4. Problema de la ruta más corta

Un viaje de miles de millas empieza con un único paso y si ese paso es el adecuado, él deviene el último paso. (Lao Tsé).

Este problema también se conoce como el problema del camino mínimo y es atrayente, por aparecer frecuentemente en gran variedad de aplicaciones que tienen como objeto enviar material entre dos puntos de una red, tanto rápidamente, como barato y predictible posible. Además, son fáciles de resolver eficientemente.

Como casi todos los modelos de redes simples, capturan muchos de los más interesantes ingredientes del flujo en redes y proveen de un bagaje intelectual que ayuda a resolver otros modelos más complicados. Aparecen como subproblemas en otros problemas.

Ejemplo 4.1 *Encontrar los caminos más cortos que unen el nodo 1 con los restantes nodos, la matriz de incidencia es la siguiente:*

	Nodo 1	*Nodo 2*	*Nodo 3*	*Nodo 4*	*Nodo 5*	*Nodo 6*
Nodo 1		*6*	*4*			
Nodo 2			*2*	*2*		
Nodo 3				*1*	*2*	
Nodo 4						*7*
Nodo 5				*1*		*3*
Nodo 6						

Notesé que en este grafo los arcos son dirigidos.

4.1. Formulación en forma de programa lineal

Consideramos una red dirigida $G = (N, A)$ con c_{ij} la *longitud* (arc length) asociada a cada arco $(i, j) \in A$. La red tiene un nodo distinguible s, llamado *fuente* (source). Llamamos $A(i)$ a la *lista de nodos adyacentes* al nodo i y sea $C = \max\{c_{ij} : (i, j) \in A\}$ el mayor valor asociado a los arcos del conjunto de arcos A.

Definimos la *longitud de un camino dirigido* (length of a directed path) como la suma de las longitudes de los arcos en el camino. El problema de la ruta más corta consiste en determinar para cada nodo no fuente $i \in N$ el camino más corto desde el nodo s al nodo i.

Alternativamente, podemos ver el problema como enviar una unidad de flujo tan económicamente como sea posible desde el nodo s a cada uno de los nodos $i \in N \setminus \{s\}$ en una red sin capacidades.

La formulación de este problema en términos de programación lineal será:

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in A} c_{ij}x_{ij}, \\
& \sum_{(j;(i,j)\in A)} x_{ij} - \sum_{(j;(j,i)\in A)} x_{ji} = \begin{cases} n-1 & \text{para } i = s. \\ -1 & \text{para todo } i \in N\backslash\{s\}. \end{cases} \\
& x_{ij} \in \{0,1\} \qquad\qquad \text{para todo } (i,j) \in A.
\end{aligned}$$

4.2. Condiciones iniciales del problema

En el estudio del problema de la ruta más corta haremos las siguientes suposiciones:

1. La longitud de los arcos son números enteros.
2. La red contiene un camino directo desde el nodo s hasta cada uno de los nodos i.
3. La red no contiene nigún ciclo negativo. (i.e. ciclo dirigido de longitud negativa).
4. La red es dirigida.

La primera suposición es para facilitar algunos algoritmos de solución.

La segunda, se puede obviar añadiendo un arco ficticio (s, i) de un gran costo.

La tercera, evita que el problema sea no acotado o complicado de resolver. La última, nos dice que si la red fuese no dirigida y el coste de todos los arcos fuera nonegativo, podriamos transformar este problema en uno de redes dirigidas mediante alguna transformación.

4.3. Tipos de problemas

1. Encontrar el camino más corto de un nodo a otro cuando las longitudes de los arcos son no negativas.

2. Encontrar el camino más corto de un nodo a otro cuando las longitudes de los arcos son arbitrarias.

3. Encontrar el camino más corto desde cualquier nodo a otro nodo.

4.4. Algoritmos

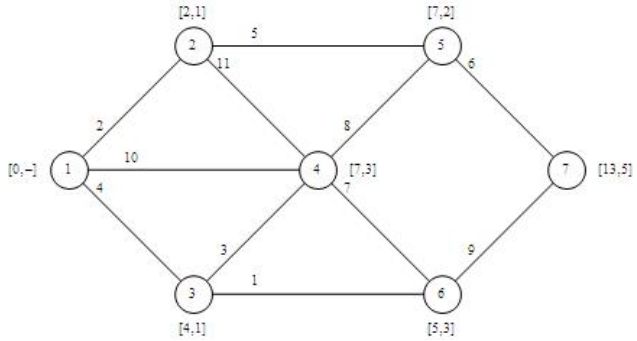
Se dice que una red es acíclica si no contiene lazos, de otra manera es cíclica. El algoritmo cíclico es más general ya que incluye el caso acíclico. El algoritmo acíclico es más eficiente porque se necesitan menos cálculos.

4.4.1. Algoritmo acíclico

En este cso se emplean cálculos recursivos que son la base para los cálculos de la programación dinámica. El algoritmo acíclico no funcionará de manera correcta si la red contiene bucles, en este caso usaremos un algoritmo cíclico.

4.4.2. Ejemplo

Dado el siguiente grafo



Encontrar el camino mínimo que une los nodos 1 y 7.

Notaciones

► Nodo 1 es el nodo fuente o inicio.
► Nodo 7 es el nodo sumidero o destino.
► $d_i(j)$ son las distancias entre el Nodo i y el Nodo j que se indican sobre cada rama (i.e. $d_1(2) = 2$).
► u_i es la distancia más corta entre el Nodo 1 y el Nodo i.
► $u_1 = 0$ por definición.

La ecuación

$$u_j = \min_i \{u_i + d_i(j)\}$$

da la distancia más corta al Nodo fuente desde un Nodo j. Lo hace comparando los valores resultantes de las sumas de u_i con $d_i(j)$ para todos los nodos predecesores al actual Nodo j, es decir, para todo $i \in A(j)$.

Procedimiento de etiquetado

Etiqueta Nodo $j = [u_j, n]$ donde n es el nodo que precede inmediatamente a j y que da la distancia más corta u_i

$$u_j = \min_i \{u_i + d_i(j)\} = u_n + d_n(j)$$

El Nodo 1 por definición tiene de etiqueta $[0, -]$ que nos indica que Nodo 1 es la fuente.

El algoritmo acíclico permite reevaluar un nodo. Cuando resulta evidente que se ha alcanzado la distancia más corta este nodo se excuye de cualquier consideración posterior (i.e. Nodo 2 tiene de etiqueta $[2, 1]$ donde el 1 nos dice que Nodo 1 es el nodo que precede inmediatamente al Nodo 2 y 2 es la distancia más corta).

Algoritmo

Primero se calculan las etiquetas y después la ruta óptima procediendo hacia atrás y utilizando la información de los nodos.

Nodo	Cálculo de u_i	Etiqueta
1	$u_1 = 0$	$[0, -]$
2	$u_2 = u_1 + d_1(2) = 0 + 2 = 2$	$[2, 1]$
3	$u_3 = u_1 + d_1(3) = 0 + 4 = 4$	$[4, 1]$
4	$u_4 = \text{mín}\{u_2 + d_2(4), u_1 + d_1(4), u_3 + d_3(4)\}$ $= \text{mín}\{2 + 11, 0 + 10, 4 + 3\} = 7$ desde 3	$[7, 3]$
5	$u_5 = \text{mín}\{u_2 + d_2(5), u_4 + d_4(5)\}$ $= \text{mín}\{2 + 5, 7 + 5\} = 7$ desde 2	$[7, 2]$
6	$u_6 = \text{mín}\{u_4 + d_4(6), u_3 + d_3(6)\}$ $= \text{mín}\{7 + 7, 4 + 1\} = 5$ desde 3	$[5, 3]$
7	$u_7 = \text{mín}\{u_5 + d_5(7), u_6 + d_6(7)\}$ $= \text{mín}\{7 + 6, 5 + 9\} = 13$ desde 5	$[13, 5]$

La solución es: $7 \to [13, 5] \to [7, 2] \to [2, 1] \to 1$.
Resumiendo: Nodo 7 → Nodo 5 → Nodo 2 → Nodo 1, valor 13.

Variaciones:

Como práctica, podemos hacer el mismo ejercicio anterior con:
a) Conectamos Nodo 4 con Nodo 7 con valor 5.
b) Conectamos Nodo 5 con Nodo 6 con valor 2.
c) Haciendo a) y b) a la vez.

4.5. Algoritmo cíclico. Algoritmo de Dijkstra

Un algoritmo diseñado para encontrar las rutas más cortas entre el nudo origen y cada uno de los nodos de la red es el de Dijkstra (1959-2002).

Considere una red conexa y dirigida con dos nodos especiales llamados origen y destino. A cada ligadura se asocia una distancia no negativa. El objetivo es encontrar la ruta más corta (la trayectoria con la mínima distancia total) del origen al destino.

Se dispone de un algoritmo bastante sencillo para este problema. La esencia del procedimiento es que analiza toda la red a partir del origen; identifica de manera sucesiva la ruta más corta a cada uno de los nodos en orden ascendente de sus distancias (más cortas), desde el origen. El problema queda resuelto en el momento de llegar al nodo destino.

4.5.1. Algoritmo de la ruta más corta

Paso 1. Objetivo de la n-ésima iteración: encontrar el n-ésimo nodo más cercano al origen.

(Este paso se repetirá para n=1,2,... hasta que el n-ésimo nodo más cercano sea el nodo destino)

Paso 2. Datos para la n-ésima iteración: n-1 nodos más cercanos al origen (encontrados en las iteraciones previas), incluida su ruta más corta y la distancia desde el origen.

(Estos nodos y el origen se llaman nodos resueltos, el resto son nodos no resueltos.)

Paso 3. Candidatos para el n-ésimo nodo más cercano: Cada nodo resuelto que tiene conexión directa por una ligadura con uno o más nodos no resueltos proporciona un candidato, y éste es el nodo no resuelto que tiene la ligadura más corta.

(Los empates proporcionan candidatos adicionales.)

Paso 4. Cálculo del n-ésimo nodo más cercano: para cada nodo resuelto y sus candidatos, se suma la distancia entre ellos y la distancia de la ruta más corta desde el origen a este nodo resuelto. El candidato con la distancia total más pequeña es el n-ésimo nodo más cercano (los empates proporcionan nodos resueltos adicionales), y su ruta más corta es la que genera esta distancia.

4.5.2. Construcción del algoritmo

Vamos a construir ese algoritmo recordando que una de las características de este algoritmo es la utilización de **etiquetas** en cada nodo, cuya función es *indicar en cada iteración la distancia entre dicho nodo y el origen*. Una de las etiquetas será permanente, e indicará la distancia minimal del nodo inicial a dicho nodo.

Consideramos I nodo inicial y F nodo final. Denotamos por $d_k(i)$ la etiqueta del nodo i que indica la distancia del nodo i al inicial en la iteración k-ésima.

Iteración 0:

Etiquetamos $d_0(I) = 0$ al nodo inicial permanentemente y $d_0(i) = \infty$ todos los demás. La variable p recoge el último nodo etiquetado de forma permanente, en el primer paso $p = I$.

Iteración 1:

Para todos los nodos j conectados con el nodo p, es decir, para los que existe el arco (p, j), calculamos su etiqueta temporal de la iteración k-ésima

$$d_k(j) = \text{mín}\left\{d_{k-1}(j), d_{k-1}(p) + a(p, j)\right\},$$

Observamos que los nodos no conectados conservan la etiqueta de la iteración anterior

$$d_k(i) = d_{k-1}(i).$$

Iteración 2:

Para todos los nodos que no tienen etiqueta permanente, encontramos el de menor etiqueta que pasará a permanente y este nodo pasará a ser nodo p .

$$d_k(p) = \min_j \left\{d_k(j)\right\}.$$

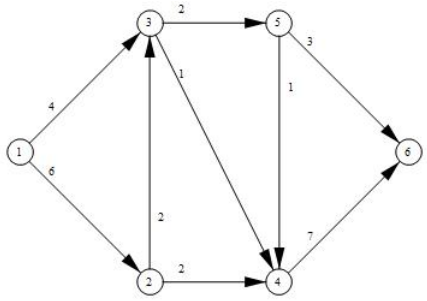
Volvemos a la iteración 1 hasta que todos los nodos tengan etiquetas permanentes.

Nota: Todos los datos de las iteraciones se recogerán en una tabla que nos indicará el estado del proceso y como se conectan los nodos.

4.5.3. Solución del ejemplo

Ahora Aplicamos Dijkstra para encontrar la solución del ejemplo 4.1.

El grafo dirigido que nos representa el problema es:



Aplicando el algoritmo resulta:

Iteración 0: k=0 donde k es el número de la iteración. El nodo inicial es $I = 1$, el nodo final es $F = 6$.

Etiquetas: $d_0(I) = d_0(1) = 0$ y $d_0(i) = \infty$ para todo $i \in N \backslash \{1\}$.

$EP = \{1\}$ conjunto de nodos con etiqueta permanente.

$NEP = \{2, 3, 4, 5, 6\}$ conjunto de nodos con etiqueta no permanente.

Iteración 1: $k = 1$.

Etiquetas permanentes: $d_0(1) = 0$.

Nodos conectados con el nodo 1: $\{2, 3\}$. Calculamos sus etiquetas temporales:

$d_1(2) = \text{mín}\{d_0(2), d_0(1) + a(1,2)\} = \text{mín}\{\infty, 0 + 6\} = 6$.

$d_1(3) = \text{mín}\{d_0(3), d_0(1) + a(1,3)\} = \text{mín}\{\infty, 0 + 4\} = 4$.

Los nodos no conectados son: $\{4, 5, 6\}$.

$d_1(i) = \infty$ para todo $i = 4, 5, 6$.

Iteración 2:

Seleccionando nodo:

$\text{mín}\{d_1(2), d_1(3), d_1(4), d_1(5), d_1(6)\} = \{6, 4, \infty, \infty, \infty\} = 4$.

Por consiguiente el nodo seleccionado es $p = 3$.

$EP = \{1, 3\}$ conjunto de nodos con etiqueta permanente.

$NEP = \{2, 4, 5, 6\}$ conjunto de nodos con etiqueta no permanente.

▲ Como $NEP \neq \{\varnothing\}$, continuamos con el algoritmo.

Iteración 1: $k = 2$.

Etiquetas permanentes: $d_1(1) = 0$, $d_1(3) = 4$.

Los nodos conectados con el nodo 3 son : $\{4, 5\}$.

$d_2(4) = \text{mín}\{d_1(4), d_1(3) + a(3,4)\} = \text{mín}\{\infty, 4 + 1\} = 5$.

$d_2(5) = \text{mín}\{d_1(5), d_1(3) + a(3,5)\} = \text{mín}\{\infty, 4+2\} = 6.$

Los nodos no conectados son: $\{2, 6\}$.

$d_1(2) = d_2(2) = 6$ y $d_2(6) = \infty$.

Iteración 2:

Nodos no seleccionados con etiqueta no permanente:

$\text{mín}\{d_2(2), d_2(4), d_2(5), d_2(6)\} = \{6, 5, 6, \infty\} = 5.$

Por consiguiente el nodo seleccionado es $p = 4$.

$EP = \{1, 3, 4\}$ conjunto de nodos con etiqueta permanente.

$NEP = \{2, 5, 6\}$ conjunto de nodos con etiqueta no permanente.

▲ Como $NEP \neq \{\varnothing\}$, continuamos con el algoritmo.

Iteración 1: $k = 3$.

Etiquetas permanentes: $d_2(1) = 0$, $d_2(3) = 4$, $d_2(4) = 5$.

Los nodos conectados con el nodo 4 son: $\{6\}$.

$d_3(6) = \text{mín}\{d_2(6), d_2(4) + a(4,6)\} = \text{mín}\{\infty, 5+7\} = 12.$

Los nodos no conectados son: $\{2, 5\}$.

$d_1(2) = d_2(2) = 6$ y $d_2(5) = 6$.

Iteración 2:

Nodos no seleccionados con etiqueta no permanente:

$\text{mín}\{d_3(2), d_3(5), d_3(6)\} = \{6, 6, 12\} = 6.$

En esta iteración podemos tomar como nodos permanentes los nodos 2 y 5. Nosotros, arbitrariamente tomamos el nodo 2.

Por consiguiente el nodo seleccionado es $p = 2$.

$EP = \{1, 2, 3, 4\}$ conjunto de nodos con etiqueta permanente.

$NEP = \{5, 6\}$ conjunto de nodos con etiqueta no permanente.

▲ Como $NEP \neq \{\varnothing\}$, continuamos con el algoritmo.

Iteración 1: $k = 4$.

Etiquetas permanentes: $d_2(1) = 0$, $d_3(2) = 6$, $d_2(3) = 4$, $d_2(4) = 5$.

Los nodos conectados con el 2 son todos con etiquetas permanentes, por consiguiente, es un camino cerrado.

Iteración 2:

$EP = \{1, 2, 3, 4\}$ conjunto de nodos con etiqueta permanente.

$NEP = \{5, 6\}$ conjunto de nodos con etiqueta no permanente.

▲ Como $NEP \neq \{\varnothing\}$, continuamos con el algoritmo.

Iteración 1: $k = 5$.

El nodo seleccionado es $p = 5$.

Etiquetas permanentes: $d_2(1) = 0$, $d_3(2) = 6$, $d_2(3) = 4, d_2(4) = 5$, $d_4(5) = 6$.

Los nodos conectados con el nodo 5 son: $\{6\}$.

$d_4(6) = \text{mín}\{d_3(6), d_3(5) + a(5,6)\} = \text{mín}\{12, 6+3\} = 9.$

Iteración 2:

Nodos no seleccionados con etiqueta no permanente:

$\text{mín}\{d_4(6)\} = \{9\} = 9.$

Por consiguiente el nodo seleccionado es $p = 6$.

$EP = \{1, 2, 3, 4, 5, 6\}$ conjunto de nodos con etiqueta permanente.
$NEP = \{\varnothing\}$ conjunto de nodos con etiqueta no permanente.

▲ Como $NEP = \{\varnothing\}$, finalizamos con el algoritmo.
Todos los nodos tienen etiqueta permanente

$$d_2(1) = 0, d_2(2) = 6, d_2(3) = 3, d_2(4) = 4, d_2(5) = 5, d_5(6) = 9.$$

4.5.4. Tablas resumen

La tabla resumen de los resultados es:

	p	$d(1)$	$d(2)$	$d(3)$	$d(4)$	$d(5)$	$d(6)$
	init	0	∞	∞	∞	∞	∞
ite 1	1	(per)	6	4			
ite 2	3			(per)	5	6	
ite 3	4				(per)		12
ite 4	2		(per)				
ite 5	5					(per)	9
ite 6	6						(per)
	fin	0	6	4	5	6	9

Donde ite significa iteración y per significa permanente.

La tabla de conexiones es:

	p	$d(1)$	$d(2)$	$d(3)$	$d(4)$	$d(5)$	$d(6)$
	init						
iteración 1	1		1	1			
iteración 2	3				3	3	
iteración 3	4						
iteración 4	2						
iteración 5	5						5
iteración 6	6						
FINAL		-	1	1	3	3	5

La tabla nos indica que: Nodos 2 y 3 unidos al nodo 1. Nodos 4 y 5 unidos al nodo 3. Nodo 6 unido al nodo 5.

Los dos cuadros nos dan la completa la solución del problema de la siguiente manera:

Comenzamos con la segunda tabla, tomamos $p = 1$, es decir comenzamos por el Nodo 1. Este Nodo se conecta con los Nodos 2 y 3, en la primera tabla el valor mínimo corresponde al Nodos 3 que es el que elegimos.

Ahora, hacemos $p = 3$, este Nodo se conecta con los Nodos 4 y 5, en la primera tabla el valor mínimo corresponde al Nodos 4 que es el que elegimos.

Hacemos $p = 4$, este Nodo se conecta con el Nodo 6 y se acaba el proceso.

Solución: $1 \to 3 \to 4 \to 6$.

4.6. Problema de la mochila

Una motivación para la aplicación de encontrar el camino más largo puede ser el problema siguiente:

Un excursionista debe decidir cuales de entre p objetos incluye en su mochila para un próximo viaje, teniendo en cuenta que cada uno tiene un peso w_i (en kilos) y una utilidad u_i para él.

El objetivo es maximizar la utilidad del excursionista sujeta al peso que él puede llevar que es de w kilos.

Formularemos el problema de encontrar el camino más largo sobre una red aciclica y, posteriormente, lo trasformaremos en un problema de encontrar el camino mínimo para proceder a su resolución con los métodos estudiados. Este esquema lo realizaremos mediante el ejemplo siguiente:

4.6.1. Ejemplo

Sea un excursionista que dispone de 4 objetos y tiene una mochila on una limitación de peso de $w = 6$ kilos. Los datos de los objetos vienen dados en la siguiente tabla

j	1	2	3	4
u_j	40	15	20	10
w_j	4	2	3	1

Para resolver el problema y como técnica general consideraremos el problema dividido en capas (una por cada objeto mas dos, una por el nodo inicial y otra por el nodo final), además identificaremos *escenarios* que están relacionados con el peso de la mochila (uno por cada kilo de peso que la mochila puede llevar mas el escenario inicial). Notese, que en general, el número de escenarios es el mínimo común múltiplo de las unidades que nos dan los pesos de los objetos mas uno.

En nuestro caso tendremos 6 capas numeradas del 0 al 5 y 7 escenarios. Para dibujar el digrafo correspondiente tendremos que tener en cuenta lo siguiente:

1. Cada capa correspondiente al objeto i tiene 7 nodos $(w+1)$ notados como i^0,i^1,i^2,i^3,i^4,i^5 y i^6 (i^k donde $k = 0, 1, 2, 3, \cdots, w$).

2. El nombre i^k de cada nodo de la red significa que el objeto i aporta k kilos a la mochila.

3. Cada nodo tiene al menos dos arcos salientes correspondientes a las dos decisiones; incluimos el objeto i en la mochila y no incluimos el objeto i en la mochila.

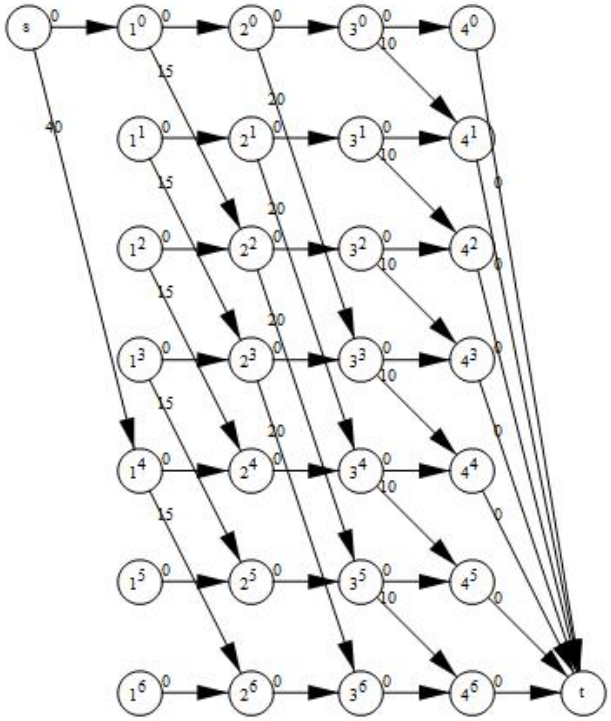
4. Sólo incluiremos el objeto en la mochila si ésta tiene capacidad suficiente.

5. Los arcos que salen de cada nodo o valen 0 (no incluimos los objetos) o valen la utilidad del objeto.

6. Los objetos no se pueden dividir en piezas más pequeñas.

7. Cada escenario acaba en el nodo terminal con un arca de utilidad 0.

La red resultante es:



Correspondiendo a un problema de encontrar el camino más largo.

Podemos transformar el problema de encontrar el camino más largo a un problema de encontrar el camino más corto definiendo los arcos de coste iguales al negativo de los arcos de utilidad.

Notamos que el grafo correspondientea la formulación del problema del camino más largo es aciclico, así, en el problema de camino más corto ser soluble.

La solución de nuestro problema es el camino entre los nodos s, i^4, 2^6, 3^6, 4^6 y t. Los objetos que se escojen son el primero con 4 kilos y 40 de utilidad y el segundo con 2 kilos y 15 de utilidad.

Notese que si el problema de encontrar el camino más largo contiene algún ciclo, el problema de encontrar el camino más corto contiene un ciclo negativo y no lo podremos resolver con las técnicas estudiadas en este curso.

4.7. Formulación como problema de optimización lineal

Un excursionista debe decidir cuales de entre p objetos incluye en su mochila para un próximo viaje, teniendo en cuenta que cada uno tiene un peso w_i (

en kilos) y una utilidad u_i para él. El objetivo es maximizar la utilidad del excursionista sujeta al peso que él puede llevar que es de w kilos. Es un problema de programación entera.

Si definimos x_i de tal forma que $x_i = \{0, 1\}$ para todo i número de objetos que se pueden transportar, tenemos

$$\begin{array}{ll} \text{máx} & \sum_{i=1}^{p} u_i x_i, \\ \text{s.a:} & \sum_{i=1}^{p} w_i x_i \leq w, \\ & x_i = \{0, 1\} \text{ para todo } i. \end{array}$$

4.8. Ejemplo

Un transportista dispone de un tren con capacidad de hasta 700 toneladas. El tren transporta contenedores de diferentes pesos para una determinada ruta. El tren puede transportar alguno de los siguientes contenedores

Contenedor	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
Peso	100	155	50	112	70	80	60	118	110	55

El analista de la empresa de transporte ha de determinar el envío que maximiza la carga transportada.

Las variables de decisión son:

$$x_i = \begin{cases} 1 & \text{si el contenedor } i \text{ se carga,} \\ 0 & \text{si no se carga.} \end{cases}$$

El objetivo es maximizar la carga que llevará el tren:

$$\text{máx}\, 100x_1 + 155x_2 + 50x_3 + 112x_4 + 70x_5 + 80x_6 + 60x_7 + 118x_8 + 110x_9 + 55x_{10}.$$

y la restricción es que el peso de la carga transportada no pueda exceder de la capacidad del tren:

$$100x_1 + 155x_2 + 50x_3 + 112x_4 + 70x_5 + 80x_6 + 60x_7 + 118x_8 + 110x_9 + 55x_{10} \leq 700.$$

Tén gase en cuenta que $a_i = ci$ para todo i, ya que la utilidad en este caso es el peso.

La decisión óptima es transportar los contenedores

(c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9, c_{10})=(Si, No, Si, Si, Si, Si, Si, Si, Si, No)

El valor óptimo es 700, lo que indica que el tren está completo.