:= 2;
    K:= 30;
    heu:= 10;
    norma:= 99999;
    LB:= -99999;
    Obtenemos UB_0 con Greedy;
    UB:= UB_0;
    tol1:= 1*10^-4;
    tol2:= 1*10^-4;
    itermax:= 900;

while (iter≤ itermax, norma > tol1, theta > tol2) do
       for (j=iter, j≤iter+K, j++) do
       Resolvemos el subproblema Lagrangiano para lambda;
       Obtenemos solución óptima x;
            calculamos:
              g:= Ax-b;
              L:= f(x) + lambda · g;
              norma:= norma(g)^2;
            if (L>LB) then
                LB := L;
            end-if
```

```
            if (iter/heu = núm entero) then
                 aplicamos heurística lagrangiana;
                 obtenemos posible nueva cota superior ZUB;
                 if (ZUB<UB) then
                      UB:= ZUB;
                 end-if
            end-if

            Realizamos la iteración subgradiente;
              paso:= theta · (UB-L)/norma ;
              lambda:= max{lambda + paso · g, 0};

            iter:= iter++;
        end-for
theta:=theta/2;
end-while

return LB,UB;
```

Partimos de la base que queremos resolver un problema de optimización de la forma

$$\text{Minimizar } f(x) = \sum cx$$
$$\text{sujeto a}$$
$$Ax \leq b$$
$$x \in X$$

Donde habíamos acordado que $g(x) = Ax - b$. Como explicamos en el capítulo 1, para este problema se plantea el problema dual Lagrangiano, donde nuestro objetivo es maximizar la función $L(\lambda)$ que se ha definido tras añadir las restricciones complicantes a $f(x)$ mediante unos multiplicadores de Lagrange. Por tanto, para implementar este algoritmo lo primero que tenemos que hacer es conocer estas funciones que acabamos de nombrar.

A continuación, para resolver nuestro problema u obtener una buena aproximación con una máquina, **inicializamos** el algoritmo dando valor a los parámetros. En el cuadro anterior se han tomado valores usuales, pero el usuario puede cambiarlos a su gusto para cada problema.

Como primera cota superior UB tomamos una obtenida con el **algoritmo Greedy**, la cual iremos actualizando a medida que encontremos alguna mejor mediante **heurística Lagrangiana**. El método heurístico lo realizaremos cada *heu* iteraciones, por eso si el resultado de la división de las iteraciones que llevamos entre la frecuencia heurística es un número entero, procedemos a realizar la heurística para una posible mejora de la cota superior.

En cada iteración, a partir de un lambda **resolvemos el subproblema Lagrangiano** obteniendo una solución óptima x para ese lambda. Con ello, calculamos el valor $g(x)$, que sabemos que es el **subgradiente** de L en el punto λ y la norma de este al cuadrado para utilizarla en la iteración subgradiente. Conocido el valor de $g(x)$, se calcula fácilmente el **valor de L**(λ), pues sabemos que $L(\lambda) = f(x) + \lambda g(x)$.

Para la primera **cota inferior** LB se toma un valor muy bajo asegurando su mejora. Se va actualizando si el valor del Lagrangiano evaluado en lambda es mayor que la cota que tenemos, es decir si $LB < L(\lambda)$.

Si no cumplimos ningún criterio de parada, se realiza la **iteración subgradiente** obteniendo un nuevo lambda para repetir este proceso.

El algoritmo me devolverá el valor de las cotas inferior y superior del óptimo de mi problema original, Z. Por lo que tendremos un **GAP** conocido, GAP=$\frac{UB-LB}{UB}100$ y una acotación del óptimo tal que $LB \leq Z \leq UB$.

Capítulo 3

Problema de cubrimiento máximo

Una de las aplicaciones prácticas del método Greedy y los algoritmos subgradiente son los problemas de localización, en los cuales tendremos un conjunto de nodos $i \in I$ denominados "demandantes", que son las unidades que requieren de cierto servicio, y otro conjunto de nodos $j \in J$, que serán los lugares candidatos en los que se podrá colocar un servicio.

Existen varios tipos de problemas de localización, pero a *grosso modo*, estos tratan de minimizar unos costos o maximizar las unidades de demandantes que pueden estar cubiertos por un determinado servicio dependiendo de cuál sean los requerimientos iniciales. Partimos de la base de querer abastecer al conjunto I, para ello se pueden colocar servicios en los puntos de J. Un servicio colocado en $j \in J$ puede alcanzar a todos los nodos $i \in I$ situados a una distancia menor de la que llamaremos distancia de cubrimiento. Con estas notas, el objetivo será decidir dónde localizar los servicios dentro de la lista de candidatos.

En este capítulo profundizaremos en el modelo del problema de cubrimiento máximo, el cual pretende maximizar las unidades cubiertas sujetas a un número prefijado de servicios que podremos colocar.

Además, mediante el uso de *Xpress*, vamos a ser capaces de aplicar todo lo visto hasta ahora para buscar una solución numérica a problemas reales de este estilo.

La referencia principal para este capítulo ha sido el libro de Daskin [7].

3.1. Problemas de localización

Supongamos el caso de que el presidente de la comunidad de Cantabria, Miguel Ángel Revilla, quiere construir una serie de colegios innovadores y modernos con unas instalaciones como nunca antes se habían visto en toda su región. Los alumnos pueden ir a estudiar a un colegio que tengan como mucho a 10 kilómetros de distancia desde su casa. Se pueden dar dos casos:

CASO 1. Tiene un presupuesto infinito y por tanto, le da igual el número de edificios a construir con tal de que todos los alumnos de esa comunidad puedan alcanzar al menos una escuela a menos de esa distancia.
Su objetivo por tanto es minimizar los costes, pero con la idea de que todos los alumnos puedan tener una academia a menos de 10 kilómetros de su casa.
Este es un modelo de problema denominado de **cubrimiento total**, o en inglés, *set covering*.

CASO 2. Tiene un presupuesto limitado y desde el Gobierno Autonómico le avisan de que como máximo puede construir P colegios. Entonces él tiene que pensar cómo localizar esos servicios para que el número de alumnos que alcance sea el máximo, teniendo en cuenta que los alumnos estarán bien cubiertos sólo si la distancia que les separa es menor que 10 kilómetros.
Este modelo de problema se llama **cubrimiento máximo**, o en inglés *maximum covering*.

Ambos modelos son dos de los tipos de problemas de localización más habituales y comparten una serie de características en común, por lo que será interesante tratar primero por encima el problema de cubrimiento total, pues es el más sencillo de los problemas de localización. A continuación, se verá en detalle el problema de cubrimiento máximo y será lo que nos dedique gran parte del capítulo con su aplicación práctica.

Definición 3.1. Dado un problema de localización, se define su **distancia de cubrimiento**, denotado por *dc*, como el radio que es capaz de cubrir un servicio ubicado en un determinado lugar. Si la distancia entre el nodo $i \in I$ y el servicio ubicado en $j \in J$ es menor que la distancia de cubrimiento, el nodo i puede ser abastecido por j.

$$d(i,j) \leq dc \Rightarrow i \textit{ cubierto por } j$$

3.1.1. Problema de cubrimiento total (*set covering*)

Como hemos adelantado en la sección anterior, el **objetivo** de este problema es encontrar el **mínimo coste** para ubicar los servicios de una determinada manera que estén cubiertos **todos** los nodos de demanda.
Vamos a dar un modelo matemático para este problema. Para modelizar el problema necesitamos definir primero su estructura.

I: es el conjunto de nodos demandantes los cuales requieren del servicio. A cada nodo $i \in I$ le está asociado el conjunto $\mathbf{N_i}$ de los posibles nodos $j \in J$ que le pueden cubrir esa demanda. En el ejemplo anterior el conjunto I serían los niños cántabros en edad escolar y N_i los colegios a menos de 10 kilómetros del alumno i.

J: es el conjunto de nodos de los lugares donde se puede establecer un servicio.

Los costes de ubicar un servicio en el nodo $j \in J$ vienen dados por la función $\mathbf{f_j}$.

Los conjuntos N_i vienen dados en términos de coeficientes binarios a_{ij}, por lo que tendremos finalmente una matriz $\mathbf{A} \in \mathbb{M}_{nxm}$ tal que:

$$a_{ij} = \begin{cases} 1 \textit{ si el servicio } j \textit{ puede abastecer la demanda del nodo } i \\ 0 \textit{ otro caso} \end{cases}$$

Como el objetivo es ver dónde colocamos los servicios de localización dentro de $j \in J$, el vector de variables de decisión será X_j donde

$$X_j = \begin{cases} 1 \textit{ si colocamos un servicio en el candidato } j \\ 0 \textit{ si no} \end{cases}$$

Formulamos por tanto el problema de cubrimiento total de la siguiente manera:

$$\text{Minimizar} \sum_{j \in J} f_j X_j$$
$$\text{sujeto a}$$

$$\sum_{j \in J} a_{ij} X_j \geq 1 \quad \forall i \in I \tag{3.1}$$

$$X_j \in \{0, 1\} \quad \forall j \in J \tag{3.2}$$

Las restricciones (3.1) nos indican que el número de servicios que deben cubrir a un nodo $i \in I$ debe ser al menos 1, obviamente puede ser mayor. Las restricciones (3.2) son las restricciones de variables enteras.
Si el coste de establecer el servicio es el mismo para todos los nodos $j \in J$ podemos omitir la función f_j y minimizar simplemente $\sum_{j \in J} X_j$.

Ejemplo de cubrimiento máximo (set covering)

Estos problemas pueden representarse en forma de grafos, los nodos demandantes los denotaremos con letras mayúsculas y escribiremos en cada segmento que los une la distancia que hay entre ellos. Para el caso que vamos a ver a continuación los nodos demandantes $i \in I$ y los posibles lugares de localización de los servicios $j \in J$ serán los mismos. La distancia de cubrimiento dc será de 12 unidades. Por lo tanto, a partir de la figura 3.1, si situamos un servicio en el nodo A, este podrá abastecer a los nodos A,E y B, pero no lo podrá hacer al D pese a estar unidos, pues el segmento que los conecta sobrepasa las 12 unidades que habíamos marcado.

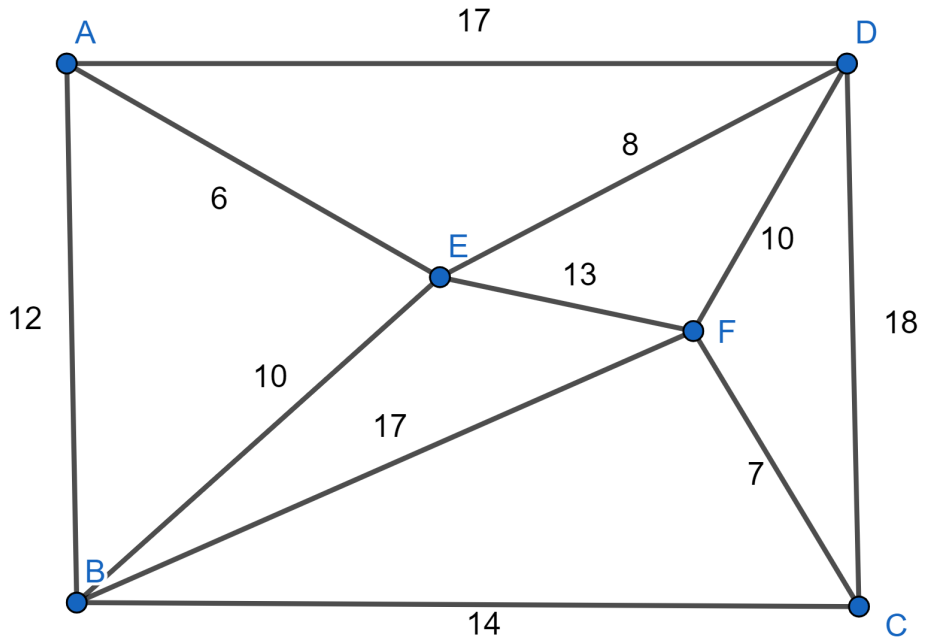


Figura 3.1: Grafo del problema de cubrimiento total

$$
\begin{array}{lrcrcrcrcrcrcl}
\textbf{Minimizar} & X_A & + & X_B & + & X_C & + & X_D & + & X_E & + X_F & & \\
\textbf{Sujeto a} & X_A & + & X_B & + & & + & & + & X_E & + & & \geq 1 \text{ (NODO A)} \\
 & X_A & + & X_B & + & & + & & + & X_E & + & & \geq 1 \text{ (NODO B)} \\
 & & + & & + & X_C & + & & + & & + X_F & & \geq 1 \text{ (NODO C)} \\
 & & + & & + & & + & X_D & + & X_E & + X_F & & \geq 1 \text{ (NODO D)} \\
 & X_A & + & X_B & + & & + & X_D & + & X_E & + & & \geq 1 \text{ (NODO E)} \\
 & & + & & + & X_C & + & X_D & + & & + X_F & & \geq 1 \text{ (NODO F)}
\end{array}
$$

$X_i \in \{0,1\}$ para todo $i = A, B, C, D, E, F$

Una solución de este problema es la dada por $X_E = X_F = 1$ y $X_A = X_B = X_C = X_D = 0$ con función objetivo $= 2$.

Para llegar a ella, hemos realizado un proceso de eliminación de filas y columnas que ahora detallaremos.

Eliminación de columnas

Si $a_{ij} \leq a_{ik}$ para todos los nodos $i \in I$ y existe algún $i \in I$ tal que la desigualdad anterior es estricta $a_{ij} < a_{ik}$, entonces se dice que la localización k es **dominante** respecto a la localización j. Podemos eliminar por tanto la columna j (equivalente a $X_j = 0$).

En nuestro problema de la figura 3.1 podemos observar cómo el nodo E es dominante sobre los nodos A y B, pues se tiene que $X_{iA} \leq X_{iE}$, $X_{iB} \leq X_{iE}$ para todas las filas con $i = 1, 2, 3, 4, 5, 6$ y además $X_{4A} < X_{4E}$, $X_{4B} \leq X_{4E}$.

Lo mismo ocurre con el nodo F, que es dominante sobre C. Por tanto, después de este proceso nos quedaría el problema de la siguiente forma

$$
\begin{array}{lrcrcrl}
\textbf{Minimizar} & X_D & + & X_E & + X_F & & \\
\textbf{Sujeto a} & & + & X_E & + & & \geq 1 \text{ (NODO A)} \\
 & & + & X_E & + & & \geq 1 \text{ (NODO B)} \\
 & & + & & + X_F & & \geq 1 \text{ (NODO C)} \\
 & X_D & + & X_E & + X_F & & \geq 1 \text{ (NODO D)} \\
 & X_D & + & X_E & + & & \geq 1 \text{ (NODO E)} \\
 & X_D & + & & + X_F & & \geq 1 \text{ (NODO F)} \\
 & X_i \in \{0,1\} \text{ para todo } i = D, E, F & & & & &
\end{array}
$$

Eliminación de filas

Si alguna variable se encuentra sola en la fila, como por ejemplo nos ha ocurrido en nuestro problema después de la reducción de columnas en las filas 1,2 y 3, esta deberá tener valor 1. En caso contrario no se cumpliría la restricción de desigualdad ≥ 1, en consecuencia $X_E = 1$ y $X_F = 1$.

Si en alguna fila ya se cumple la restricción de $\sum X_j \geq 1$ se puede también eliminar, como nos ocurre en todas las filas tras dar valores a X_E y X_F.

Después de esto, obtenemos una posible solución final del problema con $X_E = X_F = 1$, $X_A = X_B = X_C = X_D = 0$ y valor de la función objetivo = 2.

3.2. Problema de cubrimiento máximo

Acabamos de ver el más sencillo de los problemas de localización, que era el problema de cubrimiento total (*set covering*), ahora vamos a centrarnos en el problema de cubrimiento máximo (*maximum covering location model*) propuesto inicialmente por Church and ReVelle en el año 1974 [6] y que trae varias novedades respecto al anterior. Para empezar, el **número de servicios** que disponemos ahora es **limitado**, lo fijamos desde un inicio.
El modelo de *set covering* trataba a todos los nodos de demanda de la misma manera, sin embargo este modelo **distingue** entre un nodo que da servicio a 10 demandantes de uno que dé servicio a 10000.

El **objetivo** en estos problemas es que fijado el número de servicios que se pueden localizar, debemos de **maximizar** el número de demandantes cubiertos por ellos, y para ello tenemos que determinar dónde establecer estos servicios dentro de una serie de lugares candidatos.

Para modelizar este problema cambiaremos un poco la estructura anterior añadiendo unas nuevas variables.
h_i = cantidad de demanda en el nodo $i \in I$
P = número de servicios que podemos localizar

Además, ahora se añaden las variables de decisión Z_i, que me indicarán si el nodo $i \in I$ se encuentra cubierto por algún servicio o no.

$$Z_i = \begin{cases} 1 \ si \ el \ nodo \ i \ esta \ cubierto \\ 0 \ si \ no \end{cases}$$

Tomaremos las variables X_j y a_{ij} de igual forma a como lo hacíamos en el modelo de *set covering*. Con esta notación, modelizamos el problema de cubrimiento máximo de la siguiente manera:

$$Maximizar \sum_{i \in I} h_i Z_i \tag{3.3}$$

sujeto a

$$Z_i \leq \sum_{j \in J} a_{ij} X_j \quad \forall i \in I \tag{3.4}$$

$$\sum_{j \in J} X_j \leq P \tag{3.5}$$

$$X_j \in \{0, 1\} \quad \forall j \in J \tag{3.6}$$

$$Z_i \in \{0, 1\} \quad \forall i \in I \tag{3.7}$$

La función objetivo (3.3) maximiza la cantidad de demanda cubierta.
Las restricciones (3.4) nos indican que la demanda en el nodo $i \in I$ no puede ser cubierta a menos que uno de los servicios que cubre el nodo i sea seleccionado. La parte derecha de esta restricción, $\sum_{j \in J} a_{ij} X_j$, nos dará el número total de servicios que pueden cubrir al nodo i.
Las restricciones (3.5) nos indican que el número total de servicios colocados no puede ser mayor que P.
Por último se tiene (3.6) y (3.7), que nos fuerzan a que las variables Z_i y X_j sean enteras puras.

3.2.1. Ejemplo de problema de cubrimiento máximo

Al igual que antes, podemos representar estos problemas en forma de grafos. Vamos a tratar con un ejemplo sencillo donde la distancia de cubrimiento será 10 y el número total de servicios a ubicar será 1.

A diferencia del modelo de *set covering*, vemos que en la figura 3.2 los nodos presentan la cantidad de demanda, por lo que no será lo mismo cubrir el nodo C con 62 demandantes que el nodo E solamente con 6. Además se han coloreado en rojo los caminos que entran dentro de la distancia de cubrimiento, colocando un servicio en B alcanzaremos a los nodos A,E y F además del propio B, pero no llegaríamos a C.

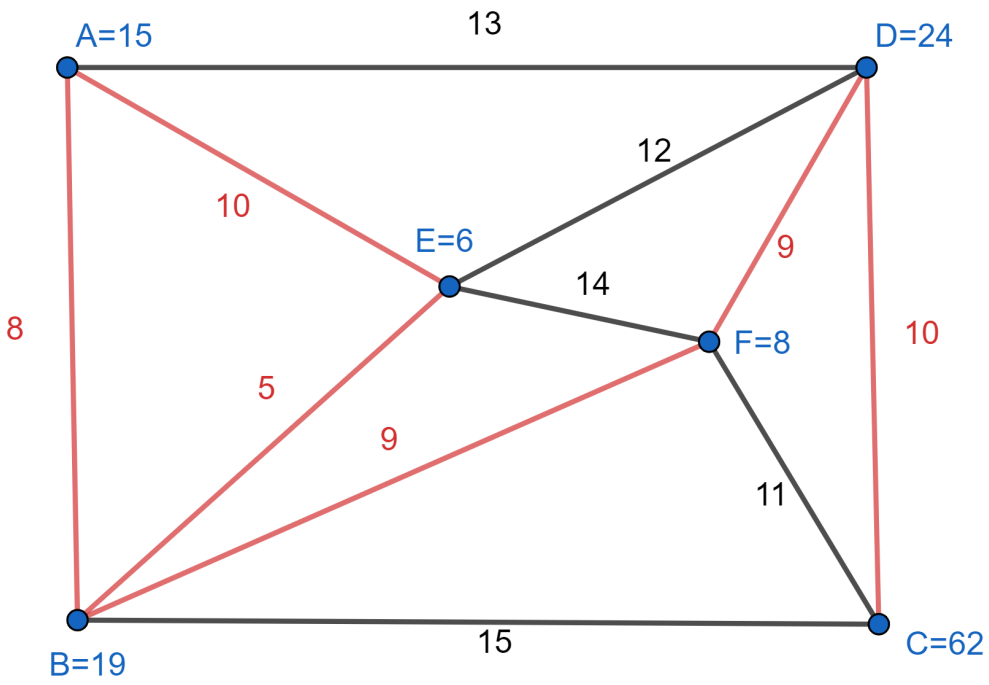


Figura 3.2: Grafo del problema de cubrimiento máximo

Maximizar $15Z_A + 19Z_B + 62Z_C + 24Z_D + 6Z_E + 8Z_F$

Sujeto a

$$
\begin{array}{rrrrrrrrrrrrl}
X_A & + & X_B & + & & + & & + & X_E & + & & \geq Z_A & \text{(NODO A)} \\
X_A & + & X_B & + & & + & & + & X_E & + & X_F & \geq Z_B & \text{(NODO B)} \\
 & + & & + & X_C & + & X_D & + & & + & & \geq Z_C & \text{(NODO C)} \\
 & + & & + & X_C & + & X_D & + & & + & X_F & \geq Z_D & \text{(NODO D)} \\
X_A & + & X_B & + & & + & & + & X_E & + & & \geq Z_E & \text{(NODO E)} \\
 & + & X_B & + & & + & X_D & + & & + & X_F & \geq Z_D & \text{(NODO F)}
\end{array}
$$

$$X_A + X_B + X_C + X_D + X_E + X_F \leq 1 \quad (\text{Nº SERVICIOS})$$

$X_i \in \{0,1\}$ para todo $i = A, B, C, D, E, F$

$Z_i \in \{0,1\}$ para todo $i = A, B, C, D, E, F$

Utilizando la técnica de **reducción de columnas** vista anteriormente, podemos eliminar las columnas de las variables A y E, pues vemos que B es dominante sobre

ellas e igualmente suprimir la columna C, pues D es dominante sobre ella.
En consecuencia deducimos que $X_A = X_E = X_C = 0$. Lo aplicamos a nuestro problema.

Maximizar $15Z_A + 19Z_B + 62Z_C + 24Z_D + 6Z_E + 8Z_F$

Sujeto a

$$
\begin{array}{lllllll}
+ X_B & + & + & + & + & \geq Z_A & \text{(NODO A)} \\
+ X_B & + & + & + & + X_F & \geq Z_B & \text{(NODO B)} \\
+ & + & + X_D & + & + & \geq Z_C & \text{(NODO C)} \\
+ & + & + X_D & + & + X_F & \geq Z_D & \text{(NODO D)} \\
+ X_B & + & + & + & + & \geq Z_E & \text{(NODO E)} \\
+ X_B & + & + X_D & + & + X_F & \geq Z_F & \text{(NODO F)}
\end{array}
$$

$$X_B + X_D + X_F \leq 1 \quad (\text{Nº SERVICIOS})$$

$X_i \in \{0,1\}$ para todo $i = B, D, F$

$Z_i \in \{0,1\}$ para todo $i = A, B, C, D, E, F$

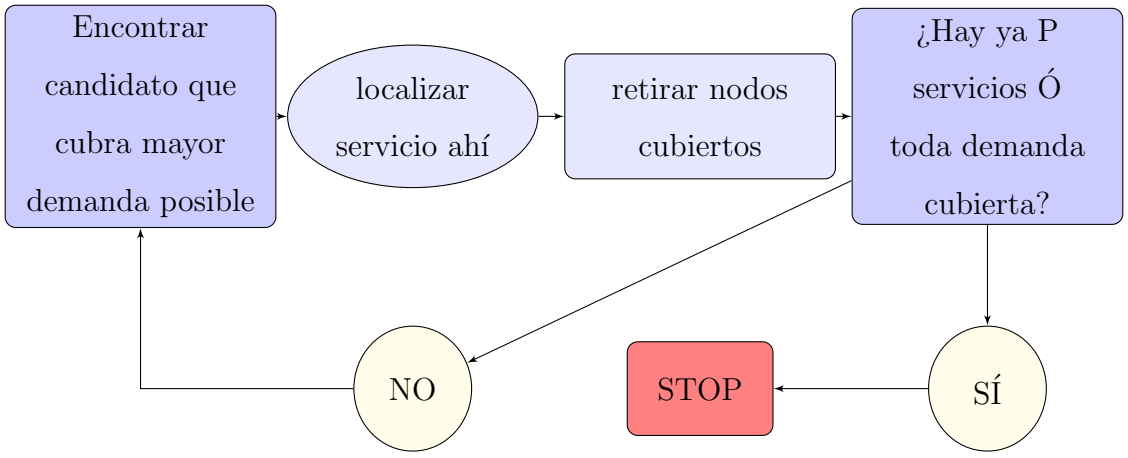
El proceso de eliminación de filas que utilizábamos con *set covering* no nos va a dar un resultado mejor para este problema, por lo que la mejor vía será probar todos los casos posibles. Ahora es fácil, pues como P=1, basta con ver por separado los valores de la función objetivo cuando $X_B = 1$, $X_D = 1$ y $X_F = 1$. Esta nos da como resultado valores de 48, 94 y 51 respectivamente, en consecuencia determinamos que localizar el servicio en el **nodo D** nos garantizará maximizar los demandantes a los que llegamos, con un total de 94 personas.
Esta técnica puede parecer una buena idea, porque con $P = 1$ a lo sumo se realizan un total de $(|I||J|)$ operaciones, donde $|I|$ es el número de nodos demandantes y $|J|$ el número de lugares candidatos para la localización. Pero, ¿qué pasa si aumentamos el número de servicios a localizar?
Cuando $P > 1$ las operaciones a realizar crecen de manera importante, como mucho se llegarían a realizar $(P|I|\binom{|J|}{P})$ contando comparaciones, donde el número combinatorio nos da las posibles combinaciones de seleccionar P servicios entre J candidatos.
Aunque tuviésemos un ordenador muy potente, con un simple problema donde tu-

viésemos valores $|I| = 200$, $|J| = 100$ y $P = 10$, el cómputo total de operaciones y comparaciones que como mucho pueden llegar a ser 34.620.618.912.880.000 podría llevarnos años realizarlo.

Por lo tanto, no podremos realizar enumeración total para resolver estos problemas por su elevado coste computacional. Hasta ciertos límites en programación entera se emplea el algoritmo *Branch and Bound* [12], que consiste en una relajación continua del problema buscando cotas enteras. Además, se han desarrollado técnicas como la relajación Lagrangiana, con la que alcanzamos un valor óptimo o aproximado al óptimo en un tiempo razonable.

3.2.2. Algoritmo Greedy para problemas de cubrimiento máximo

En el capítulo 2 ya habíamos introducido el planteamiento del algoritmo Greedy, ahora vamos a ver cómo aplicarlo a nuestro problema actual de cubrimiento máximo. La idea es ir buscando en cada movimiento el candidato que me maximice la demanda cubierta en ese instante. Si por ejemplo tenemos que localizar tres servicios (**P=3**), comenzamos buscando en qué localización la demanda cubierta será mayor y fijaríamos ahí el primer servicio (S1), después de esto, eliminamos todos los nodos demandantes que están cubiertos por S1. Repetimos el proceso para ubicar el servicio dos (S2) entre los nuevos candidatos y hacemos lo mismo para el tres (S3).



3.2.3. Relajación Lagrangiana para el problema de cubrimiento máximo

El problema que tenemos al utilizar únicamente el método Greedy en los problemas de cubrimiento máximo es que no nos garantizamos llegar a una solución óptima por sí misma. Un recurso que podemos utilizar es combinar esta técnica con la relajación Lagrangiana para así obtener una solución mejor del problema original.
Vamos a ver cómo aplicar relajación Lagrangiana a un problema de cubrimiento máximo, modelizado como en (3.3-3.7). Las restricciones (3.4) son las que denominamos **restricciones complicantes** por lo que realizaremos la relajación Lagrangiana del problema colocando los multiplicadores Lagrangianos λ_i de la siguiente manera

$$Minimizar_{\lambda} \quad Maximizar_{X,Z} \sum_{i \in I} h_i Z_i + \sum_{i \in I} \lambda_i \ (\sum_{j \in J} a_{ij} X_j - Z_i) \tag{3.8}$$

sujeto a

$$\sum_{j \in J} X_j \leq P \tag{3.9}$$

$$X_j \in \{0,1\} \qquad \forall j \in J \tag{3.10}$$

$$Z_i \in \{0,1\} \qquad \forall i \in I \tag{3.11}$$

$$\lambda_i \geq 0 \qquad \forall i \in I \tag{3.12}$$

Hemos transformado el problema original en uno dual. Para resolverlo, fijado un valor inicial de los λ_i, resolveremos el **subproblema Lagrangiano** donde nuestro propósito es **maximizar** la función objetivo (3.8) respecto a las variables de decisión $\mathbf{X_i}$, $\mathbf{Z_i}$.
El problema relajado es más sencillo, pues nos quedan las restricciones de que la suma de servicios colocados no debe superar el valor P (3.9), las de variables enteras (3.10) y (3.11), y se añaden las restricciones de no negatividad a los multiplicadores de Lagrange (3.12).
Sacando factor común a Z_i se sigue:

Problema Lagrangiano de cubrimiento máximo

$$Minimizar_{\lambda} \;\; Maximizar_{X,Z} \sum_{i \in I} (h_i - \lambda_i) Z_i + \sum_{j \in J} \; (\sum_{i \in I} a_{ij} \lambda_i) X_j \tag{3.13}$$

sujeto a

$$\sum_{j \in J} X_j \leq P \tag{3.14}$$

$$X_j \in \{0, 1\} \quad \forall j \in J \tag{3.15}$$

$$Z_i \in \{0, 1\} \quad \forall i \in I \tag{3.16}$$

$$\lambda_i \geq 0 \quad \forall i \in I \tag{3.17}$$

Resolución del subproblema Lagrangiano en cubrimiento máximo

Supongamos que tenemos un problema de cubrimiento máximo como en (3.3-3.7) y su problema Lagrangiano (3.13-3.17). Tenemos P servicios para localizar entre todos los X_j candidatos posibles.

El problema dual tiene como variables de decisión Z_i y X_j por lo que contaremos con dos subproblemas, uno para cada variable. Dado un valor inicial de los multiplicadores λ_i, buscamos la solución a los dos subproblemas Lagrangianos, los cuales se resolverán de forma trivial.

La solución para el subproblema de Z_i viene determinada por

$$Z_i = \begin{cases} 1 \; si \; (h_i - \lambda_i) > 0 \\ 0 \; si \; no \end{cases}$$

Para la solución del subproblema de X_j tenemos que encontrar los P sitios que me cubran más demanda, y esos serán los que daremos valor $X_j = 1$, el resto contarán con el valor $X_j = 0$.

Para encontrar estos P lugares que me cubran la mayor demanda, se define la variable auxiliar C_j como sigue

$$C_j = \sum_{i \in I} a_{ij} \lambda_i \quad \forall j \in J$$

Buscamos los P valores más grandes de C_j y será para esos índices j para los cuales el valor $X_j = 1$.
Una vez tenemos los valores de Z_i y X_j que nos resuelven los subproblemas Lagrangianos, se calcula el valor de la función objetivo dado por

$$Z = \sum_{i \in I} (h_i - \lambda_i) Z_i + \sum_{j \in J} (\sum_{i \in I} a_{ij} \lambda_i) X_j \tag{3.18}$$

Z será un valor cota superior del óptimo del problema original.

3.2.4. Algoritmo subgradiente para el problema de cubrimiento máximo

Para **inicializar** el algoritmo en este modelo, además de los diferentes valores de parada y frecuencia heurística que comentábamos en el capítulo dos, comenzaremos dando los valores iniciales necesarios:

- Multiplicadores λ_i^1 para la primera iteración, generalmente se les da valor 0.
- Cota inferior LB, valor de la solución obtenida con Greedy
- Cota superior UB, valor muy grande 99999999.

En cada **iteración n** del algoritmo, dado un valor de los λ_i^n que conocemos, se realizan una serie de procesos que enumeramos a continuación:

1. Se resuelven los dos subproblemas Lagrangianos de la forma que acabamos de ver, denotando sus soluciones por Z_i^n y X_j^n, indicando en el superíndice el número de la etapa.
2. Calculamos el valor de la función objetivo Z_n, sustituyendo en (3.18) y actualizamos el valor de la cota superior si $Z_n \leq UB$.
3. Si el número de iteración es divisible por la frecuencia heurística, se realiza la heurística Lagrangiana detallada a continuación (*), para ver si se actualiza la cota inferior LB.

4. Para la iteración subgradiente, en cada etapa se calculará el subgradiente para cada i y el tamaño de paso:

$$subgr_i^n = \sum_{j \in J} a_{ij} X_j^n - Z_i^n \qquad\qquad t^n = \frac{\theta^n (Z_n - LB)}{(\sum_{i \in I} subgr_i^n)^2}$$

Recordamos que θ es un valor que comienza en 2 y se va dividiendo si tras un número definido de iteraciones no se ha mejorado la cota.

Por último, actualizamos los valores de λ_i^{n+1} para la siguiente etapa, los cuales vienen dados:

$$\lambda_i^{n+1} = max\{0, \quad \lambda_i^n - t^n (\sum_{j \in J} a_{ij} X_j^n - Z_i^n)\}$$

Se reitera este proceso hasta que se cumple alguno de los **criterios de parada**, es decir si se sobrepasa el número máximo de iteraciones, o bien se alcanza la tolerancia con el parámetro θ, o bien la diferencia entre las cotas $UB - LB$ es menor que un valor prefijado.

(*) Iteración divisible por la frecuencia heurística:

El método, que a continuación detallaremos, es lo que conocemos como **heurística Lagrangiana** y que realizaremos cada vez que el número de etapa sea divisible por la frecuencia heurística.

Puede ocurrir que los valores de las variables X_j^n y Z_i^n que nos resuelvan los subproblemas no sean factibles para nuestro problema original. Como han sido "relajadas" es común que violen la restricción que hemos relajado, en nuestro caso la dada en el problema original por (3.4).

A partir de estos X_j^n y Z_i^n podemos obtener una solución factible y que además sea cota inferior del óptimo del original. Para ello simplemente tenemos que reajustar los valores de Z_i^n de la siguiente forma:

$$Zheur_i = \begin{cases} 1 \ si \ Z_i \leq \sum_{j \in J} a_{ij} X_j \\ 0 \ si \ Z_i > \sum_{j \in J} a_{ij} X_j \end{cases}$$

A continuación, calculamos el valor de la función objetivo original para estos $Zheur_i$, que vendrá dada por

$$Z = \sum_{i \in I} h_i \ Zheur_i$$

y finalmente, actualizamos la cota inferior LB en caso de que se mejore la que teníamos guardada anteriormente, es decir si $Z \geq LB$.

Lógica del algoritmo

La lógica que este algoritmo sigue con la forma de actualizar los multiplicadores Lagrangianos λ_i es que si al nodo i lo cubre más de un servicio, es decir si $\sum_{i \in I} a_{ij} X_j^n > Z_i^n$, entonces se reduce λ_i.

De la misma manera, si $Z_i^n = 1$ pero no han sido seleccionados ninguno de los nodos j que pueden cubrir a i, entonces se incrementa el valor de λ_i para que así en la siguiente iteración sea más probable que o bien ya no se tenga $Z_i^{n+1} = 1$ o bien sí que se seleccione uno de los nodos j que cubren a i.

3.3. Resolución de problemas de cubrimiento máximo

Para esta sección, se ha llevado a cabo una programación en lenguaje *Mosel* de la suite *Xpress* en la que hemos reunido y combinado cuatro métodos para la resolución de un mismo problema de cubrimiento máximo.
Estos procedimientos han sido en primer lugar la relajación lineal del problema, a continuación la resolución del modelo con el *solver* de Xpress, luego la aplicación del método Greedy y para finalizar la relajación y heurística Lagrangiana del problema.
Al reunir todo en el mismo programa, hemos conseguido que al leer un fichero de datos nos dé la solución de cada procedimiento (Sol Método) y el tiempo empleado para ello (T.Método).
Los ficheros que se han leído son de 3 tipos, recogidos en las carpetas ORLIB, CYL y SGEN. Explicamos brevemente el origen de estos datos:

1. ORLIB, archivos de la forma *.txt*
   OR-Library [2] es una de las librerías públicas de datos más conocida y utilizada. Hemos seleccionado varios archivos, los cuales son de demanda unitaria.

2. CYL, archivos de la forma *.dat*
   Corresponden a datos reales sanitarios de Castilla y León, donde los puntos de demanda son los consultorios médicos de las distintas áreas de la comunidad y los puntos de servicio los centros de salud. Las distancias vienen dadas en tiempo.

3. SGEN, archivos de la forma *.pln*
   Datos de distintos tamaños desde 50 hasta 800 generados aleatoriamente. La posición de los nodos viene dada en forma de coordenadas.

A continuación se recoge en tablas los datos obtenidos de la lectura de varios ficheros de estos tipos. Cada fichero leído aparece en una nueva columna, encabezada con el nombre de este y en cada fila aparece un dato relevante que seguidamente desarrollaremos.
P indica el número de servicios máximos a ubicar. Las dos siguientes filas son los puntos

demandantes (m) y los puntos candidatos a ubicar un servicio en ellos (n).
Seguido se presenta la solución de la relajación lineal del problema, para la que se ha empleado el comando *XPRS_LIN*. Continuamos con la resolución del modelo utilizando el *solver* de *Xpress*, en el que además del porcentaje de demanda cubierta se ha tenido el cuenta el tiempo de resolución.
Para el método Greedy se recoge también la solución, el porcentaje de demanda cubierta y el tiempo de ejecución.
Finalmente, en la Relajación Lagrangiana se reúnen los valores de la cota inferior LB, la cota superior UB, el hueco o 'GAP', el porcentaje de demanda cubierta y el tiempo empleado en realizarse este método.

Ficheros ORLIB.txt (Cuadros 3.1 y 3.2)

Comenzamos la recogida de datos con 12 archivos de la carpeta ORLIB. El valor de P se ha ido aumentando con la idea de hacer distintas observaciones, ya que en estos archivos el tamaño de puntos demandantes es el mismo y el tamaño de puntos de servicios candidatos toma el valor de 1000 o 2000, por lo que no supone una gran diferencia. En las tablas 3.1 y 3.2 podemos ver los resultados del análisis.

Ficheros CYL.data (Cuadros 3.3, 3.4 y 3.5)

Para este tipo de archivos necesitamos conocer la máxima distancia mínima (dmm) entre un punto de demanda y un candidato a servicio, ya que si la distancia de cubrimiento (dc) es menor que este valor, aunque se cubran todos los puntos de servicio no tendríamos una demanda cubierta del 100 %.
En el caso de que $dmm > dc$ se puede resolver el problema, pero teniendo en cuenta que algunos nodos demandantes nunca serán cubiertos. Por tanto, para la recogida de estos datos se añaden las filas dmm, máxima mínima distancia, y dc distancia de cubrimiento por su importancia.
En 3.3 y 3.4 aparecen los datos sanitarios de Castilla y León en los que se ha aplicado

una misma distancia de cubrimiento 30 y un mismo número de servicios a localizar 5. En la tabla 3.5 se ha fijado el archivo *aint8.dat* y se han ido haciendo variaciones de la distancia de cubrimiento y del número de servicios.

Ficheros SGEN.pln (Cuadros 3.6, 3.7, 3.8 y 3.9)

Estos archivos han sido generados de manera aleatoria y presentan el mismo número de nodos demandantes que de nodos candidatos a ubicar un servicio en ellos, pues nos encontramos en el caso en que cada nodo demandante puede ser a su vez candidato a ubicar un servicio en él.
Se han tomado muestras de distintos tamaños y para cada una se ha realizado un experimento de qué pasaría si se duplicase P con la misma distancia de cubrimiento y qué pasaría en cambio si se duplicase *dc* dejando fijo P. Puede ser interesante ya que si deseamos ampliar la demanda cubierta, teniendo en cuenta los costes que cada opción conllevaría, se podría decidir entre aumentar la distancia de cubrimiento o aumentar el número de servicios.
En la tabla 3.6 se recogen los datos de los ficheros *s50_1.pln* y *s100_1.pln*.
En la tabla 3.7 se recogen los datos de los ficheros *s200_2.pln* y *s300_2.pln*.
En la tabla 3.8 se recogen los datos de los ficheros *s500_3.pln* y *s800_3.pln*.

Se ha añadido la tabla extra 3.9 porque en ciertos archivos se ha decidido repetir el proceso disminuyendo la frecuencia heurística para observar cómo mejora así la cota inferior LB.

.	scp41.txt	scp42.txt	scp43.txt	scp44.txt	scp45.txt	scp46.txt
P	5	5	10	10	15	15
P.Dem m=	200	200	200	200	200	200
P.Serv n=	1000	1000	1000	1000	1000	1000
Sol RelLin	48	47.5	85.5	84.904	120.601	120.824
Sol Xpress	48	47	85	84	118	117
% cub	24 %	23.5 %	42.5 %	42 %	59 %	58.5 %
T. Xpress	0.11s	0.11s	0.17s	0.35s	2.13s	2.55s
Sol Greedy	48	47	85	82	116	112
% cub	24 %	23.5 %	42.5 %	41 %	58 %	56 %
T. Greedy	0.06s	0.05s	0.13s	0.14s	0.25s	0.29s
LB RelLagran	48	47	85	82	116	112
UB RelLagran	48	47.5	85.573	85.025	120.739	120.923
Gap	0 %	1.05 %	0.67 %	3.55 %	3.93 %	7.37 %
% cub	24 %	23.7 %	42.78 %	42.51 %	60.37 %	60.46 %
T. RelLagran	1.39s	4.41s	8.30s	6.42s	6.29s	5.86s

Cuadro 3.1: Datos cubrimiento máximo ORLIB.txt 1

.	scp47.txt	scp48.txt	scp49.txt	scp51.txt	scp52.txt	scp53.txt
P	20	20	25	25	25	30
P.Dem m=	200	200	200	200	200	200
P.Serv n=	1000	1000	1000	2000	2000	2000
Sol RelLin	147.886	150.796	172.100	186.544	186.536	200
Sol Xpress	141	143	164	166	170	185
% cub	70.5 %	71.5 %	82 %	83 %	85 %	92.5 %
T. Xpress	3.59s	4.8s	8.09s	29.50s	29.78s	29.67s
Sol Greedy	140	140	159	169	170	187
% cub	70 %	70 %	79.5 %	84.5 %	85 %	93.5 %
T. Greedy	0.35s	0.32s	0.39s	0.66s	0.62s	0.73s
LB RelLagran	140	140	159	169	170	187
UB RelLagran	148.066	150.949	172.255	186.856	186.741	200
Gap	5.45 %	7.25 %	7.69 %	9.56 %	8.97 %	6.5 %
% cub	74.03 %	75.47 %	86.12 %	93.43 %	93.37 %	100 %
T. RelLagran	6.25s	5.97s	7.54s	11.29s	14.23s	4.55s

Cuadro 3.2: Datos cubrimiento máximo ORLIB.txt 2

.	aint1.dat	aint2.dat	aint3.dat	aint4.dat	aint5.dat	aint6.dat
P	5	5	5	5	5	5
dc	30	30	30	30	30	30
dmm	226	304	337	337	209	269
P.Dem m=	356	628	620	765	328	145
P.Serv n=	22	34	25	35	19	10
Dem total	158005	353867	331187	477012	166142	145825
Sol RelLin	79542	253708	167286	175254	97036	86583
Sol Xpress	79542	253708	167286	175254	97036	86583
% cub	50.34 %	71.70 %	50.51 %	36.74 %	58.41 %	59.37 %
T. Xpress	0.024s	0.031s	0.031s	0.034s	0.021s	0.023s
Sol Greedy	79542	253708	167286	175254	97036	86583
% cub	50.34 %	71.70 %	50.51 %	36.74 %	58.41 %	59.37 %
T. Greedy	0.002s	0.006s	0.006s	0.002s	0.25s	0.002s
LB RelLagran	79542	253708	167286	175254	97036	86583
UB RelLagran	79542	253708.007	167286	175254	97036	86583
Gap	0 %	$2.63e^{-}6$ %	0 %	0 %	0 %	0 %
% cub	50.34 %	71.70 %	50.51 %	36.74 %	58.41 %	59.37 %
T. RelLagran	0.03s	0.162s	0.062s	0.064s	0.024s	0.006s

Cuadro 3.3: Datos cubrimiento máximo CYL.dat 1

.	aint7.dat	aint8.dat	aint9.dat	aint10.dat	aint11.dat	aint12.dat
P	5	5	5	5	5	5
dc	30	30	30	30	30	30
dmm	277	290	325	179	185	185
P.Dem m=	451	300	358	132	134	266
P.Serv n=	34	15	14	18	21	39
Dem total	336843	144216	91590	250939	260787	511726
Sol RelLin	178567	75395	58064	170261	198609	298398
Sol Xpress	178567	75395	58064	170261	198609	298398
% cub	53.01 %	52.28 %	63.40 %	67.85 %	76.16 %	58.31 %
T. Xpress	0.025s	0.022s	0.025s	0.019s	0.019s	0.022s
Sol Greedy	178567	75395	58064	170261	198609	298398
% cub	53.01 %	52.28 %	63.40 %	67.85 %	76.16 %	58.31 %
T. Greedy	0.005s	0.001s	0.001s	0.001s	0.002s	0.002s
LB RelLagran	178567	75395	58064	170261	198609	298398
UB RelLagran	178567	75395	58064	170261	198609	298398
Gap	0 %	0 %	0 %	0 %	0 %	0 %
% cub	53.01 %	52.28 %	63.40 %	67.85 %	76.16 %	58.31 %
T. RelLagran	0.074s	0.01s	0.006s	0.007s	0.027s	0.025s

Cuadro 3.4: Datos cubrimiento máximo CYL.dat 2

.	aint8.dat	aint8.dat	aint8.dat	aint8.dat	aint8.dat	aint8.dat
P	5	5	5	5	5	6
dc	100	150	200	250	300	300
Sol RelLin	104481	120665	133536	142328	144139	144216
Sol Xpress	104481	120665	133536	142328	144063	144216
% cub	72.45 %	83.67 %	92.59 %	98.69 %	99.89 %	100 %
T. Xpress	0.02s	0.024s	0.023s	0.023s	0.027s	0.025s
Sol Greedy	104481	120665	132785	139914	144062	144216
% cub	72.45 %	83.67 %	92.07 %	97.02 %	99.89 %	100 %
T. Greedy	0.002s	0.001s	0.002s	0.002s	0.001s	0.001s
LB RelLagran	104481	120665	132785	142328	144062	144216
UB RelLagran	104481	120713.52	134048.31	142349.60	144211.50	144216
Gap	0 %	0.04 %	0.94 %	0.015 %	0.10 %	0 %
% cub	72.45 %	83.70 %	92.95 %	98.71 %	99.99 %	100 %
T. RelLagran	0.069s	0.332s	0.288s	0.737s	0.123s	0.007s

Cuadro 3.5: Datos cubrimiento máximo aint8.dat variando dc y P. Con demanda m= 300, servicios n=15, demanda total=144216 y dmm=290.

.	s50_1.pln	s50_1.pln	s50_1.pln	s100_1.pln	s100_1.pln	s100_1.pln
P	5	10	5	5	10	5
dc	10	10	20	10	10	20
P.Dem m=	50	50	50	100	100	100
P.Serv n=	50	50	50	100	100	100
Dem total	2174	2174	2174	4880	4880	4880
Sol RelLin	958	1454	1798	2072	3249	4022
Sol Xpress	958	1454	1798	2072	3249	4016
% cub	44.07 %	66.88 %	82.70 %	42.46 %	66.58 %	82.30 %
T. Xpress	0.02s	0.02s	0.025s	0.022s	0.024s	0.041s
Sol Greedy	958	1454	1771	2072	3249	3863
% cub	44.07 %	66.88 %	81.46 %	42.46 %	66.58 %	79.15 %
T. Greedy	0.001s	0.002s	0.001s	0.004s	0.006s	0.003s
LB RelLagran	958	1454	1771	2072	3249	3863
UB RelLagran	958.00002	1454.009	1798	2072	3249.004	4023.17
Gap	$2.42e^{-}10$ %	0.0006 %	1.50 %	0 %	0.0001 %	3.98 %
% cub	44.07 %	66.88 %	82.70 %	42.46 %	66.58 %	82.44 %
T. RelLagran	0.015s	0.033s	0.073s	0.041s	0.119s	0.41s

Cuadro 3.6: Datos SGEN s50_1.pln y s100_1.pln duplicando P y dc

.	s200_2.pln	s200_2.pln	s200_2.pln	s300_2.pln	s300_2.pln	s300_2.pln
P	5	10	5	5	10	5
dc	10	10	20	10	10	20
P.Dem m=	200	200	200	300	300	300
P.Serv n=	200	200	200	300	300	300
Dem total	10339	10339	10339	15164	15164	15164
Sol RelLin	3798	6234	8464	4916	8504	12408
Sol Xpress	3798	6234	8464	4916	8504	12408
% cub	36.73 %	60.30 %	81.90 %	32.41 %	56.08 %	81.83 %
T. Xpress	0.035s	0.04s	0.046s	0.05s	0.053s	0.086s
Sol Greedy	3798	6234	7992	4906	8376	11387
% cub	36.73 %	60.30 %	72.30 %	32.35 %	55.24 %	75.09 %
T. Greedy	0.016s	0.028s	0.008s	0.032s	0.062s	0.028s
LB RelLagran	3798	6234	8468	4916	8504	12408
UB RelLagran	3798	6234.00008	8468.01	4916.001	8504.27	12408
Gap	0 %	$1.28e^{-}6$ %	0.0001 %	$2.02e^{-}5$ %	0.003 %	0 %
% cub	36.73 %	60.30 %	81.90 %	32.42 %	56.08 %	81.83 %
T. RelLagran	0.094s	0.958s	1.38s	1.45s	2.33s	1.544s

Cuadro 3.7: Datos SGEN s200_2.pln y s300_2.pln duplicando P y dc

.	s500_3.pln	s500_3.pln	s500_3.pln	s800_3.pln	s800_3.pln	s800_3.pln
P	5	10	5	5	10	5
dc	10	10	20	10	10	20
P.Dem m=	500	500	500	800	800	800
P.Serv n=	500	500	500	800	800	800
Dem total	24954	24954	24954	40208	40208	40208
Sol RelLin	7246	13347	19739	11308	20630	32146
Sol Xpress	7246	13347	19739	11308	20630	32146
% cub	29.04 %	53.49 %	79.10 %	28.12 %	51.31 %	79.95 %
T. Xpress	0.13s	0.09s	0.25s	0.19s	0.19s	0.37s
Sol Greedy	7246	13184	19461	11308	20439	29544
% cub	29.04 %	52.83 %	77.99 %	28.12 %	50.83 %	73.48 %
T. Greedy	0.11s	0.22s	0.09s	0.27s	0.48s	0.26s
LB RelLagran	7246	13184	19739	11308	20439	29544
UB RelLagran	7246	13347.29	19739	11308	20630.077	32184.44
Gap	0 %	1.22 %	0 %	0 %	0.93 %	8.20 %
% cub	29.04 %	53.49 %	79.10 %	28.12 %	51.30 %	80.04 %
T. RelLagran	4.67s	8.43s	6.29s	4.24s	18.75s	21.08s

Cuadro 3.8: Datos SGEN s500_3.pln y s800_3.pln duplicando P y dc

.	s50_1.pln	s100_1.pln	s800_3.pln
P	5	5	5
dc	20	20	20
P.Dem m=	50	100	800
P.Serv n=	50	100	800
Dem total	2174	4880	40208
Sol RelLin	1798	4022	32146
Sol Xpress	1798	4016	32146
% cub	82.70 %	82.30 %	79.95 %
T. Xpress	0.021s	0.051s	0.745s
Sol Greedy	1771	3863	29544
% cub	81.46 %	79.16 %	73.48 %
T. Greedy	0.306s	0.004s	0.001s
LB RelLagran	1798	4016	32146
UB RelLagran	1798.008	4022.50	32174.63
Gap	0.0004 %	0.16 %	0.08 %
% cub	82.70 %	82.42 %	80.02 %
T. RelLagran	0.093s	0.919s	100.78s

Cuadro 3.9: Datos SGEN con frecuencia heurística=1

3.4. Conclusiones

Una vez realizada la recogida de datos, vamos a proceder a sacar conclusiones. En primer lugar, aclarar que no se está realizando una competición entre el *solver* de *Xpress* y la relajación Lagrangiana, si no que la idea de dar la solución de *Xpress* es realizar comparaciones a la hora de ver el funcionamiento de los otros métodos.
El *solver* de *Xpress* resuelve los problemas de una forma muy eficaz pero presenta dos desventajas frente a la relajación Lagrangiana:

1. Las empresas no suelen disponer de él por ser un *software* de licencia muy cara.

2. Es más fácil y menos engorroso realizar la programación de la relajación Lagrangiana en otro lenguaje como puede ser C++.

Como ya habíamos probado anteriormente, podemos comprobar numéricamente que en todos los ficheros la **Relajación Lineal** nos ofrece una cota superior de la solución óptima la cual no va a ser nunca mejorada con Relajación Lagrangiana.

Respecto al **método Greedy**, vemos que nos da una solución aceptable en todos los tipos de archivo. En el caso de los datos sanitarios de CYL, cabe remarcar que el método Greedy coincide con la solución *Xpress* y en un tiempo menor.
En algunos casos en los que la solución Greedy no llega a la óptima, podemos observar cómo mediante la relajación y heurística Lagrangiana se mejora esa cota inferior, por ejemplo ocurre en 3.7 en todos los casos de *s300_2.pln*.
Apuntar que en las tablas de ORLIB 3.1 y 3.2 la solución Greedy a veces es mayor que la solución *Xpress* porque para este último método se impuso un tiempo máximo de 30 segundos y la solución que se nos da es por tanto la mejor alcanzada dentro de ese tiempo sin alcanzar el óptimo.

La última tabla 3.9 se ha añadido rebajando el valor de la **frecuencia heurística** a 1 en aquellos casos en los que con *freq_heur=10* no sé mejoraba la cota inferior en la relajación Lagrangiana. Como podemos observar, ahora LB alcanza el mismo valor que en *Xpress* y el GAP es prácticamente nulo.
El único inconveniente de aumentar la frecuencia heurística, es que el tiempo de ejecución aumenta bastante, pues se realiza la heurística en cada iteración. Es por eso que hasta ahora habíamos mantenido ese valor en 10. Aún así, es un coste asumible y obtenemos un muy buen resultado.

Podemos observar en los cuadros 3.1 y 3.2 como la **relajación Lagrangiana** en el caso de los ficheros ORLIB nos ofrece un GAP relativamente grande, esto es debido a que los datos vienen dados con demanda unitaria. En un artículo de Beasley [3] se hace referencia a que la Relajación Lagrangiana para problemas de demanda unitaria no es tan eficiente como para otros problemas, aún así se han añadido estos datos por ser una de las librerías públicas más conocidas.
En el resto de archivos se puede ver como el GAP nos ofrece unos buenos resultados, pues es nulo o alcanza valores muy pequeños. Esto nos da un reflejo de lo útil que es el método de relajación y heurística Lagrangiana para problemas de cubrimiento máximo sin demanda unitaria.

Apéndice A

Algunos resultados de análisis convexo

El análisis convexo es una rama muy amplia del análisis matemático, es por ello por lo que hemos recogido algunos resultados y definiciones necesarias para este trabajo. El libro de referencia ha sido *Fundamentals of convex analysis* de Jean-Baptiste Hiriart-Urruty y Claude Lemarechal [10]. Se recalca que es un campo muy amplio del que solo se abordará en este apéndice lo indispensable para el trabajo.

Definición A.1. Sea $a \in \mathbb{R}^n$ con $a \neq 0$ y a_0 un escalar, definimos **hiperplano** al conjunto:

$$H = \{x \ / \ ax = a_0\}$$

Definición A.2. Sea y un punto de la frontera de un conjunto cerrado y convexo X. Se considera a H **hiperplano de soporte** del conjunto X en el punto y si se cumplen:

$$ay = a_0$$

$$X \subseteq \{x/ax \geq a_0\}$$

Teorema A.1. Sea X un conjunto cerrado, convexo de $\mathbb{R}^n$ e y un punto que no pertenezca a este. Entonces existe un hiperplano que separa estrictamente y de X

Demostración. Sea x^* un óptimo del problema $min_{x \in X}||x-y||^2 = (x^*-y)^T(x^*-y)$. Como la función objetivo es continua y X es un conjunto cerrado, la solución óptima existe y satisface $(x^*-y)^T(x-x^*) \geq 0$ para todo $x \in X$.
Como $y \notin X$, también se tiene que $(x^*-y)^T(x^*-y) > 0$.
Combinando ambas desigualdades se tiene $(x^*-y)^T x > (x^*-y)^T y$ para todo $x \in X$.
Sea v el mínimo de $(x^*-y)^T x$ sobre X. Como X es cerrado, $v > (x^*-y)^T y$. El hiperplano $\{x/(x^*-y)^T x = \frac{1}{2}[(x^*-y)^T y + v]\}$ separa X e y estrictamente. □

Teorema A.2. Sea y un punto de la frontera de un conjunto cerrado convexo X. Existe un hiperplano de soporte de X en el punto y.

Demostración. Tomamos una sucesión de puntos $\{y^k\}$ convergente a y fuera de X. Sean $H^k = \{x \ / \ a^k x = a_0^k\}$ los hiperplanos que separan estrictamente y^k de y por el Teorema A.1. Sin pérdida de generalidad, suponemos que la norma $||(a^k, a_0^k)|| = 1$. Esto implica que existe una subsucesión $\{a^{k_i}, a_0^{k_i}\}$ convergente a (a^*, a_0*).
El hiperplano $H = \{x \ / \ a^* x = a_0*\}$ es hiperplano de soporte de X en el punto y, pues para todo $i \in \mathbb{N}$ se tiene $a^{k_i} x > a_0^{k_i}$ para todo $x \in X$ y $a^{k_i} y^{k_i} < a_0^{k_i}$.
Por lo tanto, en el límite tendremos $a^* x \geq a_0^*$ para todo $x \in X$ y $a^* y \leq a_0^*$.

□

Teorema A.3. Sea una función convexa $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, dado cualquier $y \in \mathbb{R}^n$, existe un subgradiente $\gamma_y \in \mathbb{R}^n$ de f en y tal que

$$f(x) \geq f(y) + \gamma_y(x-y)$$

para todo $x \in X$.

Demostración. Consideramos el conjunto convexo:

$$S = \{(w,x) \in \mathbb{R}^{n+1} / x \in \mathbb{R}^n, w \geq f(x)\}$$

Tomamos $z \in \mathbb{R}^n$ y su valor evaluado en la función $f(z) = w_z$.
Por hipótesis, como el conjunto es convexo, por el Teorema A.2 existe H un hiperplano

de soporte de S en el punto (w_z, z) perteneciente a la frontera del conjunto S. Específicamente, tomamos los coeficientes $a_o, b \in \mathbb{R}$ y $a \neq 0 \in \mathbb{R}^n$ tales que:

$$\{(w,x)/bw + ax \geq a_0\} \supseteq \{(w,x)/w \geq f(x)\} = S$$

y $bf(z) + az = a_0$.
El coeficiente b ha de ser distinto de cero, pues de otra forma, $ax \geq az \ \ \forall x \in \mathbb{R}^n$.
Por otra parte, si tomamos $x = -a + z$, $||a||^2 \leq 0$, lo cual es imposible si $a \neq 0$.
Fijamos cualquier $x \in \mathbb{R}^n$ y consideramos el punto $(f(x), x)$, que verifica:

$$bf(x) + ax \geq a_0 = bf(z) + az$$

$$f(x) \geq f(z) + \frac{-a}{b}(x - z)$$

Como habíamos fijado un x arbitrario, la definición no depende de este, por lo que acabamos de ver que existe subgradiente de f en z para todo $x \in \mathbb{R}^n$, como queríamos demostrar. □

Corolario A.1. Sea $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ una función cóncava, existe subgradiente $\gamma_y \in \mathbb{R}^n$ de f en y para cualquier $y \in \mathbb{R}^n$ tal que:

$$f(x) \leq f(y) + \gamma_y(x - y)$$

para todo $x \in X$.

Bibliografía

[1] Avella, P., Boccia, M., Sforza, A., and Vasil'ev, I. An effective heuristic for large-scale capacitated facility location problems. Journal of Heuristics 15, 6 (2009), 597–615. https://doi.org/10.1007/s10732-008-9078-y.

[2] Beasley, J. Libería pública de ficheros orlib para problemas computacionales. http://people.brunel.ac.uk/ mastjjb/jeb/info.html (1990 - Last update 2018).

[3] Beasley, J. Lagrangean heuristics for location problems. European Journal of Operation Research 65 (1993), 383–399. https://doi.org/10.1016/0377-2217(93)90118-7.

[4] Caprara, A., Toth, P., and Fischetti, M. Algorithms for the set covering problem. Annals of Operations Research 98 (2000), 353–371. https://doi.org/10.1023/A:1019225027893.

[5] Chen-Hua, C. Recent applications of the maximal covering location planning (m.c.l.p.) model. Operational Research Society 37, 8 (1986), 735–746. https://doi.org/10.1057/jors.1986.134.

[6] Church, R., and ReVelle, C. The maximal covering location problem. Papers of the Regional Science Association 32 (1974), 101–118. https://doi.org/10.1007/BF01942293.

[7] Daskin, M. Network and Discrete Location, Models, Algorithms and Applications, 2nd ed. Wiley, (2013). https://doi.org/10.1002/9781118537015.

[8] Fazel Zarandi, M., Davari, S., and Haddad Sisakht, S. The large scale maximal covering location problem. Scientia Iranica 18, 6 (2011), 1564–1570. https://doi.org/10.1016/j.scient.2011.11.008.

[9] Held, M., Wolfe, P., and Crowder, H. Validation of subgradient optimization. Mathematical Programming 6 (1974), 62–88. https://doi.org/10.1007/BF01580223.

[10] Hiriart-Urruty, J.-B., and Lemaréchal, C. Fundamentals of Convex Analysis. Springer, (2001). https://doi.org/10.1007/978-3-642-56468-0.

[11] Kiwiel, K. Methods of Descend for Nondifferentiable Optimization. Springer, (1985). ISBN 978-3-540-39509-6.

[12] Krumke, S. Integer programming: Polyhedra and algorithms, (2017). Disponible en: https://www.twirpx.com/file/2289926/.

[13] Polyak, B. Existence theorems and convergence of minimizing sequences in extremum problems with restrictions. Soviet Mathematics Docklady 7 (1966), 72–75.

[14] Shapiro, J. Mathematical Programming, Structures and Algorithms, 1st ed. Wiley, (1979). ISBN 0471778869.

[15] Shor, N. Minimization methods for non-differentiable functions. Springer Series in Computational Mathematics, (1985). https://doi.org/10.1007/978-3-642-82118-9.

[16] Snyder, S., and Haight, R. Application of the maximal covering location problem to habitat reserve site selection: a review. International Regional Science Review 39 (2016), 28–47. https://doi.org/10.1177/0160017614551276.

[17] Sáez-Aguado, J., and Camelia Trandafir, P. Some heuristic methods for solving p-median problems with a coverage constraint. European Journal of Operational Research (2012), 320–327. https://doi.org/10.1016/j.ejor.2012.02.011.

[18] Wolsey, L. Integer Programming, 2nd ed. Wiley, (2020). https://doi.org/10.1002/9781119606475.oth1.

[19] Zaslavski, A. The Projected Subgradient Algorithm in Convex Optimization. Springer Briefs in Optimization, (2020). https://doi.org/10.1007/978-3-030-60300-7.