FACULTAD DE CIENCIAS SOCIALES Y JURÍDICAS DE ELCHE

ESTADÍSICA EMPRESARIAL

2022/2023

Modelos clásicos de la Investigación Operativa

Sandy Sailema Sailema

Tutora:

Marina Leal Palazón

ÍNDICE

Resumen

En este trabajo se describe brevemente la historia de la Investigación Operativa empezando por su definición. Presentamos también algunos problemas clásicos de esta disciplina, como por ejemplo el problema de asignación, el problema de la mochila, el problema del viajante del comercio, el problema de la localización de servicios sin capacidades y por último, tenemos el problema del camino más corto. Todos ellos están definidos tanto sus restricciones como su función objetivo de manera que entendamos sus formulaciones. Cada problema lo hemos aplicado un ejemplo en concreto, explicando en qué consiste el ejemplo, las restricciones y la función objetivo que lo componen. Todos estos ejemplos han sido implementados y resueltos utilizando la librería ('lpSolve'), del programa Rstudio. Con la ayuda de esta librería hemos resuelto los ejemplos con el fin de interpretar perfectamente las soluciones.

Introducción: Orígenes de la investigación de operaciones

La investigación de operaciones es una herramienta básica para la toma de las decisiones en muchos ámbitos como por ejemplo la industria, la agricultura, la construcción, el comercio, el transporte, la energía, la Banca, la minería, las comunicaciones, los servicios públicos y privados, las empresas por su enfoque cuantitativo apoyado por las matemáticas, etc.". Como bien indica el autor (Malquin, 2022)

Con la revolución industrial, los pequeños talleres con arcaicas técnicas de producción se convirtieron en corporaciones más desarrolladas, valorizadas en millones de euros. Este cambio trajo consigo grandes ganancias pero también dificultades para asignar los recursos a las diferentes actividades que realiza la organización, encontrar la forma más eficiente de lograrlo fue uno de los causantes del surgimiento de la investigación de operaciones.

Históricamente, el origen de la Investigación de Operaciones se dio durante la Segunda Guerra Mundial cuando las Fuerzas Armadas de EE. UU. y Gran Bretaña buscaron la ayuda de científicos para resolver problemas estratégicos y tácticos complejos y muy difíciles de la guerra, como hacer que las minas sean inofensivas o aumentar la eficiencia de la guerra aérea antisubmarina, etc. De hecho, se les pidió que hicieran *investigación sobre operaciones* militares. Estos equipos de científicos fueron los primeros equipos de IO (Investigación de operaciones). Con el desarrollo de métodos efectivos, estos equipos contribuyeron al triunfo del combate aéreo inglés. A través de sus investigaciones para mejorar el manejo de las operaciones antisubmarinas y de protección, jugaron también un papel importante en la victoria de la batalla del Atlántico Norte y de muchas otras.

Comenzó a ser evidente para un gran número de personas, incluyendo a los consultores industriales que habían trabajado con o para los equipos de investigación de operaciones durante la guerra, que estos problemas eran básicamente los mismos que los enfrentados por la milicia, pero en un contexto diferente.

Cuando comenzó la década de 1950, estos individuos habían introducido el uso de la investigación de operaciones en la industria, los negocios y el gobierno. Desde entonces, esta disciplina se ha desarrollado con rapidez.

Se pueden identificar por lo menos otros dos factores que jugaron un papel importante en el desarrollo de la investigación de operaciones durante este periodo. Uno es el gran progreso que ya se había hecho en la mejora de las técnicas disponibles en esta área. Después de la guerra, muchos científicos que habían participado en los equipos de IO, se encontraban motivados a buscar resultados sustanciales en este campo; de esto resultaron avances importantes. Un ejemplo es el método simplex para resolver problemas de programación lineal, desarrollado por George Dantzing. Muchas de las herramientas características de la investigación de operaciones, como programación lineal, programación dinámica, líneas de espera y teoría de inventarios, fueron desarrolladas casi por completo antes del término de la década de 1950.

Un segundo factor más reciente que dio gran impulso al desarrollo de este campo fue (Hillier & Lieberman, 2010). "La revolución de los ordenadores. El manejo eficaz de los complejos problemas inherentes a la IO (Investigación de Operaciones) que casi siempre requieren un gran número de cálculos. Realizarlos de forma manual puede resultar casi imposible, por lo cual el desarrollo del ordenador, con su capacidad de hacer cálculos aritméticos, miles o tal vez millones de veces más rápido que los seres humanos, fue una gran ayuda para esta disciplina"

Naturaleza de la investigación de operaciones.

El objetivo de esta disciplina implica "investigar sobre las operaciones". En consecuencia, esta disciplina se aplica a la problemática relacionada con la conducción y la coordinación de actividades en una organización. Así lo definen (Hillier & Lieberman, 2010) en su libro.

"El proceso sigue en orden unas determinadas etapas que son las nombradas a continuación.

- La observación cuidadosa y la formulación del problema (incluye la recolección de los datos pertinentes)
- La construcción de un modelo científico, generalmente matemático. En esta etapa se propone la hipótesis que permitirá que las conclusiones que se obtengan sean válidas también para el problema real.
- Después se llevan a cabo los experimentos adecuados para probar esta hipótesis, para modificarla si es necesario y para verificarla en determinado momento, paso que se conoce como validación del modelo.

Modelado

La programación matemática constituye un campo amplio de estudio que se ocupa de la teoría, aplicaciones y métodos computacionales para resolver los problemas de optimización. En estos modelos se busca el extremo de una función objetivo sometida a un conjunto de restricciones que deben cumplirse necesariamente.

El objetivo principal de la Programación Matemática resuelve problemas como:

$$\text{Min} \{ f(x) : x \in S \}$$

Donde "$S \subseteq \mathbb{R}^n$ y $f : S \rightarrow \mathbb{R}$", que se lee como encontrar un elemento de S donde la función f alcance su mínimo valor. A cada elemento de S se le llama solución o solución factible, a S se le denomina región factible y f viene a ser la función objetivo. También afronta problemas de maximización, reformulándose como problemas de minimización:

$$\text{Max} \{ f(x) : x \in S \} = - \text{Min} \{ -f(x) : x \in S \}$$

Estos problemas de optimización buscan una solución factible x de manera que f(x) sea lo más pequeña posible. Existen diferentes tipos de soluciones, a continuación, tenemos los siguientes conceptos:

Problema factible: un resultado factible de un problema es una solución que satisface todas sus restricciones. El conjunto factible de un problema es el conjunto S formado por todas sus soluciones factibles. Notemos que si un problema no tiene restricciones entonces todas las soluciones son factibles, por lo que el conjunto factible es "$S = \mathbb{R}^n$", donde n es el número de variables.

Problema no factible: cuando no existe solución ninguna, es decir $S = \emptyset$ y lo escribiremos como: $\min \{ f(x) : x \in S \} = +\infty$

Problema no acotado: cuando existan soluciones que hagan descender infinitamente a la función objetivo, es decir, cuando para cualquier valor real M siempre existe un $x \in S$ tal que $f(x) < M$, en tal caso escribiremos $\min \{ f(x) : x \in S \} = -\infty$

Problema con óptimo: cuando exista un $x^* \in S$ tal que $f(x^*) \leq f(x)$ para todo $x \in S$. En este caso, $f(x^*) = \min \{ f(x) : x \in S \}$, y x^* recibe el nombre de solución optima

(global) siendo no necesariamente única en S con esta propiedad. Un punto x' se dice que es solución óptima local cuando existe un $\sigma>0$ tal que $f(x') \leq f(x)$ para todo $x \in S$ con $||x-x'|| <\sigma$, para alguna norma predefinida en el espacio vectorial $\mathbb{R}^n$. Para más información véase (Salazar González J. , 2001).

¿Qué es un programa entero?

Programa Lineal

Supongamos que tenemos un programa lineal. La programación lineal es una parte importante de la investigación operativa, dedicada a la optimización (minimización o maximización) de funciones lineales, respetando siempre un conjunto de restricciones también lineales, de igualdad o desigualdad. Este problema lo define de la siguiente manera el autor (Wolsey, 2021)

"(LP) max /min { cx : Ax ≤ b, x ≥ 0 }"

Las *x* son las variables que representan las decisiones para las cuales buscamos una configuración de valores óptima, c es un vector columna, *cx* es la función que queremos maximizar o minimizar, representa el coste o beneficio asociado a cada combinación de variables de decisión. *A* es una matriz m × n de rango m y *b* es un vector fila del lado derecho de las restricciones. Por último tenemos la restricción $x \geq 0$, que indica que las variables de decisión deben tener sólo valores no negativos (positivos o nulos) más conocida como restricción de no negatividad.

Programa entero (lineal)

Tanto en la función objetivo como en las restricciones tenemos variables enteras.

"Max cx

(IP) $Ax \leq b$

$x \geq 0$ entero"

Así lo define el autor (Wolsey, 2021) en su libro, para nosotros las *x* son las variables que representan las decisiones para las cuales buscamos una configuración de valores óptima, c es un vector columna, cx es la función que queremos maximizar o minimizar, representa el coste o beneficio asociado a cada combinación de variables de decisión. *A* es una matriz m × n de rango m y *b* es un vector fila del lado derecho

de las restricciones. Por último tenemos la restricción $x \geq 0$, que indica que las variables de decisión deben tener sólo valores no negativos y enteros.

Programa entero mixto (lineal)

A esta categoría pertenecen aquellos problemas de optimización que consideran variables de decisión enteras y continuas según el autor (Wolsey, 2021)

“ Max / Min $cx + hy$

(MIP) $Ax + Gy \leq b$

$x \geq 0$ entera $y \geq 0$”

Las x e y son las variables que representan las decisiones para las cuales buscamos una configuración de valores óptima, $cx + hy$ es la función que queremos maximizar o minimizar, representa el coste o beneficio asociado a cada combinación de variables de decisión. $Ax + Gy$ representa el conjunto de restricciones del problema y b es un vector fila del lado derecho de las restricciones. Por último tenemos unas dos últimas restricciones, la restricción $(x \geq 0)$, que indica que las variables de decisión deben tener sólo valores no negativos (positivos o nulos) y enteros. La restricción $y \geq 0$, siendo la variable y continua no negativa.

Problema entero binario

Según (Wolsey, 2021) en problema entero binario se define de la siguiente manera:

“Max cx

(BIP) $Ax \leq b$

$x \geq 0 \in \{1,0\}$”

Las x son las variables que representan las decisiones para las cuales buscamos una configuración de valores óptima, c es un vector columna, cx es la función que queremos maximizar o minimizar, representa el coste o beneficio asociado a cada combinación de variables de decisión. A es una matriz $m \times n$ de rango m y b es un vector fila del lado derecho de las restricciones. Por último tenemos la restricción $x \geq 0 \in \{1,0\}$, que indica que las variables de decisión deben tener sólo valores 1 o 0.

Estos dos valores se suelen utilizar para modelar decisiones con dos opciones, como por ejemplo sí o no.

Otro tipo de problema es el de "optimización combinatoria". Aquí se maneja una cantidad finita de posibilidades generadas a partir de un número determinado de elementos. Aquí normalmente se nos da un conjunto N = {1,…, n} con unos pesos c_j asociados para cada j ∈ N, y un conjunto $\mathcal{F}$ de subconjuntos factibles de N. El objetivo principal con este tipo de problema es encontrar un mínimo subconjunto factible de pesos que puede expresar de la siguiente manera:

Problema de optimización combinatoria

El autor (Wolsey, 2021) lo define de la siguiente manera:

" (COP) $\frac{min}{S \subseteq N}\{\sum_{j \in S} c_j : S \in \mathcal{F}\}$"

Problemas clásicos de la Investigación Operativa

A continuación formulamos algunos problemas conocidos de programación entera extraídos del libro del autor (Wolsey, 2021).

El problema de la asignación

Hay n personas disponibles para realizar n trabajos. Cada persona es asignada para llevar a cabo exactamente un trabajo. Algunas personas se adaptan mejor a trabajos particulares que otras, entonces hay un coste estimado c_{ij} si la persona i es asignada al trabajo j. Con este tipo de problema queremos encontrar una asignación de coste mínimo.

Definición de las variables.

x_{ij} = 1 si la persona i realiza el trabajo j, y x_{ij} = 0 en caso contrario.

Definición de las restricciones.

Cada persona i podrá realizar tan sólo un trabajo j. En este tipo de restricción a cada persona le asignamos tan solo uno de los distintos trabajos posibles a realizar, por ello, para la persona i, solo una de las variables x_{ij} puede tomar el valor 1, esto lo indicamos imponiendo que el resultado de la suma en todas las j (en todas las tareas posibles), para cada persona i, tome el valor de 1.

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall \quad i = 1, \dots, n$$

Cada trabajo j tan sólo lo podrá realizar una persona i. En este tipo de restricción a cada trabajo j le asignamos tan solo una de las de las distintas personas que lo puedan realizar, por ello, para el trabajo j, solo una de las variables x_{ij} puede tomar el valor 1, esto lo indicamos imponiendo que el resultado de la suma en todas las i (en todas las personas disponibles), para cada tarea j, tome el valor de 1.

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall \quad j = 1, \dots, n$$

Las variables son 0–1:

$x_{ij} \in \{0, 1\} \quad \forall\, i = 1,\ldots, n \quad y \quad \forall\, j = 1,\ldots, n.$

Definición de la función objetivo.

El coste del encargo se minimiza:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$ es la función que queremos minimizar, representa en este caso el tiempo asociado a cada combinación de variables de decisión.

Ejemplo

Tenemos 4 personas disponibles para realizar 4 estilos de natación. Cada persona es asignada para llevar a cabo exactamente un estilo. Algunas personas realizan el estilo de natación en menor tiempo que otras, entonces hay un tiempo estimado c_{ij} si la persona i es asignada al estilo j. Por ejemplo Vanesa tarda 52 segundos en nadar estilo libre, 55 segundos en estilo espalda, 65 en estilo braza y 53 segundos en estilo mariposa. Con este tipo de problema queremos encontrar una

asignación de tiempo mínimo. A continuación, disponemos de los siguientes datos. Para más información véase el Enlace [1]

Tabla 1

Tiempos Realizados por cada Estilo.

	Libre	Espalda	Braza	Mariposa
Vanessa	52	55	65	53
Jhoana	50	57	62	55
Katy	53	56	64	54
Cyntia	51	58	64	55

Fuente: Elaboración propia

Ahora pasaremos a resolver el problema en Rstudio utilizando la librería 'lpSolve'. Al ser un problema de asignación, todas nuestras variables serán binarias, como ya se ha indicado, x_{ij} valdrá 1 si la persona i realiza el estilo j, y $x_{ij} = 0$ en caso contrario. Tenemos 16 variables de las cuales 4 de ellas tomarán el valor 1, de manera que minimice el tiempo realizado en cada uno de los estilos.

```
library('lpSolve')
#funcióno bjetivo
f=c(52,55,65,53,
   50,57,62,55,
   53,56,64,54,
   51,58,64,55)
#matriz de restricciones
mat=matrix(c(0), nrow = 8, ncol=16)
#restricción:  A cada nadador se le asigna un estilo de natación.

mat[1,1]=1
mat[1,2]=1
mat[1,3]=1
mat[1,4]=1


mat[2,5]=1
mat[2,6]=1
mat[2,7]=1
```

```
mat[2,8]=1

mat[3,9]=1
mat[3,10]=1
mat[3,11]=1
mat[3,12]=1

mat[4,13]=1
mat[4,14]=1
mat[4,15]=1
mat[4,16]=1
#restricción:  A cada  estilo se le asigna un nadador.
mat[5,1]=1
mat[5,5]=1
mat[5,9]=1
mat[5,13]=1

mat[6,2]=1
mat[6,6]=1
mat[6,10]=1
mat[6,14]=1

mat[7,3]=1
mat[7,7]=1
mat[7,11]=1
mat[7,15]=1

mat[8,4]=1
mat[8,8]=1
mat[8,12]=1
mat[8,16]=1
#vector de terminos independientes (medias)
b=c(1,1,1,1,1,1,1,1)
#vector de signos
```

```
signos=c('=','=','=','=','=','=','=','=')
#declaracion de variables que son enteras
int=c(0)
#declaracion de las variables binarias
bin=c(1:16)

#solucionamos el problema

solucion=lp('min',f,mat,signos,b,int.vec = int,binary.vec = bin)
solucion$objval

        ## [1] 222

        solucion$solution

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

La solución que nos proporciona el programa es que para realizar el menor tiempo posible que es 222 segundos entre las cuatro personas, las variables que toman el valor 1 son: x_{12}, x_{23}, x_{34} y x_{41} lo que significa que a Vanesa le asignaremos el estilo de espalda, a Jhoana el estilo de braza, a Katy el estilo mariposa y a Cintia el estilo libre.

El problema de la mochila

(Conocido como Knapsack Problem) Consiste en que un excursionista debe preparar su mochila, la cual tiene una capacidad limitada y por tanto no le permite llevar todos los artículos que quisiera llevar a la excursión. Dichos artículos serán seleccionados según la mayor utilidad, teniendo en cuenta que no sobrepasarán el peso máximo que puede soportar la mochila.

Definición de las variables.

x_j es una variable binaria que tomará el valor 1 si seleccionamos el objeto para llevarlo en la mochila y 0 si lo descartamos.

Definición de las restricciones.

Tenemos para cada objeto el peso a_j multiplicado por x_j, variable que tomará un valor positivo, indicándonos así que los objetos incluidos en la mochila no pueden exceder un peso de b.

$$\sum_{j=1}^{n} a_j \, x_j \leq b$$

Las variables son binarias:

$x_j \in \{0, 1\}$

Definición de la función objetivo.

La utilizad esperada se maximiza:

$$max \sum_{j=1}^{n} c_j \, x_j$$

$\sum_{j=1}^{n} c_j x_j$ es la función que queremos maximizar, siendo c_j el beneficio que nos reporta cada uno de los objetos.

Ejemplo

Tenemos una mochila en la que guardaremos unos objetos maximizando su utilidad con un peso máximo, a continuación tenemos los siguientes datos:

Tabla 2

Utilidad y Peso de los Objetos.

	Utilidad	Peso (Kg)
Objeto 1	4	7
Objeto 2	5	6
Objeto 3	6	8
Objeto 4	3	2
Max		15

Fuente: Elaboración propia

Para más información véase el Enlace [2]

```
library('lpSolve')

#función objetivo
#Max Z = 4X1+ 5X2+ 6X3+ 3X4
f=c(4,5,6,3)
#matriz de restricciones
mat=matrix(c(0), nrow = 1, ncol=4)
#restricción 1:  Peso máximo 15 kg.
mat[1,1]=7
mat[1,2]=6
mat[1,3]=8
mat[1,4]=2
#vector de terminos independientes (medias)
b=c(15)
#vector de signos
signos=c('<=')
#declaracion de variables que son enteras
int=c(0)
#declaracion de las variables binarias
bin=c(1:4)

solucion=lp('max',f,mat,signos,b,int.vec = int,binary.vec = bin)
solucion$objval

## [1] 12

solucion$solution

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

La solución propuesta por el programa es que x_1 , x_2 y x_4 toman el valor de 1, lo que significa que el objeto 1 2 y 4 son los que maximizan la utilizad sin sobrepasar los 15 kg máximos que puede aguantar la mochila.

Ejemplo

Al ejemplo anterior típico de la mochila podemos añadirle más restricciones como por ejemplo la altura máxima, siendo éste una variante del anterior problema. A continuación, tenemos los siguientes datos:

Tabla 3

Utilidad Peso y Altura de los Objetos.

	Utilidad	Peso (Kg)	Altura (dm)
Objeto 1	4	7	7
Objeto 2	5	6	4
Objeto 3	6	8	5
Objeto 4	3	2	5
Max		15	12

Fuente: Elaboración propia


```
library('lpSolve')
#función objetivo
f=c(4,5,6,3)


#matriz de restricciones
mat=matrix(c(0), nrow = 2, ncol=4)
#restricción 1:  Peso máximo 15 kg.
mat[1,1]=7
mat[1,2]=6
mat[1,3]=8
mat[1,4]=2
#restricción 2:  Altura máxima 12 dm.
mat[2,1]=7
mat[2,2]=4
mat[2,3]=5
mat[2,4]=5
```

```
#vector de terminos independientes (medias)
b=c(15,12)
#vector de signos
signos=c('<=','<=')
#declaracion de variables que son enteras
int=c(0)
#declaracion de las variables binarias
bin=c(1:4)

solucion=lp('max',f,mat,signos,b,int.vec = int,binary.vec = bin)
solucion$objval

## [1] 11

solucion$solution

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

La solución propuesta por el programa es que x_2 y x_3 toman el valor de 1, lo que significa que el objeto 2 y 3 son los que maximizan la utilizad sin sobrepasar los 15 kg en peso y los 12 dm en altura.

El problema de la cobertura del conjunto

Dado un cierto número de regiones, el problema es decidir dónde instalar un conjunto de centros de servicios (de emergencia por ejemplo) con el objetivo de cubrir el mayor número de regiones. Para cada posible centro, el coste de instalación de un servicio se conoce y las regiones a las que puede dar servicio. Con la matriz a_{ij} sabremos si abriendo un centro j cubrimos a un punto de servicio i, mientras que con la c_j sabremos el coste de la instalación de los centros de servicio.

Definición de las variables.

$x_j = 1$ si se selecciona el centro j, y $x_j = 0$ en caso contrario.

Definición de las restricciones.

Al menos un centro debe dar servicio a la región i de manera que todas las regiones queden cubiertas.

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1 \qquad \forall\, j = 1, \ldots, m.$$

Las variables son 0–1:

$x_j \in \{0, 1\}$ $\forall$ j = 1,…, n.

Definición de la función objetivo.

El coste total se minimiza:

$$min \sum_{j=1}^{n} c_j\, x_j$$

Ejemplo

Se requiere construir hospitales de manera que den servicio a todas las 15 comunidades minimizando el costo de construcción. A continuación se presentan los datos siguientes:

Tabla 4

Hospitales que Cubren a las Diferentes Comunidades y el Coste de Construcción de los Hospitales

Hospital	Comunidades cubiertas	Coste (millones €)
1	1,2	3.6
2	2,3,5	2.30
3	1,7,9,10	4.10
4	4,6,8,9	3.15
5	6,7,9,11	2.80
6	5,7,10,12,14	2.65
7	12,13,14,15	3.10

Fuente: Elaboración propia

Para más información véase el Enlace [3]

```
library('lpSolve')

#función objetivo

f=c(3.6,2.3,4.1,3.15,2.80,2.65,3.10)
#matriz de restricciones
mat=matrix(c(0), nrow = 15, ncol=7)
#rest 1
mat[1,1]=1
mat[1,3]=1

#rest 2
mat[2,1]=1
mat[2,2]=1
#rest 3
mat[3,2]=1
#rest 4
mat[4,4]=1
#rest 5
mat[5,2]=1
mat[5,6]=1
#rest 6
mat[6,4]=1
mat[6,5]=1
#rest 7
mat[7,3]=1
mat[7,5]=1
mat[7,6]=1

#rest 8
mat[8,4]=1
#rest 9
mat[9,3]=1
mat[9,4]=1
```

```
mat[9,5]=1
#rest 10
mat[10,3]=1
mat[10,6]=1
#rest 11
mat[11,5]=1
#rest 12
mat[12,6]=1
mat[12,7]=1
#rest 13
mat[13,7]=1
#rest 14
mat[14,6]=1
mat[14,7]=1
#rest 15
mat[15,7]=1
b=c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)
bin=c(1:7)


#vector de signos
signos=c('>=','>=','>=','>=','>=','>=','>=','>=','>=','>=','>=','>=','>=','>=','>=')
solucion=lp('min',f,mat,signos,b,binary.vec = bin)
solucion$objval

        ## [1] 15.45

        solucion$solution

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

La solución que nos proporciona Rstudio es que para minimizar el costo de construcción a 15.45 millones de euros , las variables que toman el valor 1 son: x_2 y x_3 x_4, x_5 y x_7. Lo que significa que construyendo el hospital 2, 3, 4, 5 y 7 se cubrirán todas las comunidades minimizando el coste.

El problema del viajante de comercio (TSP)

Un vendedor debe visitar cada una de las n ciudades exactamente una vez y luego regresar a su punto de partida. El tiempo que se tarda en viajar de la ciudad i a la ciudad j es c_{ij}. El objetivo de este problema es encontrar el orden en el que debe hacer su recorrido para terminar lo más rápido posible.

Definición de las variables.

$x_{ij} = 1$ si el vendedor va directamente del pueblo i al pueblo j y $x_{ij} = 0$, en caso contrario.

x_{ii} no está definido para i = 1,…, n., esto evita que el vendedor salga de la cuidad 1 y se dirija nuevamente a la cuidad 1.

Definición de las restricciones.

Se sale de la ciudad i exactamente una vez, de ahí que la suma total de las distintas ciudades a las que puede salir tome el valor de 1:

$$\sum_{j:j\neq i}^{n} x_{ij} = 1 \qquad \forall\, i = 1, \dots, n.$$

Llega a la ciudad j exactamente una vez, de ahí que la suma total de las distintas cuidades que puede llegar tome el valor de 1:

$$\sum_{i:i\neq j}^{n} x_{ij} = 1 \qquad \forall\, i = 1, \dots, n$$

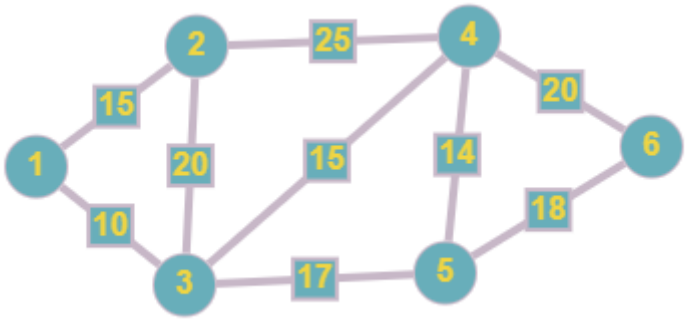
Definición de la función objetivo.

El tiempo total de viaje se minimiza:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

Ejemplo sin restricciones de ciclo

El vendedor debe visitar 6 ciudades exactamente una vez cada una, regresando por último al punto partida. El tiempo que se tarda en viajar de la ciudad i a la ciudad j es c_{ij}. El objetivo de este problema es encontrar el orden en el que debe hacer su recorrido para terminar lo más rápido posible. A continuación, el siguiente grafo representa las distintas conexiones que existen entre las ciudades y los costes de estas conexiones.



A continuación tenemos los tiempos en minutos de los posibles recorridos que puede hacer el vendedor.

Tabla 5

i *Representa el Origen y* ***j*** *el Destino del Vendedor*

		j					
		1	2	3	4	5	6
i	1		15	10			
	2	15		20	25		
	3	10	20		15	17	
	4		25	15		14	20
	5			17	14		18
	6				20	18	

Fuente: Elaboración propia

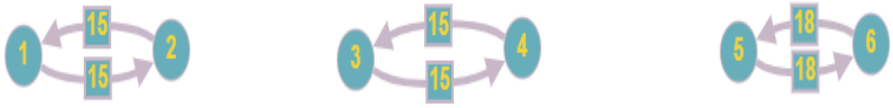
```
library('lpSolve')
#función objetivo
f=c(15,10,
   15,20,25,
```

```
   10,20,15,17,
   25,15,14,20,
   17,14,18,
   20,18)
#matriz de restricciones
mat=matrix(c(0), nrow = 12, ncol=18)
#rest 1
mat[1,1]=1
mat[1,2]=1
#rest 2
mat[2,3]=1
mat[2,4]=1
mat[2,5]=1
#rest 3
mat[3,6]=1
mat[3,7]=1
mat[3,8]=1
mat[3,9]=1
#rest4
mat[4,10]=1
mat[4,11]=1
mat[4,12]=1
mat[4,13]=1
#rest5
mat[5,14]=1
mat[5,15]=1
mat[5,16]=1


#rest6
mat[6,17]=1
mat[6,18]=1
#rest7
mat[7,3]=1
```

```
mat[7,6]=1
#rest8
mat[8,1]=1
mat[8,7]=1
mat[8,10]=1
#rest9
mat[9,2]=1
mat[9,4]=1
mat[9,11]=1
mat[9,14]=1
#rest10
mat[10,5]=1
mat[10,8]=1
mat[10,15]=1
mat[10,17]=1
#rest11
mat[11,9]=1
mat[11,12]=1
mat[11,18]=1

#rest12
mat[12,13]=1
mat[12,16]=1
signos=c('=','=','=','=','=','=','=','=','=','=','=','=')
b=c(1,1,1,1,1,1,1,1,1,1,1,1)
bin=c(1:12)
solucion=lp('min',f,mat,signos,b,binary.vec = bin)
solucion$objval

## [1] 96

solucion$solution

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

La solución que nos proporciona Rstudio es que para minimizar el tiempo de reparto en 96 minutos, las variables que toman el valor 1 son: x_{12} , x_{21} , x_{34} , x_{43} , x_{56} x_{65}.

Hasta ahora, las restricciones introducidas son precisamente las restricciones del problema de asignación. La solución del problema de la asignación da un conjunto de ciclos desconectados, tal y como se muestra en la siguiente imagen.



Para eliminar estas soluciones, necesitamos más restricciones que garantizan la conectividad al imponer que el vendedor debe pasar de un conjunto de ciudades a otro, las llamadas restricciones de conjunto de corte o rotura de ciclos:

El problema del viajante de comercio (TSP) con la restricción de ciclos.

Restricciones de rotura de ciclos. Para cada combinación de conjuntos de ciclos generaremos una restricción, de manera que ese ciclo se rompa.

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S|-1 \quad \forall\ S \subset N, \quad 2 \leq |S| \leq n-1.$$

Las variables son 0–1:

$x_{ij} \in \{0, 1\}$ $\forall$ i=1,…,n, j = 1,…, n, i ≠j.

Definición de la función objetivo.

El tiempo total de viaje se minimiza:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

Ejemplo con restricción de ciclos

Continuamos con el ejemplo anterior pero esta vez añadimos las restricciones necesarias para evitar la formulación de ciclos que es lo que ha pasado en el ejemplo anterior.

Un ejemplo de estas nuevas restricciones seria que la suma de estas dos variables sea 1 (x_{12}+ x_{21} = 1). Obligamos así a romper este ciclo formado en el ejercicio anterior.

```
        library('lpSolve')
#función objetivo
f=c(15,10,
   15,20,25,
   10,20,15,17,
   25,15,14,20,
   17,14,18,
   20,18)
#matriz de restricciones
mat=matrix(c(0), nrow = 30, ncol=18)
#rest 1
mat[1,1]=1
mat[1,2]=1
#rest 2
mat[2,3]=1
mat[2,4]=1
mat[2,5]=1
#rest 3
mat[3,6]=1
mat[3,7]=1
mat[3,8]=1
mat[3,9]=1
#rest4
mat[4,10]=1
mat[4,11]=1
mat[4,12]=1
mat[4,13]=1
```

```
#rest5
mat[5,14]=1
mat[5,15]=1
mat[5,16]=1
#rest6
mat[6,17]=1
mat[6,18]=1
#rest7
mat[7,3]=1
mat[7,6]=1
#rest8
mat[8,1]=1
mat[8,7]=1
mat[8,10]=1
#rest9
mat[9,2]=1
mat[9,4]=1
mat[9,11]=1
mat[9,14]=1
#rest10
mat[10,5]=1
mat[10,8]=1
mat[10,15]=1
mat[10,17]=1
#rest11
mat[11,9]=1
mat[11,12]=1
mat[11,18]=1
#rest12
mat[12,13]=1
mat[12,16]=1
#rest13
mat[13,1]=1
mat[13,4]=1
```

```
mat[13,6]=1
#rest14
mat[14,3]=1
mat[14,2]=1
mat[14,7]=1
#rest15
mat[15,4]=1
mat[15,8]=1
mat[15,10]=1
#rest16
mat[16,7]=1
mat[16,5]=1
mat[16,11]=1
#rest17
mat[17,8]=1
mat[17,12]=1
mat[17,14]=1
#rest18
mat[18,11]=1
mat[18,9]=1
mat[18,15]=1
#rest19
mat[19,12]=1
mat[19,16]=1
mat[19,17]=1
#rest20
mat[20,15]=1
mat[20,13]=1
mat[20,18]=1
#rest21
mat[21,4]=1
mat[21,9]=1
mat[21,15]=1
mat[21,10]=1
```

```
#rest22
mat[22,7]=1
mat[22,5]=1
mat[22,12]=1
mat[22,14]=1
#rest23
mat[23,1]=1
mat[23,3]=1
#rest24
mat[24,2]=1
mat[24,6]=1
#rest25
mat[25,4]=1
mat[25,7]=1
#rest26
mat[26,5]=1
mat[26,10]=1
#rest27
mat[27,8]=1
mat[27,11]=1
#rest28
mat[28,12]=1
mat[28,15]=1
#rest29
mat[29,13]=1
mat[29,17]=1
#rest30
mat[30,16]=1
mat[30,18]=1
signos=c('=','=','=','=','=','=','=','=','=','=','=','=','=','<=','<=','<=','<=','<=','<=','<=','<=','<=','<=','<=','<=','<=','<=','<=','<=','<=')
b=c(1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,3,3,1,1,1,1,1,1,1)
bin=c(1:18)
```

```
solucion=lp('min',f,mat,signos,b,binary.vec = bin)
solucion$objval

        ## [1] 105

        solucion$solution

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

La solución que nos proporciona Rstudio es que, para minimizar el tiempo de reparto en 105 minutos, las variables que toman el valor 1 son: x_{12}, x_{24} , x_{46} , x_{65} , x_{53} x_{31} . Lo que significa que el repartidor empezará el recorrido desde la cuidad 1, seguidamente irá a la cuidad 2, a continuación a la 4, después a la 6, continuará hacia la 5, seguidamente a la 3 y finalmente volverá a su punto de partida que es la cuidad 1. Como podemos observar en este caso, gracias a las restricciones nuevas hemos conseguido que no se formen más ciclos, esta vez el viajante recorre todas las cuidades y vuelve al punto de partida.

El problema de la localización de servicios sin capacidades (UFL)

Tenemos un conjunto de ubicaciones potenciales para nuevas instalaciones o servicios N = {1,…,n} y un conjunto M = {1,…,m} de clientes, suponemos que existe un coste fijo f_j asociado al uso del servicio j, y un costo del transporte c_{ij} si todo el pedido del cliente i se entrega desde el depósito j. El problema consiste en decidir qué servicios o instalaciones abrir y qué instalación sirve a cada cliente para minimizar la suma de los costos fijos y de transporte.

Definición de las variables

y_j $j \in N$ esta variable binaria tomará el valor 1 si se abre el servicio j, y y_j=0 en caso contrario.

Por otro lado, tenemos x_{ij} que tomara el valor 1 si al cliente i se le asigna el servicio j, y $x_{ij} = 0$ en caso contrario.

Definición de las restricciones

Satisfacción de la demanda del cliente i:

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall\, i = 1, \dots, m.$$

Cada cliente i será satisfecho por un servicio j

$$x_{ij} \leq y_j \qquad \forall\, i = 1, \dots, m \qquad \forall\, j = 1, \dots, n$$

Las variables son binarias 0–1:

$x_{ij} \in \{0, 1\}$ $\forall$ i=1,…,m $\forall$ j = 1,…, n.

$y_j \in \{0, 1\}$ $\forall$ j=1,…,n

Definición de la función objetivo.

El objetivo es $\sum_{j \in N} h_j\,(x_{1j}, \dots, x_{mj})$, donde $h_j(x_{1j}, \dots, x_{mj}) = f_j + \sum_{i \in M} c_{ij}\, x_{ij}$ si $\sum_{i \in M} x_{ij} > 0$, por lo que obtenemos:

$$min \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} + \sum_{j \in N} f_j y_j$$

Querremos que la suma de los costes de asociar cada cliente a un servicio y de abrir los servicios sean mínimos

Ejemplo

Tenemos 3 fábricas que deberán satisfacer a 4 clientes potenciales. Necesitamos saber qué fábricas nos conviene abrir y a qué clientes abastecer minimizando al máximo tanto el coste fijo como el variable.

El coste fijo de la apertura de las 3 fábricas son los siguientes:

Tabla 6

Coste Fijo asociado a cada Fábrica

Fábricas	Coste Fijo
F1	2500 €
F2	2000 €
F3	2000 €

Fuente: Elaboración propia

Los costes variables de asignar un cliente a una fábrica se presentan a continuación:

Tabla 7

Costes Variables de Asignación

		j (fábricas)		
		1	2	3
i (clientes)	A	115 €	110 €	130 €
	B	110 €	120 €	105 €
	C	102 €	150 €	140 €
	D	130 €	110 €	120 €

Fuente: Elaboración propia

```
library('lpSolve')

#función objetivo
f=c(115, 110, 102, 130,
   110, 120, 150, 110,
   130, 105, 140, 120, 2500,2000, 2000)
#matriz de restricciones
mat=matrix(c(0), nrow = 16, ncol=15)
#rest 1
mat[1,1]=1
```

```
mat[1,5]=1
mat[1,9]=1
#rest 2
mat[2,2]=1
mat[2,6]=1
mat[2,10]=1
#rest 3
mat[3,3]=1
mat[3,7]=1
mat[3,11]=1

#rest 4
mat[4,4]=1
mat[4,8]=1
mat[4,12]=1
#rest 5
mat[5,1]=1
mat[5,13]=-1
#rest 6
mat[6,2]=1
mat[6,13]=-1
#rest 7
mat[7,3]=1
mat[7,13]=-1
#rest 8
mat[8,4]=1
mat[8,13]=-1
#rest 9
mat[9,5]=1
mat[9,14]=-1
#rest 10
mat[10,6]=1
mat[10,14]=-1
#rest 11
mat[11,7]=1
mat[11,14]=-1
#rest 12
```

```
mat[12,8]=1
mat[12,14]=-1
#rest 13
mat[13,9]=1
mat[13,15]=-1
#rest 14
mat[14,10]=1
mat[14,15]=-1
#rest 15
mat[15,11]=1
mat[15,15]=-1
#rest 16

mat[16,12]=1
mat[16,15]=-1
################################
b=c(1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0)
bin=c(1:15)
#vector de signos
signos=c('=','=','=','=','<=','<=','<=','<=','<=','<=','<=','<=','<=','<=','<=','<=')
solucion=lp('min',f,mat,signos,b,binary.vec = bin)
solucion$objval

## [1] 2490

solucion$solution

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

El resultado que nos proporciona Rstudio es que para minimizar el coste tanto fijo como variable, las variables que toman el valor 1 son: x_{12}, x_{22} , x_{32} , x_{42} y y_2. Lo que significa que se pondrá en funcionamiento la fábrica dos y será la única que será asignada a todos los clientes.

El problema del camino más corto

Es el problema que consiste en encontrar un camino entre los vértices o nodos, de tal manera que la suma de los pesos de las aristas que lo constituyen sea mínima.

Definición de las variables

Tenemos x_{ij} que tomara el valor 1 si el viajante va de un nodo i al nodo j, y $x_{ij} = 0$ en caso contrario.

Definición de las restricciones

Salimos del nodo 1. La suma de las conexiones que salen del nodo uno ha de tomar el valor de 1.

G=(V,A) V son los nodos y A las aristas.

$$\sum_{j:(1,j)\in A}^{n} x_{1j} = 1 \qquad i = 1$$

Llegamos finalmente al nodo 5 esto supone que la suma de las variables que llegan al nodo 5 tiene que ser 1.

$$\sum_{i:(i,5)\in A}^{n} x_{i5} = 1 \qquad j = 5$$

En esta última restricción se formará un camino, para cada nodo restante 2, 3 y 4 tomará valor de 1 la variable de partida como la variable de salida, indicando así que por ejemplo el viajante entrará al nodo 2 y enseguida saldrá del mismo, lo mismo pasará como en nodo 3 y 4.

$$\sum_{i:(i,k)\in A} x_{ik} = \sum_{j:(k,j)\in A} x_{kj} \qquad k = 2,3,4$$

Las variables son binarias 0–1:

$x_{ij} \in \{0, 1\}$ $\forall$ i=1,…,m $\forall$ j = 1,…, n.

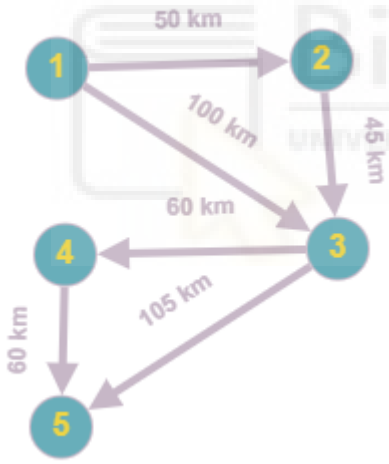
Definición de la función objetivo.

$$min \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij}$$

Queremos conseguir que la suma de los pesos de las aristas que lo constituyen sea mínima. Es decir, que el recorrido sea mínimo.

Ejemplo

Queremos encontrar el camino más corto para ir de una cuidad a otra. En este caso los vértices representarían las cuidades y las aristas las carreteras que las unen, cuya ponderación viene dada por los kilómetros que se realizan al atravesar las distintas cuidades hasta llegar a la cuidad de destino. Los datos lo tenemos a continuación.




```
library('lpSolve')
#función objetivo
f=c(50, 100,
   45,
   60,105,
   60)
#matriz de restricciones
mat=matrix(c(0), nrow = 5, ncol=6)
```

```
#rest 1 salimos del nodo 1
mat[1,1]=1
mat[1,2]=1
#rest 2 llegamos al nodo 5
mat[2,5]=1
mat[2,6]=1



#rest 3 se ha de formar un camino Nodo 2
mat[3,1]=1
mat[3,3]=-1
#rest 4 se ha de formar un camino Nodo 3
mat[4,2]=1
mat[4,3]=1
mat[4,4]=-1
mat[4,5]=-1
#rest 5 se ha de formar un camino Nodo 4
mat[5,4]=1
mat[5,6]=-1
b=c(1,1,0,0,0)
bin=c(1:6)
#vector de signos
signos=c('=','=','=','=','=')
solucion=lp('min',f,mat,signos,b,binary.vec = bin)
solucion$objval

## [1] 200

solucion$solution

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

La solución que nos proporciona el programa es que para realizar el trayecto del camino más corto en 200 km partiendo de la cuidad 1 y llegando a la cuidad 5. Las variables que toman el valor 1 son: x_{12}, x_{23}, y x_{35} lo que significa que el recorrido optimo es partir de la cuidad 1, dirigirse a la cuidad 2, a continuación ir a la cuidad 3 y finalmente terminar el recorrido en la cuidad 5.

Conclusión

Existen en la literatura más problemas clásicos de la investigación operativa, aquí solo hemos considerado algunos de entre los más conocidos.

A modo de cierre de este trabajo podemos señalar que hoy en día la Investigación de Operaciones destaca en el ámbito empresarial ya que nos permite plantear una resolución a muchos problemas operacionales, problemas tipo la distribución de recursos, disposición de medios, control de inventarios, calendarización de personal y sistemas de distribución.. etc. Para seguir profundizando en la Investigación Operativa lo que podríamos hacer es añadir ejemplos nuevos que pudiesen ocurrir dentro de una empresa siguiendo el modelo de los ejemplos propuestos en el trabajo, iniciando con una pequeña introducción del problema, formulándolos matemáticamente, definiendo sus restricciones y función objetivo e implementándolos y resolviéndolos en el programa Rstudio.

Referencias

Hillier, F., & Lieberman, G. (2010). Introducción a la Investigación de Operaciones. En F. Hillier, & G. Lieberman, *Introducción a la Investigación de Operaciones* (pág. 978). Lomas de Santa Fe: McGraw-Hill.

Malquin, S. (25 de 09 de 2022). *CAMPOS DE APLICACION INVESTIGACION OPERACIONES*. Obtenido de CAMPOS DE APLICACION INVESTIGACION OPERACIONES: https://issuu.com/soniamarilu/docs/2._campos_de_la_io

Salazar González, J. (2001). Programación Matemática. En J. J. Salazar González, *Programación Matemática* (pág. 438). Tenerife: Díaz de Santos.

Wolsey, L. (2021). Integer Programming. En L. Wolsey, *Integer Programming* (pág. 360). Nueva Jersey: Office.

Enlaces web

[1] https://www.youtube.com/watch?v=tvx7XqnGIlY

[2] https://www.youtube.com/watch?v=86DaXndBVI4

[3] https://www.youtube.com/watch?v=AIIBq4lm3bU