Capítulo 5
Programación lineal entera

1. Introducción.

Un problema de programación lineal entera es un problema de programación lineal con la restricción adicional de que algunas de las variables deben tomar valores enteros. Cuando todas las variables deben tomar valores enteros decimos que se trata de un problema de programación lineal entera *puro*, en caso contrario decimos que es *mixto*. Diremos que una variable es *binaria* si sólo puede tomar los valores 0 y 1.
Una gran variedad de problemas combinatorios pueden ser planteados como problemas de programación lineal entera. Veamos algunos ejemplos.

Ejemplo 1.1. El problema de la mochila.

Se desea cargar en una mochila x_j unidades del producto j $(1 \leq j \leq n)$. Supongamos que cada unidad del producto j tiene asignado un peso p_j y un valor v_j. Queremos determinar la carga de máximo valor con la condición de que el peso total de la carga de la mochila no supere un dado peso P, es decir, queremos resolver el problema

$$\begin{aligned} &\max \sum_j v_j x_j \\ &\sum_j p_j x_j \leq P \\ &x_j \geq 0, \quad x_j \text{ entero} \end{aligned}$$

Si además agregamos la condición de que todas las variables x_j sean binarias esto resuelve el problema de elegir un subconjunto de productos con máximo valor cuyo peso total no supere P.

Ejemplo 1.2. El problema de la carga fija.

Un centro industrial posee una usina con tres generadores. Supongamos que cada generador i puede operar a niveles x_i de kw/h, $0 \leq x_i \leq a_i$ y que conocemos una función $y_i = y_i(x_i)$ que calcula la cantidad de kilogramos de vapor por hora consumidos en función del nivel x_i de kw/h al que opera el generador i. Supongamos además que la función y_i es lineal en el intervalo $(0, a_i]$ y tiene una discontinuidad en el origen, donde vale cero, es decir, que

$$y_i(x_i) = \begin{cases} c_i x_i + b_i & \text{si } x_i > 0 \\ 0 & \text{si } x_i = 0 \end{cases}$$

Esta discontinuidad se debe a que cuando el generador no está funcionando $x_i = 0 = y_i(x_i)$ pero para comenzar a funcionar necesita una cantidad fija b_i de vapor para que se ponga en marcha la turbina.
Sea C la cantidad de kw/h requeridos en un momento dado. Queremos determinar cuál es el nivel x_i al que debe operarse cada generador i para producir los kw/h requeridos de modo que la cantidad total de consumo de vapor sea mínima, es decir, queremos resolver el problema

$$\begin{aligned} &\min \sum_i y_i(x_i) \\ &x_1 + x_2 + x_3 = C \\ &0 \leq x_i \leq a_i \end{aligned}$$

Notemos que este no es un problema de programación lineal ya que el funcional no es una función lineal. Para resolverlo planteamos un problema auxiliar introduciendo tres variables binarias δ_i que valdrán 1 si el generador i está funcionando y 0 si no. Dejamos a cargo del lector verificar que si (x, δ) es una solución óptima del problema auxiliar

$$\min \sum_i c_i x_i + b_i \delta_i$$
$$x_1 + x_2 + x_3 = C$$
$$0 \leq x_i \leq a_i \delta_i$$
$$0 \leq \delta_i \leq 1, \quad \delta_i \text{ entero}$$

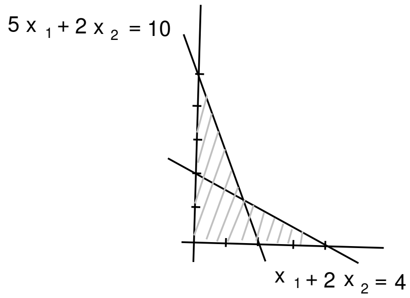
entonces x es una solución óptima del problema original.

Ejemplo 1.3. Condición *either ... or.*

Consideremos el problema

$$\begin{aligned} &\max x_1 + x_2 \\ &5x_1 + 2x_2 \leq 10 \text{ o } x_1 + 2x_2 \leq 4 \qquad (1) \\ &x_i \geq 0 \end{aligned}$$

Observemos que este no es un problema de programación lineal ya que el conjunto de soluciones factibles no es convexo:



Sin embargo, podemos convertirlo en un problema de programación lineal entera introduciendo una variable binaria δ y una constante c adecuada.
Veamos cómo hacer esto: sea c una constante que satisface

$$5x_1 + 2x_2 \leq 10 \Longrightarrow x_1 + 2x_2 \leq 4 + c$$
$$x_1 + 2x_2 \leq 4 \Longrightarrow 5x_1 + 2x_2 \leq 10 + c$$

para todo $x_1, x_2 \geq 0$. Por ejemplo, en este caso podemos tomar $c = 15$ ya que dados $x_1, x_2 \geq 0$, si $5x_1 + 2x_2 \leq 10$ entonces $x_1 \leq 2$ y $x_2 \leq 5$ de donde $x_1 + 2x_2 \leq 12 \leq 4 + 15$ y si $x_1 + 2x_2 \leq 4$ entonces $x_1 \leq 4$ y $x_2 \leq 2$ de donde $5x_1 + 2x_2 \leq 24 \leq 10 + 15$.
Ahora consideramos el problema de programación lineal entera

$$\begin{aligned} &\max x_1 + x_2 \\ &5x_1 + 2x_2 \leq 10 + c\delta \\ &x_1 + 2x_2 \leq 4 + c(1 - \delta) \qquad (2) \\ &x_i \geq 0 \\ &0 \leq \delta \leq 1, \quad \delta \text{ entero} \end{aligned}$$

Si $(\overline{x}_1, \overline{x}_2, \delta)$ es una solución óptima de (2), como $\delta = 0$ o 1 entonces se tiene que $(\overline{x}_1, \overline{x}_2)$ es una solución óptima de (1). En efecto, si $\delta = 0$ entonces $5\overline{x}_1 + 2\overline{x}_2 \leq 10$ y si $\delta = 1$ entonces $\overline{x}_1 + 2\overline{x}_2 \leq 4$, de donde resulta que $(\overline{x}_1, \overline{x}_2)$ es una solución factible de (1). Veamos que es óptima. Sea (x_1, x_2) una solución factible de (1). Por la forma en la que elegimos c se tiene

i) si $5x_1 + 2x_2 \leq 10$ entonces $x_1 + 2x_2 \leq 4 + c$. Luego, $(x_1, x_2, 0)$ es una solución factible de (2) y por lo tanto $x_1 + x_2 \leq \overline{x}_1 + \overline{x}_2$.

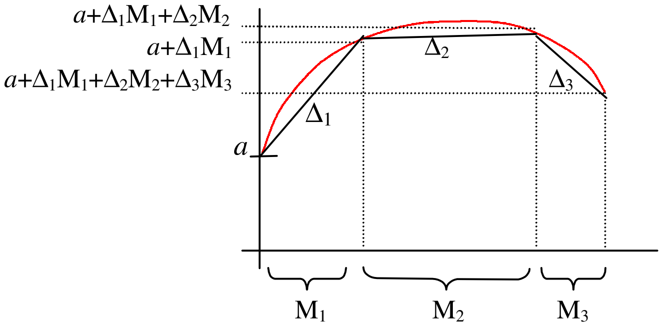
ii) si $x_1 + 2x_2 \leq 4$ entonces $5x_1 + 2x_2 \leq 10 + c$. Luego, $(x_1, x_2, 1)$ es una solución factible de (2) y por lo tanto $x_1 + x_2 \leq \overline{x}_1 + \overline{x}_2$.

Ejemplo 1.4. Función objetivo no lineal.

Consideremos el problema

$$\min \sum_j f_j(x_j)$$
$$Ax = b$$
$$x \geq 0$$

Si fuera $f_j(x_j) = c_j x_j$ se trataría de un problema de programación lineal. Pero supongamos que las funciones f_j sean contínuas pero no lineales. En tal caso, a cada f_j la podemos aproximar por trozos lineales como indica el siguiente gráfico



donde $M_i > 0$ y en cada trozo lineal hemos indicado su pendiente Δ_i. Veamos cómo podemos describir esto. Hacemos el reemplazo $x_j = y_1 + y_2 + y_3$, $f_j = a + \Delta_1 y_1 + \Delta_2 y_2 + \Delta_3 y_3$.
Con las restricciones

$$0 \leq y_i \leq M_i$$
$$y_1 \geq M_1 \delta_1$$
$$M_2 \delta_2 \leq y_2 \leq M_2 \delta_1$$
$$y_3 \leq M_3 \delta_2$$
$$0 \leq \delta_i \leq 1, \quad \delta_i \text{ entero}$$

resulta que

$$y_2 > 0 \Longrightarrow y_1 = M_1$$
$$y_3 > 0 \Longrightarrow y_2 = M_2$$

Luego, como $x_j = y_1 + y_2 + y_3$ se tiene que

i) $0 \leq x_j \leq M_1 \Longleftrightarrow y_2 = 0 = y_3$

ii) $M_1 < x_j \leq M_1 + M_2 \Longleftrightarrow y_2 > 0, y_3 = 0$

iii) $M_1 + M_2 < x_j \leq M_1 + M_2 + M_3 \Longleftrightarrow y_3 > 0$

En efecto,

i) Supongamos que $0 \leq x_j \leq M_1$. Si fuese $y_2 > 0$ entonces $y_1 = M_1$ de donde $x_j = M_1 + y_2 + y_3 > M_1$. Luego debe ser $y_2 = 0$. Pero entonces debe ser $y_3 = 0$ ya que si $y_3 > 0$ entonces $y_2 = M_2 > 0$. Recíprocamente, si $y_2 = 0 = y_3$ entonces $x_j = y_1$ y como $0 \leq y_1 \leq M_1$ entonces $0 \leq x_j \leq M_1$.

ii) Supongamos que $M_1 < x_j \leq M_1 + M_2$. Si fuese $y_3 > 0$ entonces $y_2 > M_2 > 0$ y por lo tanto $y_1 = M_1$. Pero entonces $x_j = M_1 + M_2 + y_3 > M_1 + M_2$. Luego debe ser $y_3 = 0$ y, por i), también debe ser $y_2 > 0$. Recíprocamente, si $y_2 > 0$ e $y_3 = 0$ entonces $y_1 = M_1$, de donde $x_j = M_1 + y_2$ y como $0 < y_2 \leq M_2$ entonces $M_1 < x_j \leq M_1 + M_2$.

Dejamos al lector la tarea de verificar la validez de iii).
Esto implica que para x_j tal que $0 \leq x_j \leq M_1$ estamos reemplazando x_j por y_1 y f_j por $a + \Delta_1 y_1$, es decir, por el primer trozo lineal, para x_j tal que $M_1 < x_j \leq M_1 + M_2$ estamos reemplazando x_j por $M_1 + y_2$ con $0 < y_2 \leq M_2$ y f_j por $a + \Delta_1 M_1 + \Delta_2 y_2$, es decir, por el segundo trozo lineal y, para x_j tal que $M_1 + M_2 < x_j \leq M_1 + M_2 + M_3$ estamos reemplazando x_j por $M_1 + M_2 + y_3$ con $0 < y_3 \leq M_3$ y f_j por $a + \Delta_1 M_1 + \Delta_2 M_2 + \Delta_3 y_3$, es decir, por el tercer trozo lineal.

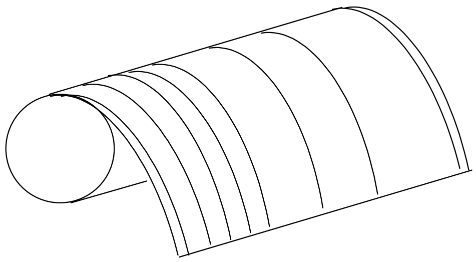
Ejemplo 1.5. Variables discretas.

Supongamos que en un problema de programación lineal queremos agregar la restricción de que una variable x sólo pueda tomar un número finito de valores $a_1, \ldots, a_k$. Esto se puede describir mediante la introducción de k variables binarias $\delta_1, \ldots, \delta_k$ y reemplazando x por $a_1\delta_1 + \cdots + a_k\delta_k$ con las restricciones

$$\begin{gathered} \delta_1 + \cdots + \delta_k = 1 \\ 0 \leq \delta_i \leq 1 \qquad \delta_i \text{ entero} \end{gathered}$$

Ejemplo 1.6. Cutting-stock problem.

La máquina de casting produce bobinas de celofán de 200 pulgadas de ancho. Los clientes realizan pedidos por bobinas de anchos menores. Sean $w_1, \ldots, w_m$ los posibles anchos pedidos por los clientes. Llamaremos *pattern* a cualquier sucesión de enteros no negativos $(a_1, \ldots, a_m)$ tal que $a_1 w_1 + \cdots + a_m w_m \leq 200$, con la que luego se instruye al operario para que corte la bobina original de 200 pulgadas para que resulten a_i bobinas de ancho w_i $(1 \leq i \leq m)$. La siguiente figura muestra una bobina con el pattern de corte $(4, 2, 1)$ para los anchos $w_1 = 20$, $w_2 = 30$ y $w_3 = 40$.



En este caso queda un borde de 10 pulgadas de descarte a cada lado.
El problema consiste en satisfacer los pedidos utilizando un mínimo número de bobinas. Sea b_i el número de bobinas de ancho w_i pedidas por los clientes $(1 \leq i \leq m)$.
Notemos que el conjunto de patterns $\{(a_1, \ldots, a_m\} \,/\, a_1 w_1 + \cdots + a_m w_m \leq 200\}$ es finito. Sean $p_1, \ldots, p_n$ sus elementos. Para cada j denotemos por $a_{1j}, \ldots, a_{mj}$ a los enteros no negativos tales que $p_j = (a_{1j}, \ldots, a_{mj})$ y $a_{1j} w_1 + \cdots + a_{mj} w_m \leq 200$. Sea x_j la cantidad de bobinas de 200 pulgadas de ancho que se cortan según el j-ésimo pattern $(1 \leq j \leq n)$. Entonces el problema se puede plantear en la forma

$$\begin{aligned} &\min \sum_j x_j \\ &\sum_j a_{ij} x_j \geq b_i \\ &\qquad x_j \geq 0 \quad x_j \text{ entero} \end{aligned}$$

En la práctica este problema no puede resolverse con la condición de que los valores de x_j sean enteros porque el número de columnas de la matriz $A = ||a_{ij}||$ es enorme (la cantidad n de patterns es el cardinal del conjunto de m-uplas $(a_1, \ldots, a_m)$ tales que $\sum a_i w_i \leq 200$). Lo que se hace, en general, es resolver el problema sin la restricción de que x_j sea entero y luego se redondea el resultado. Aún así no es fácil hallar la solución cuando n es muy grande. En ese caso se procede de la siguiente manera:
Como el problema que queremos resolver es

$$\begin{aligned} &\min \sum_j x_j \\ &\sum_j a_{ij} x_j \geq b_i \\ &\qquad x_j \geq 0 \end{aligned} \qquad (3)$$

que puede escribirse, agregando las correspondientes variables de holgura, en la forma

$$\begin{aligned} &\min \sum_j x_j + 0s_1 + \cdots + 0s_m \\ &\sum_j a_{ij} x_j - s_i = b_i \\ &\qquad x_j, s_i \geq 0 \end{aligned}$$

es decir, en la forma

$$\begin{aligned} &\min \sum_j x_j + 0s_1 + \cdots + 0s_m \\ &(\, A \quad -I\,) \begin{pmatrix} x \\ s \end{pmatrix} = b \\ &\qquad x, s \geq 0 \end{aligned}$$

entonces el problema dual de (3) resulta ser

$$\begin{aligned} &\max yb \\ &y.(\, A \quad -I\,) \leq (\underbrace{1, \ldots, 1}_{n}, \underbrace{0, \ldots, 0}_{m}) \end{aligned}$$

es decir,

$$\begin{aligned} &\max yb \\ &y.A_j \leq 1 \\ &-y \leq 0 \end{aligned} \qquad (4)$$

donde A_j es la j-ésima columna de A. Además, las condiciones de holgura complementaria son

$$\begin{aligned} (1 - y.\, A_j).x_j &= 0 \qquad (1 \leq j \leq n) \\ -\, y_i.s_i &= 0 \qquad (1 \leq i \leq m) \end{aligned}$$

Luego, si x es una solución óptima del problema primal e y es el correspondiente óptimo del dual, entonces $y \geq 0$ e $y.A_j \leq 1$. Además, si x e y son soluciones de los problemas primal y dual respectivamente, y satisfacen las condiciones de holgura complementaria entonces son óptimos.

Supongamos que hemos resuelto el problema utilizando en lugar de la matriz A la submatriz formada por un subconjunto relativamente pequeño $\{A_{j_1}, \ldots, A_{j_r}\}$ de columnas de A obteniendo una solución óptima $(u_{j_1}, \ldots, u_{j_r})$ y que $v = (v_1, \ldots, v_m)$ es el correspondiente óptimo del dual de este nuevo problema.

Luego $v \geq 0$ y satisface $v.A_{j_t} \leq 1$ para $t = 1, \ldots, r$.
Si fuese $v.A_j \leq 1$ para todo $1 \leq j \leq n$ entonces v sería una solución factible de (4). Además, como $u_{j_t} \geq 0$ para $t = 1, \ldots, r$ y se verifica $a_{ij_1}u_{j_1} + \cdots + a_{ij_r}u_{j_r} \geq b_i$ para $i = 1, \ldots, m$, tomando $u_j = 0$ para $j \neq j_1, \ldots, j_r$ resulta que $u = (u_1, \ldots, u_n)$ es una solución factible de (3). Como además v y $(u_{j_1}, \ldots, u_{j_r})$ satisfacen las condiciones de holgura complementaria del problema con reducción de columnas entonces verifican

$$\begin{aligned} (1 - y.\, A_{j_t}).u_{j_t} &= 0 \qquad (1 \leq t \leq r) \\ - y_i.s_i &= 0 \qquad (1 \leq i \leq m) \end{aligned}$$

de donde resulta que u y v satisfacen las condiciones de holgura complementaria del problema original. Luego u es una solución óptima de (3).

Veamos cómo determinar si $v.A_j \leq 1$ para todo $1 \leq j \leq n$ y qué hacer si esto no se satisface.
En primer lugar, resolvemos el problema de la mochila

$$\begin{aligned} &\max \sum_i v_i z_i \\ &\sum_i w_i z_i \leq 200 \qquad\qquad (5) \\ &z_i \geq 0 \quad z_i \text{ entero} \end{aligned}$$

Si el óptimo $(\overline{z}_1, \ldots, \overline{z}_m)$ de (5) satisface $\sum v_i \overline{z}_i \leq 1$, todos los patterns $(a_{1j}, \ldots, a_{mj})$ satisfacen $\sum v_i a_{ij} \leq 1$, es decir, $v.A_j \leq 1$ para todo j como queríamos (notar que todo pattern es una solución factible de (5)). En caso contrario, al terminar el algoritmo para el problema reducido que utiliza, en lugar de la matriz A, la submatriz formada por las columnas $A_{j_1}, \ldots, A_{j_r}$ de A, lo continuamos, ahora incorporando la columna

$$\begin{pmatrix} 1 \\ \overline{z}_1 \\ \vdots \\ \overline{z}_m \end{pmatrix}$$

(ver capítulo 1, sección 10: utilizando el algoritmo simplex revisado es posible incorporar una columna en cualquier momento). Como $(\overline{z}_1, \ldots, \overline{z}_m)$ es una solución factible de (5) entonces es un pattern y por lo tanto alguna de las columnas de A, y como estamos suponiendo que $\sum v_i \overline{z}_i > 1$ entonces no es ninguna de las columnas $j_1, \ldots, j_r$. Luego, al continuar el algoritmo incorporando esta columna lo que estamos haciendo es repetir el proceso anterior, ahora para un problema reducido que utiliza $r+1$ columnas de A. Repetimos este procedimiento hasta que el óptimo v correspondiente al óptimo del problema reducido verifique $v.A_j \leq 1$ para todo $1 \leq j \leq n$.

Ejemplo 1.7. Job scheduling.

Los tubos de acero se elaboran en cuatro etapas sucesivas: laminación, ajuste, tratamiento térmico y roscado. Se agrupan en lotes, cada uno conteniendo todos los tubos con una misma especificación técnica. Cada lote pasa por las cuatro etapas y, en cada etapa, el orden de procesamiento de los lotes no tiene porqué coincidir. Además, entre dos etapas puede haber un stock intermedio.
Supongamos que hay n lotes y sea p_{ik} $(1 \leq i \leq n, \quad 1 \leq k \leq 4)$ el tiempo de procesamiento del lote i en la etapa k. Supongamos además que el procesamiento se inicia en el instante cero y sea z el instante en que se terminan de procesar todos los lotes. Se desea determinar en qué instante x_{ik} se debe iniciar el proceso del lote i en la etapa k $(1 \leq i \leq n, \quad 1 \leq k \leq 4)$ de manera tal que el valor de z sea mínimo. Las restricciones a tener en cuenta son:

i) el procesamiento de un lote no puede iniciarse en una etapa si no ha sido terminado su procesamiento en la etapa anterior.

ii) dos lotes no pueden procesarse simultáneamente en una misma etapa, es decir, para cada par de lotes $i \neq j$, el lote i debe haber salido de la etapa k antes de que entre en ella el lote j o viceversa.

Luego, el problema puede plantearse en la forma

$$\begin{aligned} &\min z \\ &x_{i1}+p_{i1} \leq x_{i2} \\ &x_{i2}+p_{i2} \leq x_{i3} \\ &x_{i3}+p_{i3} \leq x_{i4} \\ &x_{i4}+p_{i4} \leq z \\ &x_{ik}+p_{ik} \leq x_{jk} \text{ o } x_{jk}+p_{jk} \leq x_{ik} \quad (i \neq j) \\ &x_{ik} \geq 0 \end{aligned}$$

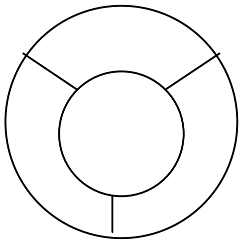
Notemos que es la condición

$$x_{ik}+p_{ik} \leq x_{jk} \text{ o } x_{jk}+p_{jk} \leq x_{ik} \quad (i \neq j)$$

la que hace que este sea un problema de programación lineal entera. Históricamente, el planteo del problema de scheduling llamó la atención por la simplicidad de su solución en el caso en que haya dos etapas (ver [Jo, 1954]).
No se ha encontrado una solución simple para tres o más etapas. En el caso de dos o tres etapas la solución óptima tiene la particularidad de que el orden de procesamiento de los lotes es el mismo en todas las etapas.

Ejemplo 1.8. El problema de los cuatro colores.

Supongamos que queremos pintar las distintas regiones de un mapa que está dibujado sobre un plano de manera tal que regiones adyacentes estén pintadas de colores distintos. La siguiente figura muestra un ejemplo donde se necesitan cuatro colores.



Durante 150 años se conjeturó que cualquier mapa se podía pintar, con la mencionada restricción, con a lo sumo cuatro colores. El esfuerzo hecho para resolver el problema fue uno de los principales motivos del desarrollo de la teoría de grafos. Finalmente, en 1976 se demostró esta conjetura (ver [Ap-Ha, 1978]).
Sea $G = (V, E)$ un grafo no dirigido. Colorear el grafo significa asignar un color a cada vértice i de manera tal que i y j tengan colores distintos para toda rama $(i, j) \in E$.
El mínimo número de colores necesarios para colorear a G se llama el número cromático de G.
Todo mapa se puede representar en un grafo planar no dirigido (para la definición de grafo planar ver sección 3 del capítulo 4) donde cada vértice representa una región del mapa y dos vértices están unidos por una rama si y sólo si las regiones que representan son adyacentes. El teorema de los cuatro colores afirma que el número cromático de cualquier grafo planar es a lo sumo cuatro.
Supongamos ahora que tenemos un grafo no dirigido cualquiera (no necesariamente planar) y queremos ver si podemos colorearlo con a lo sumo cuatro colores. Sean $0, 1, 2, 3$ los cuatro colores. Entonces estos colores

son suficientes si y sólo si para cada vértice i existe $x_i \in \{0,1,2,3\}$ (el color asignado al vértice i) tal que $x_i \neq x_j$ para todo $(i,j) \in E$.
Observando que $x_i \neq x_j$ si y sólo si $x_i > x_j$ o $x_j > x_i$ si y sólo si $1 \leq x_i - x_j$ o $1 \leq x_j - x_i$, el problema se traduce en determinar si

$$\begin{array}{ll} 1 \leq x_i - x_j \text{ o } 1 \leq x_j - x_i & ((i,j) \in E) \\ 0 \leq x_i \leq 3, & x_i \text{ entero} \end{array} \tag{6}$$

es factible. Sean s_{ij}, t_{ij} $((i,j) \in E)$ nuevas variables y consideremos el problema

$$\begin{array}{c} \min z \\ z = \displaystyle\sum_{(i,j)\in E} s_{ij} + t_{ij} \\ 1 \leq x_i - x_j + s_{ij} \text{ o } 1 \leq x_j - x_i + t_{ij} \qquad ((i,j) \in E) \\ s_{ij}, t_{ij} \geq 0 \\ 0 \leq x_i \leq 3, \qquad x_i \text{ entero} \end{array} \tag{7}$$

Entonces (6) es factible si y sólo si (7) tiene una solución óptima tal que $z = 0$.

Ejemplo 1.9. El problema del viajante.

Un viajante debe visitar las ciudades $1, 2, \ldots, n$ partiendo de la ciudad 0, pasando por cada una de las ciudades $1, 2, \ldots, n$ una y sólo una vez y volviendo luego a la ciudad 0. Sea c_{ij} el costo de viajar de la ciudad i a la ciudad j $(0 \leq i, j \leq n)$. Si no hay camino de i a j tomamos $c_{ij} = \infty$.
A cada una de las posibles maneras de hacer el recorrido la llamamos un *circuito Hamiltoniano*. Definimos el costo de un tal circuito como la suma de los costos de los tramos que lo componen. El problema consiste en hallar un circuito Hamiltoniano de mínimo costo.
Representamos la situación en un grafo dirigido completo $G = (V, E)$ donde

$$V = \{0, 1, 2, \ldots, n\}$$

y asignamos costo c_{ij} a cada rama (i,j).
Dado un circuito Hamiltoniano $\mathcal{C}$ (es decir, un circuito dirigido en G que pasa por cada vértice una y sólo una vez), para cada (i,j) sea

$$\delta_{ij} = \begin{cases} 1 & \text{si } (i,j) \in \mathcal{C} \\ 0 & \text{si no} \end{cases}$$

Entonces el costo de $\mathcal{C}$ es $\sum c_{ij}\delta_{ij}$.
Observemos que para cada vértice i hay una sola rama de $\mathcal{C}$ cuya cola es i y una sola rama de $\mathcal{C}$ cuya punta es i. Luego se satisface

$$\begin{array}{ll} \displaystyle\sum_j \delta_{ij} = 1 & (0 \leq i \leq n) \\ \displaystyle\sum_i \delta_{ij} = 1 & (0 \leq j \leq n) \end{array} \tag{8}$$

Supongamos ahora que para cada (i,j) tenemos definido un δ_{ij} tal que $\delta_{ij} = 0$ o $\delta_{ij} = 1$ y de manera tal que valga (8).
Si el conjunto de ramas (i,j) tales que $\delta_{ij} = 1$ fuese un circuito entonces podríamos pensar a cada circuito Hamiltoniano como una solución factible de

$$\begin{array}{ll} \displaystyle\sum_j \delta_{ij} = 1 & (0 \leq i \leq n) \\ \displaystyle\sum_i \delta_{ij} = 1 & (0 \leq j \leq n) \\ 0 \leq \delta_{ij} \leq 1, & \delta_{ij} \text{ entero} \end{array} \tag{9}$$

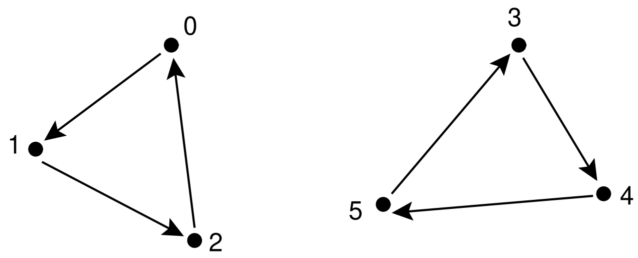
cuyo costo es $\sum c_{ij}\delta_{ij}$. Lamentablemente esto no es así, como lo muestra el siguiente ejemplo.
Sea $V = \{0, 1, 2, 3, 4, 5\}$ y tomemos

$$\delta_{01} = \delta_{20} = \delta_{12} = \delta_{34} = \delta_{45} = \delta_{53} = 1$$

y los restantes δ_{ij} iguales a cero. Es decir, el conjunto (i, j) tales que $\delta_{ij} = 1$ es

$$\{(0, 1), (2, 0), (1, 2), (3, 4), (4, 5), (5, 3)\}$$

En este caso δ_{ij} $(0 \leq i, j \leq 5)$ es una solución factible de (9) pero el conjunto de ramas (i, j) tales que $\delta_{ij} = 1$ no forman un circuito sino dos subcircuitos:



Pero agregando la ingeniosa condición

$$u_i - u_j + n\delta_{ij} \leq n - 1 \qquad (1 \leq i, j \leq n,\ i \neq j)$$

donde u_i $(1 \leq i \leq n)$ son números reales nos permitirá plantear el problema del viajante por programación lineal entera. Esto se debe a que si δ_{ij} es una solución factible de (9) entonces existen u_i satisfaciendo esta nueva condición si y sólo si el conjunto de ramas (i, j) tales que $\delta_{ij} = 1$ es un circuito. En tal caso el costo de ese circuito es $\sum c_{ij}\delta ij$.

Afirmación. El problema de programación lineal entera

$$\begin{aligned} \min \sum c_{ij}\delta_{ij} & \\ \sum_j \delta_{ij} = 1 & \qquad (0 \leq i \leq n) \\ \sum_i \delta_{ij} = 1 & \qquad (0 \leq j \leq n) \\ u_i - u_j + n\delta_{ij} \leq n - 1 & \qquad (1 \leq i, j \leq n,\ i \neq j) \\ 0 \leq \delta_{ij} \leq 1, & \qquad \delta_{ij} \text{ entero} \end{aligned} \tag{10}$$

resuelve el problema del viajante.

Demostración: Mostraremos que cada circuito Hamiltoniano puede representarse como una solución factible de (10).
Supongamos que $\mathcal{C}$ es un circuito Hamiltoniano. Para cada (i, j) sea

$$\delta_{ij} = \begin{cases} 1 & \text{si } (i, j) \in \mathcal{C} \\ 0 & \text{si no} \end{cases}$$

Sea $u_i = 1$ si i es la primera ciudad visitada, $u_i = 2$ si i es la segunda ciudad visitada, etc. Por ejemplo, si $n = 5$ y el circuito es

$$0 \longrightarrow 2 \longrightarrow 5 \longrightarrow 3 \longrightarrow 1 \longrightarrow 4 \longrightarrow 0$$

entonces

$$u_1 = 4, \quad u_2 = 1, \quad u_3 = 3, \quad u_4 = 5, \quad u_5 = 2$$

Entonces

$$u_i - u_j + n\delta_{ij} = \begin{cases} u_i - u_j & \text{si } (i,j) \notin \mathcal{C} \\ -1 + n & \text{si } (i,j) \in \mathcal{C} \end{cases}$$

ya que si $(i,j) \notin \mathcal{C}$ entonces $\delta_{ij} = 0$ y si $(i,j) \in \mathcal{C}$ entonces el viajante visita j justo después de visitar i, de donde $u_j = u_i + 1$ y como $(i,j) \in \mathcal{C}$ entonces $\delta_{ij} = 1$.
Luego, como $1 \leq u_i \leq n$ entonces $u_i - u_j + n\delta_{ij} \leq n - 1$ para todo $i \neq j$ $(1 \leq i, j \leq n)$.
Esto muestra que cada circuito Hamiltoniano determina una solución factible de (10).
Recíprocamente, supongamos ahora que tenemos una solución factible de (10). Probaremos que el conjunto de ramas $\mathcal{C} = \{(i,j) \,/\, \delta_{ij} = 1\}$ es un circuito Hamiltoniano. Como cada δ_{ij} es cero o uno y valen

$$\sum_j \delta_{ij} = 1 \qquad (0 \leq i \leq n)$$
$$\sum_i \delta_{ij} = 1 \qquad (0 \leq j \leq n)$$

entonces para cada i hay un único j tal que $\delta_{ij} = 1$ y un único j tal que $\delta_{ji} = 1$ (notemos que la segunda igualdad puede escribirse $\sum_j \delta_{ji} = 1$ para $0 \leq i \leq n$). Esto significa que para cada vértice i hay una sola rama de $\mathcal{C}$ cuya cola es i y una sola rama de $\mathcal{C}$ cuya punta es i. Luego, sólo debemos ver que las ramas de $\mathcal{C}$ forman un circuito y no dos o más subcircuitos. Supongamos que formaran dos o más subcircuitos. Entonces podemos elegir uno de ellos que no pase por el vértice 0. Sea $\mathcal{E}$ el conjunto de ramas que lo forman y sea $k = \#\mathcal{E}$. Como $u_i - u_j + n\delta_{ij} \leq n - 1$ para todo $1 \leq i, j \leq n$ tal que $i \neq j$ entonces

$$\sum_{(i,j)\in\mathcal{E}} u_i - u_j + n\delta_{ij} \leq k(n-1)$$

Pero como para cada i hay una única rama en $\mathcal{E}$ cuya cola es i y una única rama cuya punta es i entonces

$$\sum_{(i,j)\in\mathcal{E}} u_i - u_j = 0$$

ya que cada u_i aparece una vez sumando y una vez restando. Además, como las ramas de $\mathcal{E}$ forman un subcircuito de $\mathcal{C}$ entonces $\delta_{ij} = 1$ para todo $(i,j) \in \mathcal{E}$. Por lo tanto,

$$\sum_{(i,j)\in\mathcal{E}} n\delta_{ij} = kn$$

de donde

$$\begin{aligned} kn = 0 + kn &= \sum_{(i,j)\in\mathcal{E}} u_i - u_j + \sum_{(i,j)\in\mathcal{E}} n\delta_{ij} = \\ &= \sum_{(i,j)\in\mathcal{E}} u_i - u_j + n\delta_{ij} \leq k(n-1) \end{aligned}$$

con lo cual se tiene que $kn \leq k(n-1)$, lo que es un absurdo pues $k > 0$. Luego, cada solución factible de (10) determina un circuito Hamiltoniano. □

Ejemplo 1.10. Cuadrados latinos ortogonales.

Supongamos que queremos comparar el rendimiento de n variedades de trigo. Si plantamos cada variedad en una franja del terreno podría ocurrir que el trigo plantado en la primera franja tuviera un alto rendimiento debido a que esa zona del terreno es más fértil que las otras y no porque esa variedad de trigo sea en realidad la de mayor rendimiento. Para evitar que las diferencias en la fertilidad del suelo influyan en los resultados de nuestro experimento, conviene dividir el terreno en n^2 parcelas y plantar cada una de las n variedades de trigo en n parcelas de manera que cada franja (horizontal o vertical) del terreno contenga las n variedades. Por ejemplo, para $n = 3$, las variedades a, b y c podrían disponerse en la forma

a	b	c
b	c	a
c	a	b

Un *cuadrado latino* de $n \times n$ es una disposición de n símbolos en una matriz de $n \times n$ de manera tal que cada símbolo aparezca exactamente una vez en cada fila y en cada columna.
El nombre de cuadrado latino tiene se debe a que Euler utilizaba como símbolos las letras $a, b, c \dots$
Es fácil ver que para cualquier n siempre existe un cuadrado latino de $n \times n$. En efecto, basta tomar la disposición

$$\begin{pmatrix} 1 & 2 & 3 & \dots & n-1 & n \\ 2 & 3 & 4 & \dots & n & 1 \\ 3 & 4 & 5 & \dots & 1 & 2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ n-2 & n-1 & n & \dots & n-4 & n-3 \\ n-1 & n & 1 & \dots & n-3 & n-2 \\ n & 1 & 2 & \dots & n-2 & n-1 \end{pmatrix}$$

Supongamos ahora que queremos estudiar el comportamiento de las tres variedades de trigo a, b, c cuando se les aplica cada uno de los fertilizantes α, β, γ. Para que nuestro experimento no se vea afectado por las posibles variaciones en la fertilidad del suelo, debemos disponer cada uno de los 9 pares (variedad de trigo, fertilizante) en las 9 parcelas de manera en cada fila y en cada columna aparezcan todas las variedades de trigo y todos los fertilizantes, es decir, de manera que tanto las primeras coordenadas como las segundas formen un cuadrado latino.
Si $A = ||a_{ij}||$ y $B = ||b_{ij}||$ son dos cuadrados latinos de $n \times n$, la matriz de $n \times n$ cuyo coeficiente en el lugar ij es el par (a_{ij}, b_{ij}) se llama la superposición de A con B. Por lo tanto, lo que necesitamos es construír dos cuadrados latinos (uno para las variedades de trigo y otro para los fertilizantes) de forma tal que la superposición de ambos contenga todos los posibles pares (variedad de trigo, fertilizante). Por ejemplo, si consideramos los cuadrados latinos

a	b	c
b	c	a
c	a	b

β	γ	α
α	β	γ
γ	α	β

vemos que su superposición contiene todos los posibles pares (variedad de trigo, fertilizante)

(a, β)	(b, γ)	(c, α)
(b, α)	(c, β)	(a, γ)
(c, γ)	(a, α)	(b, β)

Esto motiva la siguiente definición:

Dos cuadrados latinos de $n \times n$ se dicen *ortogonales* (o *grecolatinos*) si los coeficientes de su superposición son n^2 pares distintos. En general, m cuadrados latinos $L_1, \dots, L_m$ de $n \times n$ se dicen ortogonales si L_i y L_j son ortogonales para todo $i \neq j$ $(1 \leq i, j \leq m)$.

Euler conjeturó que no existen pares de cuadrados latinos ortogonales de $n \times n$, para n de la forma $4k + 2$ $(k \in \mathbb{N})$. Recién en 1960 se pudo determinar que esta conjetura era falsa cuando Bose, Parker y Shrikhande probaron que, salvo para $n = 2$ o 6, siempre existe un par de cuadrados latinos ortogonales de $n \times n$.

Se puede demostrar que para todo $n \geq 2$ hay a lo sumo $n-1$ cuadrados latinos ortogonales de $n \times n$. Si para un dado n hay exactamente $n-1$ cuadrados latinos ortogonales decimos que para n hay un sistema completo. Se sabe que para los n que son potencias de primos siempre hay un sistema completo pero, en general, para aquellos n que no son potencia de un primo (salvo para algunos casos particulares) no se sabe aún si hay o no un sistema completo.

A continuación daremos un planteo por programación lineal entera para resolver el problema de determinar, para un n dado, un par de cuadrados latinos ortogonales de $n \times n$.
Supongamos que tenemos dos cuadrados latinos ortogonales $A = ||a_{ij}||$, $B = ||b_{ij}||$. Entonces su superposición es la matriz C cuyo coeficiente en el lugar ij es el par (a_{ij}, b_{ij}). Sin pérdida de generalidad podemos suponer que $a_{ij}, b_{ij} \in \{1, 2, \ldots, n\}$.

Para cada i, j, k, s $(1 \leq i, j, k, s \leq n)$ sea $\delta_{ijks} = \begin{cases} 1 & \text{si } a_{ij} = k \text{ y } b_{ij} = s \\ 0 & \text{si no} \end{cases}$

Como A y B son cuadrados latinos entonces cada k aparece en una única fila y en una única columna de A y cada s aparece en una única fila y en una única columna de B. Luego se verifican las condiciones

1) para cada i, j existen únicos k, s tales que $\delta_{ijks} = 1$.
2) para cada i, k existen únicos j, s tales que $\delta_{ijks} = 1$.
3) para cada i, s existen únicos j, k tales que $\delta_{ijks} = 1$.
4) para cada j, k existen únicos i, s tales que $\delta_{ijks} = 1$.
5) para cada j, s existen únicos i, k tales que $\delta_{ijks} = 1$.

En efecto, la validez de 1) resulta de que fijados i y j entonces $\delta_{ijks} = 1$ si y sólo si $k = a_{ij}$ y $s = b_{ij}$. La de 2) de que fijados i y k entonces $\delta_{ijks} = 1$ si y sólo si j es la única columna de A tal que $a_{ij} = k$ y $s = b_{ij}$. La de 3) de que fijados i y s entonces $\delta_{ijks} = 1$ si y sólo si j es la única columna de B tal que $b_{ij} = s$ y $k = a_{ij}$. La de 4) de que fijados j y k entonces $\delta_{ijks} = 1$ si y sólo si i es la única fila de A tal que $a_{ij} = k$ y $s = b_{ij}$. Finalmente, la de 5) de que, fijados j y s, $\delta_{ijks} = 1$ si y sólo si i es la única fila de B tal que $b_{ij} = s$ y $k = a_{ij}$.
Además, como A y B son ortogonales, entonces cada (k, s) aparece una y sólo una vez en C. Luego, fijados k y s existen únicos i, j tales que $c_{ij} = (k, s)$, es decir, tales que $(a_{ij}, b_{ij}) = (k, s)$. Por lo tanto también vale la condición

6) para cada k, s existen únicos i, j tales que $\delta_{ijks} = 1$.

El hecho de que cada δ_{ijks} es cero o uno y que satisfagan las condiciones 1) a 6) es equivalente a que los δ_{ijks} sean una solución factible de

$$\begin{aligned}
&\sum_{k,s} \delta_{ijks} = 1 \qquad (1 \leq i, j \leq n) \\
&\sum_{j,s} \delta_{ijks} = 1 \qquad (1 \leq i, k \leq n) \\
&\sum_{j,k} \delta_{ijks} = 1 \qquad (1 \leq i, s \leq n) \\
&\sum_{i,s} \delta_{ijks} = 1 \qquad (1 \leq j, k \leq n) \\
&\sum_{i,k} \delta_{ijks} = 1 \qquad (1 \leq j, s \leq n) \\
&\sum_{i,j} \delta_{ijks} = 1 \qquad (1 \leq k, s \leq n) \\
&0 \leq \delta_{ijks} \leq 1, \qquad \delta_{ijks} \text{ enteros}
\end{aligned} \tag{11}$$

Esto muestra que a cada par de cuadrados latinos le podemos asociar una solución factible de (11). Notemos que el sistema lineal en (11) tiene $6n^2$ ecuaciones con n^4 incógnitas.

Recíprocamente, supongamos que δ_{ijks} $(1 \leq i, j, k, s \leq n)$ es una solución factible de (11). Entonces, como cada δ_{ijks} es cero o uno resulta que los δ_{ijks} verifican las condiciones 1) a 6).
Sean $A = ||a_{ij}||$ y $B = ||b_{ij}||$ las matrices definidas por

$$(a_{ij}, b_{ij}) = (k, s) \quad \Longleftrightarrow \quad \delta_{ijkl} = 1$$

Notemos que la validez de la condición 1) garantiza que A y B están bien definidas. Además, la validez de las condiciones 2) a 6) implican que A y B son cuadrados latinos ortogonales. Veremos que la condición 2) implica que en cada fila de A cada símbolo aparece exactamente una vez y dejamos el resto a cargo del lector.
Dado i, supongamos que existen $j \neq j'$ tales que $a_{ij} = a_{ij'}$. Sean $k = a_{ij}$, $s = b_{ij}$ y $s' = b_{ij'}$. Entonces $(a_{ij}, b_{ij}) = (k, s)$ y $(a_{ij'}, b_{ij'}) = (k, s')$, de donde $\delta_{ijks} = 1 = \delta_{ij'ks'}$, lo que contradice 2).
Esto muestra que podemos construír los dos cuadrados latinos ortogonales buscados a partir de una solución factible de (11).
La dificultad de este problema radica en el hecho de que el tamaño del sistema lineal en (11) es muy grande ($6n^2$ ecuaciones con n^4 incógnitas) y el tiempo de ejecución del algoritmo que resuelve problemas de programación lineal entera crece exponencialmente con el tamaño del problema. Por ejemplo, para $n = 10$ tendríamos un sistema con 600 ecuaciones y 10000 incógnitas.

Ejemplo 1.11. El problema SAT (satisfiability).

Sean $x_1, \ldots, x_n$ variables que sólo pueden tomar los valores de verdad V (verdadero) o F (falso). A estas variables las llamaremos *variables lógicas*. Consideremos el conjunto de todas las expresiones que pueden obtenerse a partir de las variables lógicas $x_1, \ldots, x_n$ utilizando los conectivos lógicos $\wedge$, $\vee$ y $\neg$. A los elementos de este conjunto los llamaremos *expresiones booleanas*. Notemos que conectando dos expresiones booleanas con $\wedge$, $\vee$ o $\neg$ obtenemos otra expresión booleana, es decir, los conectivos lógicos definen operaciones en este conjunto.
Cuando en una expresión booleana sustituimos cada una de las variables por un valor de verdad obtenemos un valor de verdad para la expresión booleana. Por ejemplo, cuando sustituimos x_2, x_3 y x_4 por V y x_1 por F en la expresión booleana $(x_1 \vee x_3) \wedge (x_4 \vee \neg x_2) \wedge x_3$ obtenemos el valor V pero cuando sustituimos x_1, x_2 y x_3 por V y x_4 por F obtenemos el valor F.
Una *cláusula* es una expresión booleana donde las variables sólo están conectadas por $\vee$ o $\neg$. Luego, la expresión booleana anterior es una conjunción de tres cláusulas. Si una expresión booleana ϕ es una conjunción de cláusulas, entonces, para cualquier sustitución de las variables por V o F, ϕ será V si y sólo si todas las cláusulas lo son.
Consideremos la conjunción de cinco cláusulas

$$\phi : \ (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_3 \vee x_1) \wedge (\neg x_1 \neg x_2 \neg x_3)$$

Dejamos a cargo del lector verificar que, cualquiera sea la sustitución que hagamos en las variables, el valor de verdad de ϕ obtenido es F. Esto muestra que, dada una expresión booleana, tiene sentido el problema de determinar si existe alguna asignación de V o F a las variables tal que la expresión resulte verdadera. Este problema se conoce como el problema de satisfabilidad (SAT).
Una manera de resolver este problema consiste en hallar el valor de verdad de la expresión booleana para cada una de las posibles sustituciones de las variables por V o F pero eso significaría, en el caso de n variables, hallar el valor de verdad de la expresión en cada uno de los 2^n casos posibles.
Veremos una forma de resolver este problema por programación lineal entera. Esto no significa que este método sea mejor que el anterior ya que no hay ningún algoritmo conocido de complejidad polinomial que resuelva los problemas de programación lineal entera.

Como toda expresión booleana es equivalente a una conjunción de cláusulas nos limitaremos al caso en que nuestra expresión es de esa forma. Supongamos que queremos determinar si existe alguna asignación de V o F a las variables $x_1, \ldots, x_4$ tal que la expresión

$$\phi: \ (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_3 \vee x_4)$$

resulte verdadera.
Como ϕ es verdadera sii cada una de sus cláusulas lo es entonces el problema es equivalente a determinar si existe alguna asignación de V o F a las variables $x_1, \ldots, x_4$ tal que todas las cláusulas sean verdaderas.
Con la convención

$$x_i = 1 \text{ si y sólo si } x_i \text{ es V}$$
$$x_i = 0 \text{ si y sólo si } x_i \text{ es F}$$

y teniendo en cuenta que $\neg x_i$ es V sii x_i es F sii $x_i = 0$ sii $1 - x_i = 1$ resulta que existe una asignación de V o F a $x_1, \ldots, x_4$ de manera tal que todas las cláusulas sean verdaderas sii existen $x_1, \ldots, x_4 \in \{0, 1\}$ tales que

$$\begin{aligned}
&x_1 + x_2 + (1 - x_3) \geq 1 \\
&(1 - x_1) + x_3 + (1 - x_4) \geq 1 \\
&x_2 + (1 - x_3) + x_4 \geq 1 \\
&(1 - x_1) + (1 - x_2) + (1 - x_3) \geq 1 \\
&x_1 + x_2 + x_3 + x_4 \geq 1
\end{aligned}$$

(Notar que como, para cada i, x_i es un entero no negativo entonces $x_1 + x_2 + (1 - x_3) \geq 1$ sii $x_1 \neq 0$ o $x_2 \neq 0$ o $(1 - x_3) \neq 0$ sii $x_1 = 1$ o $x_2 = 1$ o $(1 - x_3) = 1$ ya que x_i sólo puede tomar los valores 0 y 1. Por lo tanto, $x_1 + x_2 + (1 - x_3) \geq 1$ sii x_1 es V o x_2 es V o $\neg x_3$ es V sii $x_1 \vee x_2 \vee \neg x_3$ es V)
Luego, existe alguna asignación de V o F a las variables $x_1, \ldots, x_4$ tal que ϕ sea verdadera si y sólo si el problema de programación lineal entera

$$\begin{aligned}
&x_1 + x_2 + (1 - x_3) \geq 1 \\
&(1 - x_1) + x_3 + (1 - x_4) \geq 1 \\
&x_2 + (1 - x_3) + x_4 \geq 1 \\
&(1 - x_1) + (1 - x_2) + (1 - x_3) \geq 1 \\
&x_1 + x_2 + x_3 + x_4 \geq 1 \\
&0 \leq x_i \leq 1 \quad x_i \text{ entero}
\end{aligned}$$

es factible, lo que es equivalente a que el problema de programación lineal entera mixta

$$\begin{aligned}
&\qquad \max z \\
&x_1 + x_2 + (1 - x_3) \geq z \\
&(1 - x_1) + x_3 + (1 - x_4) \geq 1 \\
&x_2 + (1 - x_3) + x_4 \geq 1 \\
&(1 - x_1) + (1 - x_2) + (1 - x_3) \geq 1 \\
&x_1 + x_2 + x_3 + x_4 \geq 1 \\
&0 \leq x_i \leq 1 \quad x_i \text{ entero}
\end{aligned}$$

tenga una solución óptima (z, x_1, x_2, x_3, x_4) que satisfaga $z \geq 1$.

2. El método de redondeo.

Supongamos que para resolver un problema de programación lineal entera nos olvidamos de la restricción entera, lo resolvemos como un problema de programación lineal y luego redondeamos la solución hallada tomando la solución entera más próxima que sea factible. En general, esto funciona bien cuando las componentes de la solución óptima son "grandes". El siguiente ejemplo muestra lo que puede ocurrir si la solución óptima es "pequeña".

Ejemplo 2.1. Consideremos el problema de programación lineal entera

$$\begin{aligned} &\max\ 21x_1 + 11x_2 \\ &7x_1 + 4x_2 \leq 13 \\ &x_i \geq 0, \quad x_i \text{ entero} \end{aligned}$$

En este caso la solución óptima es $(x_1, x_2) = (0, 3)$ con valor del funcional $21.0 + 11.3 = 33$.
En cambio, si nos olvidamos de la restricción entera, la solución óptima es $(x_1, x_2) = (\frac{13}{7}, 0)$. Redondeando por arriba obtenemos $(2, 0)$ que no es una solución factible y redondeando por abajo obtenemos la solución factible $(1, 0)$ con valor del funcional $21.1 + 11.0 = 21$, que está lejos del valor del funcional en el óptimo.

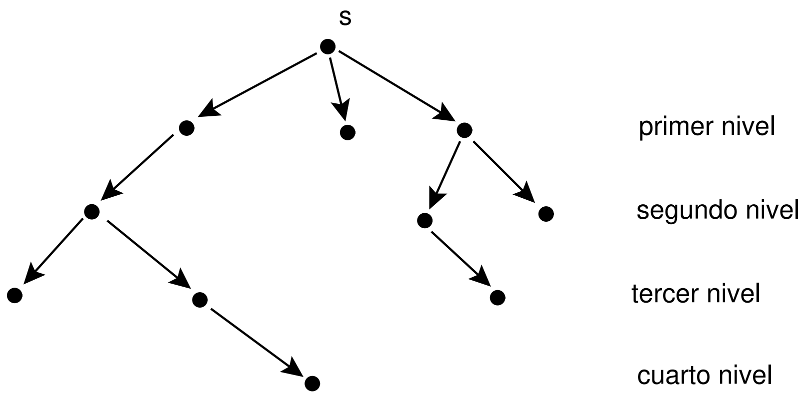
3. El método de branch and bound.

Definición 3.1. Un *árbol dirigido con raíz* es un grafo dirigido, conexo y acíclico que tiene un vértice distinguido s al que llamamos raíz tal que para cualquier otro vértice v hay un camino dirigido de s a v.

Observación 3.2. Si (u, v) es una rama de un árbol dirigido con raíz entonces (v, u) no lo es.

Definición 3.3. Si (u, v) es una rama de un árbol dirigido con raíz diremos que u es el *padre* de v y que v es el *hijo* de u. Dado un vértice u, el conjunto de vértices v tales que existe un camino dirigido de u a v se llama la *descendencia* de u. Decimos que el árbol es *binario* si cada vértice que no sea una hoja (ver definición 2.3. del capítulo 2) tiene exactamente dos hijos (notemos que las hojas son los vértices que no tienen hijos). Al conjunto de hijos de la raíz lo llamamos el *primer nivel*, al conjunto de nietos de la raíz (es decir, al conjunto de hijos de los hijos de la raíz) lo llamamos el *segundo nivel*, etc.

Ejemplo 3.4. El grafo



es un árbol dirigido con raíz s y cuatro niveles.

Observación 3.5. Si un árbol dirigido con raíz es binario y tiene n niveles entonces tiene $2^{n+1} - 1$ vértices.

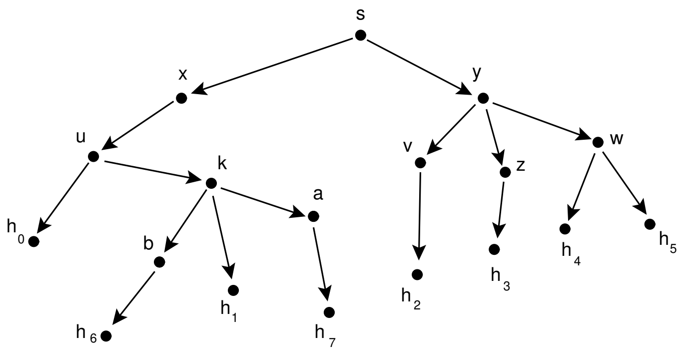
Consideremos el siguiente problema: dado un árbol dirigido con raíz s, donde cada vértice x tiene asignado un costo $c(x)$ que satisface $c(x) \leq c(y)$ para toda rama (x, y) queremos hallar una hoja de mínimo costo.

Describiremos un algoritmo, conocido como branch and bound, que resuelve este problema. El procedimiento utiliza una mezcla de *backtracking* (volver al vértice anterior para examinar alguno de sus hijos que todavía no ha sido examinado) y un criterio particular de *poda* que consiste en eliminar toda la descendencia de un vértice x cuando se satisfaga que $c(x) \geq c(h)$ para alguna hoja h encontrada previamente (es decir, cuando ninguna hoja descendiente de x puede ser la solución óptima del problema).

Descripción del algoritmo.

0. $L = \{s\}$, $c = \infty$.
1. Si x es, de los elementos de L, el último que ingresó, $L = L - \{x\}$. Calcular $c(x)$. Si $c(x) \geq c$, goto 4.
2. Si x no es una hoja, $L = L \cup \{$ hijos de $x\}$, goto 4.
3. $h = x$, $c = c(x)$.
4. Si $L \neq \emptyset$ goto 1.
5. STOP.

Ejemplo 3.6. Consideremos el árbol dirigido con raíz s



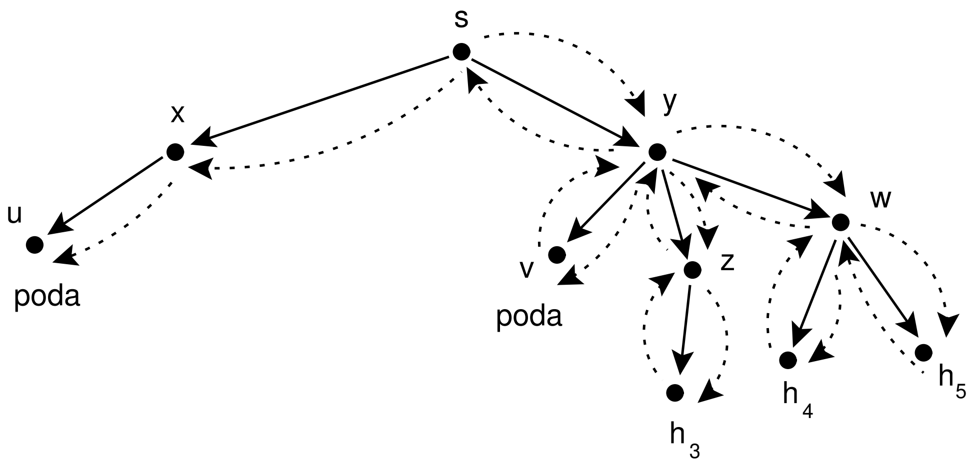
donde $c(s) = 3$, $c(x) = 4$, $c(y) = 4$, $c(z) = 4$, $c(u) = 7$, $c(v) = 8$, $c(w) = 7$, $c(k) = 10$, $c(a) = 12$, $c(b) = 13$, $c(h_0) = 8$, $c(h_1) = 11$, $c(h_2) = 9$, $c(h_3) = 6$, $c(h_4) = 8$ y $c(h_5) = 11$, $c(h_6) = 14$, $c(h_7) = 14$.

Aplicando el algoritmo obtenemos

0. $L = \{s\}$, $c = \infty$.
1. $L = \emptyset$, $c(s) = 3$.
2. $L = \{x, y\}$, goto 4.
4. $L \neq \emptyset$, goto 1.
1. $L = \{x\}$, $c(y) = 4 < \infty = c$.
2. $L = \{x, v, z, w\}$, goto 4.
4. $L \neq \emptyset$, goto 1.
1. $L = \{x, v, z\}$, $c(w) = 7 < \infty = c$.
2. $L = \{x, v, z, h_4, h_5\}$, goto 4.
4. $L \neq \emptyset$, goto 1.
1. $L = \{x, v, z, h_4\}$, $c(h_5) = 11 < \infty = c$.
3. $h = h_5$, $c = 11$.
4. $L \neq \emptyset$, goto 1.
1. $L = \{x, v, z\}$, $c(h_4) = 8 < 11 = c$.
3. $h = h_4$, $c = 8$.
4. $L \neq \emptyset$, goto 1.
1. $L = \{x, v\}$, $c(z) = 4 < 8 = c$.

2. $L = \{x, v, h_3\}$, goto 4.
4. $L \neq \emptyset$, goto 1.
1. $L = \{x, v\}$, $c(h_3) = 6 < 8 = c$.
3. $h = h_3$, $c = 6$.
4. $L \neq \emptyset$, goto 1.
1. $L = \{x\}$, $c(v) = 8 \geq 6 = c$, goto 4.
4. $L \neq \emptyset$, goto 1.
1. $L = \emptyset$, $c(x) = 4 < 6 = c$.
2. $L = \{u\}$, goto 4.
4. $L \neq \emptyset$, goto 1.
1. $L = \emptyset$, $c(u) = 7 \geq 6 = c$, goto 4.
4. $L \neq \emptyset$.
5. STOP.

El siguiente gráfico es una interpretación de los pasos seguidos por el algoritmo en nuestro ejemplo.



Los vértices son examinados en el orden en que indican las flechas punteadas descendentes. Cada flecha punteada ascendente indica un backtracking. El valor de c en cada iteración del algoritmo es el costo de la hoja más barata encontrada hasta el momento y h guarda la información de cuál es la hoja cuyo costo es c (cuando el algoritmo encuentra una hoja de costo menor que el valor de c presente en ese momento, actualiza c y h).

Si al examinar un nodo i resulta que $c(i) \geq c$ entonces toda su descendencia es podada ya que ninguna hoja que sea un descendiente de i puede tener costo menor que c, es decir, costo menor que el de la hoja más barata hallada hasta ese momento.

Como una hoja no es examinada por el algoritmo sólo cuando es seguro que no puede ser una solución óptima entonces en alguna iteración del algoritmo una hoja de mínimo costo es examinada. Cuando el algoritmo encuentra la primer hoja de mínimo costo los valores de h y c son actualizados ya que su costo es menor que el valor de c que estaba presente en ese momento, y a partir de allí c y h permanecen constantes hasta terminar el algoritmo ya que ninguna otra hoja encontrada más tarde puede tener costo menor que el de esta hoja de mínimo costo. Luego, al terminar el algoritmo la hoja que se encuentra almacenada en h es una hoja de mínimo costo y $c = c(h)$.
Luego, en este caso, la hoja de mínimo costo es $h = h_3$, con costo $c = 6$.

En este ejemplo la función $c(x)$ estaba dada explícitamente. Esto no es así en general, sino que es una función que se deberá calcular.

En el caso en que la función c satisfaga $c(x) \geq c(y)$ para toda rama (x, y), podemos resolver el problema de hallar una hoja de máximo costo utilizando el algoritmo que resulta de reemplazar, en el paso 0., $c = \infty$ por $c = -\infty$ y, en el paso 1., la condición $c(x) \geq c$ por la condición $c(x) \leq c$, es decir, el algoritmo

0'. $L = \{s\}$, $c = -\infty$.
1'. Sea x es el último vértice que ingresó en L, $L = L - \{x\}$. Calcular $c(x)$. Si $c(x) \leq c$, goto 4.
2'. Si x no es una hoja, $L = L \cup \{$ hijos de $x\}$, goto 4.
3'. $h = x$, $c = c(x)$.
4'. Si $L \neq \emptyset$ goto 1.
5'. STOP.

4. Aplicación al problema de programación lineal entera.

Consideremos el problema de programación lineal

$$\begin{aligned} &\min cx \\ &Ax \leq b \\ &0 \leq x_j \leq u_j \qquad (1 \leq j \leq n) \end{aligned} \tag{12}$$

donde $u_j < \infty$ para todo j y consideremos el problema de programación lineal entera que resulta de agregar a (12) la restricción adicional

$$x_i \text{ entero} \qquad (1 \leq i \leq m) \tag{13}$$

donde $m \leq n$ es un número natural dado. Resolveremos este problema de programación lineal entera generando un árbol binario dirigido cuya raíz será el problema (12) y los restantes vértices serán subproblemas que resulten de agregar a (12) ciertas resticciones del tipo $x_j \leq k$ o $x_j \geq k+1$ para algunos j. Las hojas serán los subproblemas cuyas solución óptima verifica la restricción (13) y los subproblemas que no sean factibles. Para cada vértice u definiremos $c(u)$ como el valor del funcional en la solución óptima del subproblema u si este es factible y definiremos $c(u) = \infty$ si el subproblema u no es factible. La función $c(u)$ verificará $c(u) \leq c(v)$ para toda rama (u, v) del árbol. Usaremos el método de branch and bound para encontrar una hoja de mínimo costo y eso nos dará la solución del problema de programación lineal entera.

Observación 4.1. La condición $0 \leq x_j \leq u_j$ asegura que el poliedro definido por las restricciones es acotado. Como también es cerrado (pues es intersección finita de de semiespacios del tipo $\{x \,/\, a.x \leq b\}$ o $\{x \,/\, a.x \geq b\}$ que son cerrados) entonces es compacto. Luego, si (12) es factible entonces siempre existe una solución óptima ya que el funcional es una función contínua y toda función contínua sobre un compacto alcanza su máximo y su mínimo. Lo mismo vale para cualquier subproblema que resulte de agregar a (12) restricciones del tipo $x_j \leq k$ o $x_j \geq k+1$ para algunos j.

Observación 4.2. Si z^* es el valor del funcional en una solución óptima de un problema u de programación lineal y z es el valor del funcional en el óptimo de cualquier subproblema v que se obtenga agregándole una restricción a u entonces $z^* \leq z$ ya que toda solución óptima del subproblema v es una solución factible del problema u. Notemos también que si el problema fuese maximizar el funcional en lugar de minimizarlo entonces tendríamos que $z^* \geq z$.
Veamos ahora con más detalle cómo construír el árbol.
La raíz s del árbol es el problema (12). Notemos que toda solución factible del problema de programación lineal entera es una solución factible de s. Supongamos que hemos construído una parte del árbol donde cada vértice es un subproblema que resulta de agregar a (12) restricciones del tipo $x_j \leq k$ o $x_j \geq k+1$ para algunos j, donde k es un entero no negativo y tal que toda solución factible del problema de programación

lineal entera es una solución factible de alguna hoja de este subárbol. Para cada subproblema u que sea una hoja del subárbol que tenemos construído hasta ahora (es decir, cada vértice u que no tenga hijos) hacemos lo siguiente:

Si u tiene una solución óptima que no satisface (13), sea j $(1 \leq j \leq m)$ el primer índice para el cual x_j no es entero y sea r un entero tal que $r < x_j < r+1$. Entonces creamos dos hijos de u, uno agregando al problema u la restricción $x_j \leq r$ y el otro agregando a u la restricción $x_j \geq r+1$. Notemos que si una solución factible del problema de programación lineal entera es una solución factible de u entonces es una solución factible de alguno de los hijos de u ya que, como r es entero, si la j-ésima coordenada de esta solución es entera entonces o bien es menor o igual que r o bien es mayor o igual que $r+1$.

Si en cambio u tiene una solución óptima que satisface (13) o si no tiene soluciones factibles (notar que, por la observación 4.1., si u es factible entonces necesariamente tiene una solución óptima) entonces no creamos ningún hijo de u.

De esta manera obtenemos un árbol binario dirigido con raíz s cuyas hojas son los subproblemas cuyas solución óptima verifica la restricción (13) y los subproblemas que no sean factibles. Además, toda solución factible del problema de programación lineal entera es una solución factible de alguna hoja del árbol.

Para cada vértice u de este árbol definimos $c(u)$ como el valor del funcional en la solución óptima del subproblema u si este es factible y $c(u) = \infty$ si no. Debido a la forma en que hemos construído el árbol esta función satisface $c(u) \leq c(v)$ para toda rama (u, v) del árbol ya que si (u, v) es una rama del árbol entonces v es un subproblema que se obtuvo agreagando una restricción a u (ver observación 4.2.). De esta manera, el costo de una hoja que corresponda a un problema factible será el valor del funcional en una solución factible del problema de programación lineal entera. Como las soluciones factibles del problema de programación lineal entera son las soluciones factibles de cada una de las hojas del árbol entonces una hoja de mínimo costo será una solución óptima del problema de programación lineal entera.

Para hallar una hoja de mínimo costo utilizaremos el método de branch and bound, pero en lugar de generar todo el árbol y luego aplicar el algoritmo, en cada paso del algoritmo calcularemos sólo los vértices y los costos que necesitemos. Es decir, cada vez que realizamos el paso 1. debemos resolver el subproblema x para el último vértice x que ingresó en L y en el caso en que no sea una hoja generamos los dos hijos de x que ingresaremos en L en el siguiente paso. De esta manera evitamos calcular las ramas que luego el algoritmo podaría.

En el caso en que el problema sea de maximización entonces la función $c(u)$ se define como el valor del funcional en el subproblema u si u es factible y como $c(u) = -\infty$ si no y satisface $c(u) \geq c(v)$ para toda rama (u, v) del árbol (ver observación 4.2.). En este caso lo que buscamos es una hoja de máximo costo.

Ejemplo 4.3. Resolveremos el problema de programación lineal entera

$$\begin{aligned} &\max\ 3x_1 + 3x_2 + 13x_3 \\ &-3x_1 + 6x_2 + 7x_3 \leq 8 \\ &6x_1 - 3x_2 + 7x_3 \leq 8 \\ &0 \leq x_i \leq 5 \quad x_i \text{ entero} \end{aligned}$$

0'. $L = \{s\}$, $c = -\infty$

donde la raíz del árbol es el problema

$$\begin{aligned} &\max\ 3x_1 + 3x_2 + 13x_3 \\ &-3x_1 + 6x_2 + 7x_3 \leq 8 \\ &6x_1 - 3x_2 + 7x_3 \leq 8 \\ &0 \leq x_i \leq 5 \end{aligned} \qquad (s)$$

1'. $L = \emptyset$. Calculamos $c(s)$. Para ello resolvemos el problema s. En este caso una solución óptima es $(x_1, x_2, x_3) = (\frac{8}{3}, \frac{8}{3}, 0)$ y el valor del funcional en ella es 16. Luego $c(s) = 16$. Como $c(s)$ es mayor que $c = -\infty$ vamos a 2'.

2'. s no es una hoja, ya que x_1 no es entero. Como $2 < x_1 < 3$, los hijos de s son los subproblemas u_1 y u_2 que resultan de agregar al problema s las restricciones $x_1 \leq 2$ y $x_1 \geq 3$ respectivamente. Es decir,

$$\begin{array}{l} \max 3x_1 + 3x_2 + 13x_3 \\ -3x_1 + 6x_2 + 7x_3 \leq 8 \\ 6x_1 - 3x_2 + 7x_3 \leq 8 \\ \quad x_1 \leq 2 \\ \; 0 \leq x_i \leq 5 \end{array} \qquad (u_1) \qquad \text{y} \qquad \begin{array}{l} \max 3x_1 + 3x_2 + 13x_3 \\ -3x_1 + 6x_2 + 7x_3 \leq 8 \\ 6x_1 - 3x_2 + 7x_3 \leq 8 \\ \quad x_1 \geq 3 \\ \; 0 \leq x_i \leq 5 \end{array} \qquad (u_2)$$

Actualizamos L en la forma $L = \{u_1, u_2\}$

4'. $L \neq \emptyset$, vamos a 1'.

1'. $L = \{u_1\}$. Calculamos $c(u_2)$. En este caso u_2 no es factible. Luego, $c(u_2) = -\infty$. Como $c(u_2)$ es menor o igual que $c = -\infty$ vamos a 4'.

4'. $L \neq \emptyset$, vamos a 1'.

1'. $L = \emptyset$. Calculamos $c(u_1)$. Para ello resolvemos u_1. Una solución óptima es $(x_1, x_2, x_3) = (2, 2, \frac{2}{7})$ y el valor del funcional en ella es $15 + \frac{5}{7}$, de donde $c(u_1) = 15 + \frac{5}{7}$. Como $c(u_1)$ es mayor que $c = -\infty$ vamos a 2'.

2'. u_1 no es una hoja, ya que x_3 no es entero. Como $0 < x_3 < 1$, los hijos de u_1 son los subproblemas u_3 y u_4 que resultan de agregar al problema u_1 las restricciones $x_3 \leq 0$ y $x_3 \geq 1$ respectivamente. Es decir,

$$\begin{array}{l} \max 3x_1 + 3x_2 + 13x_3 \\ -3x_1 + 6x_2 + 7x_3 \leq 8 \\ 6x_1 - 3x_2 + 7x_3 \leq 8 \\ \quad x_1 \leq 2 \\ \quad x_3 \leq 0 \\ \; 0 \leq x_i \leq 5 \end{array} \qquad (u_3) \qquad \text{y} \qquad \begin{array}{l} \max 3x_1 + 3x_2 + 13x_3 \\ -3x_1 + 6x_2 + 7x_3 \leq 8 \\ 6x_1 - 3x_2 + 7x_3 \leq 8 \\ \quad x_1 \leq 2 \\ \quad x_3 \geq 1 \\ \; 0 \leq x_i \leq 5 \end{array} \qquad (u_4)$$

Actualizamos L en la forma $L = \{u_3, u_4\}$

4'. $L \neq \emptyset$, vamos a 1'.

1'. $L = \{u_3\}$. Calculamos $c(u_4)$. Una solución óptima de u_4 es $(x_1, x_2, x_3) = (\frac{1}{3}, \frac{1}{3}, 1)$ y el valor del funcional en ella es $c(u_4) = 15$. Como $c(u_4)$ es mayor que $c = -\infty$ vamos a 2'.

2'. u_4 no es una hoja ya que x_1 no es entero. Como $0 < x_1 < 1$ los hijos de u_4 son los subproblemas u_5 y u_6 que resultan de agregar al problema u_4 las restricciones $x_1 \leq 0$ y $x_1 \geq 1$ respectivamente. Es decir,

$$\begin{array}{l} \max 3x_1 + 3x_2 + 13x_3 \\ -3x_1 + 6x_2 + 7x_3 \leq 8 \\ 6x_1 - 3x_2 + 7x_3 \leq 8 \\ \quad x_1 \leq 2 \\ \quad x_3 \geq 1 \\ \quad x_1 \leq 0 \\ \; 0 \leq x_i \leq 5 \end{array} \qquad (u_5) \qquad \text{y} \qquad \begin{array}{l} \max 3x_1 + 3x_2 + 13x_3 \\ -3x_1 + 6x_2 + 7x_3 \leq 8 \\ 6x_1 - 3x_2 + 7x_3 \leq 8 \\ \quad x_1 \leq 2 \\ \quad x_3 \geq 1 \\ \quad x_1 \geq 1 \\ \; 0 \leq x_i \leq 5 \end{array} \qquad (u_6)$$

Actualizamos L en la forma $L = \{u_3, u_5, u_6\}$

4'. $L \neq \emptyset$, vamos a 1'.

1'. $L = \{u_3, u_5\}$. Calculamos $c(u_6)$. El problema u_6 no es factible. Luego, $c(u_6) = -\infty$. Como $c(u_6)$ es menor o igual que $c = -\infty$ vamos a 4'.

4'. $L \neq \emptyset$, vamos a 1'.

1'. $L = \{u_3\}$. Calculamos $c(u_5)$. Una solución óptima de u_5 es $(x_1, x_2, x_3) = (0, 0, \frac{8}{7})$ y el valor del funcional en ella es $c(u_5) = 14 + \frac{6}{7}$. Como $c(u_5)$ es mayor que $c = -\infty$ vamos a 2'.

2'. u_5 no es una hoja ya que x_3 no es entero. Como $1 < x_1 < 2$ los hijos de u_5 son los subproblemas u_7 y u_8 que resultan de agregar al problema u_5 las restricciones $x_3 \leq 1$ y $x_3 \geq 2$ respectivamente. Es decir,

$$\begin{array}{rl}
\max & 3x_1 + 3x_2 + 13x_3 \\
& -3x_1 + 6x_2 + 7x_3 \leq 8 \\
& 6x_1 - 3x_2 + 7x_3 \leq 8 \\
& x_1 \leq 2 \\
& x_3 \geq 1 \\
& x_1 \leq 0 \\
& x_3 \leq 1 \\
& 0 \leq x_i \leq 5
\end{array} \quad (u_7) \qquad \text{y} \qquad
\begin{array}{rl}
\max & 3x_1 + 3x_2 + 13x_3 \\
& -3x_1 + 6x_2 + 7x_3 \leq 8 \\
& 6x_1 - 3x_2 + 7x_3 \leq 8 \\
& x_1 \leq 2 \\
& x_3 \geq 1 \\
& x_1 \leq 0 \\
& x_3 \geq 2 \\
& 0 \leq x_i \leq 5
\end{array} \quad (u_8)$$

Actualizamos L en la forma $L = \{u_3, u_7, u_8\}$

4'. $L \neq \emptyset$, vamos a 1'.

1'. $L = \{u_3, u_7\}$. Calculamos $c(u_8)$. El problema u_8 no es factible. Luego, $c(u_8) = -\infty$. Como $c(u_8)$ es menor o igual que $c = -\infty$ vamos a 4'.

4'. $L \neq \emptyset$, vamos a 1'.

1'. $L = \{u_3\}$. Calculamos $c(u_7)$. Una solución óptima de u_7 es $(x_1, x_2, x_3) = (0, 0, 1)$ y el valor del funcional en ella es $c(u_7) = 13$. Como $c(u_7)$ es mayor que $c = -\infty$ vamos a 2'.

2'. Como u_7 es una hoja, no modificamos L y vamos a 3'.

3'. $h = u_7$, $c = c(u_7) = 13$

4'. $L \neq \emptyset$, vamos a 1'.

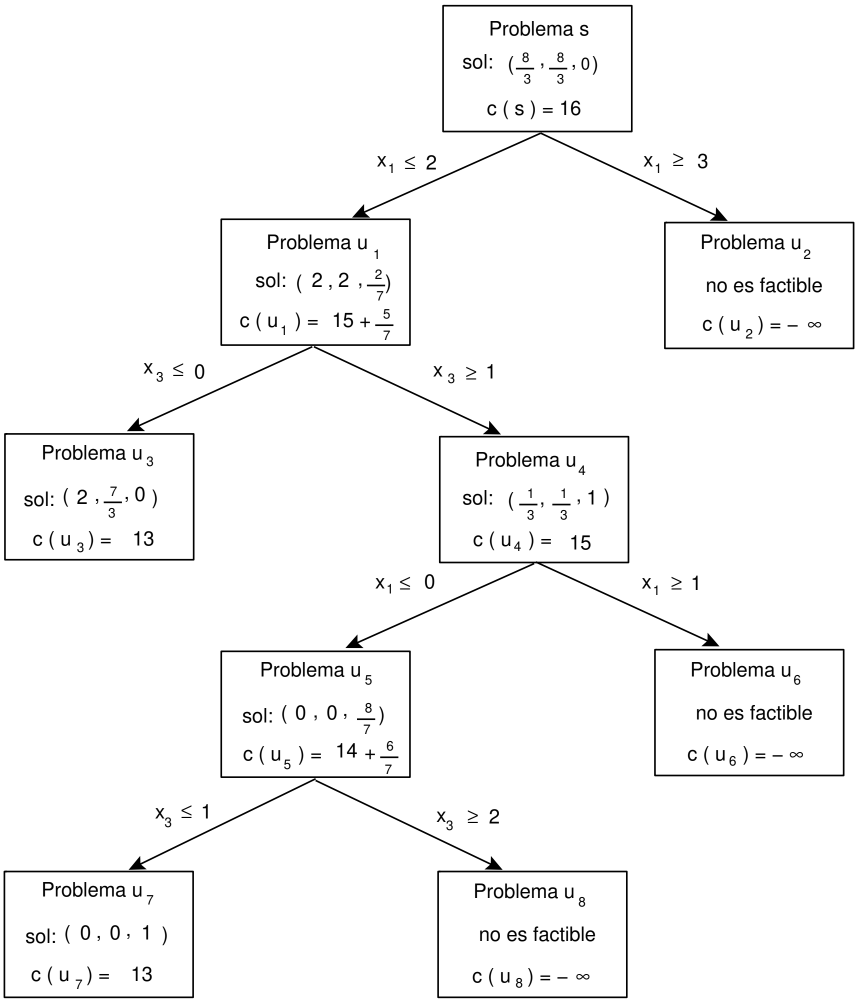
1'. $L = \emptyset$. Calculamos $c(u_3)$. Una solución óptima de u_3 es $(x_1, x_2, x_3) = (2, \frac{7}{3}, 0)$ y el valor del funcional en ella es $c(u_3) = 13$. Como $c(u_3)$ es menor o igual que $c = 13$ vamos a 4'.

4'. Como $L = \emptyset$, vamos a 5'.

5'. STOP.

Luego, la hoja de máximo costo es $h = u_7$ con costo $c = 13$. Esto significa que una solución óptima del problema de programación lineal entera que queríamos resolver es $(x_1, x_2, x_3) = (0, 0, 1)$ y el valor del funcional en ella es $c(u_7) = 13$.

El subárbol generado por el algoritmo es



5. Aplicación de branch and bound al problema del viajante.

Como vimos, el problema del viajante puede plantearse por programación lineal entera (ver ejemplo 1.9.) y por lo tanto resolverse utilizando el método de branch and bound en la forma descripta en la sección anterior. Veremos ahora otro algoritmo para resolver este problema, esencialmente distinto pero que también utiliza el método de branch and bound.

Sea $G = (V, E)$ un grafo completo dirigido, donde $V = \{1, \ldots, n\}$ y donde cada rama (i, j) de G tiene asignado un costo c_{ij} tal que $0 \leq c_{ij} \leq \infty$.

Recordemos que un circuito Hamiltoniano $\mathcal{C}$ es un ciclo dirigido en G que pasa por cada vértice una y sólo una vez. El costo $c(\mathcal{C})$ del circuito se define como la suma de los costos de las ramas que lo forman, es decir,

$$c(\mathcal{C}) = \sum_{(i,j) \in \mathcal{C}} c_{ij}$$

El problema consiste en hallar un circuito Hamiltoniano $\mathcal{C}$ de mínimo costo.

Como antes, para resolver este problema generaremos un árbol binario dirigido con raíz y definiremos una función $c(u)$ conveniente que satisfaga $c(u) \leq c(v)$ para toda rama (u, v) de manera que nuestro problema

se traduzca en hallar una hoja de mínimo costo. De ahora en más, la palabra circuito significará circuito Hamiltoniano.
La raíz del árbol será el problema del viajante, es decir, hallar un circuito $\mathcal{C}$ tal que $c(\mathcal{C})$ sea mínimo. Supongamos que hemos construído una parte del árbol donde cada vértice es un subproblema que resulta de agregar a su vértice padre la restricción $(i,j) \in \mathcal{C}$ o la restricción $(i,j) \notin \mathcal{C}$, para cierta rama $(i,j) \in E$.
Para cada subproblema u que sea una hoja del subárbol que tenemos construído hasta ahora hacemos lo siguiente:
Sea $A_u = \{e \in E \,/\,$ "$e \in \mathcal{C}$" es una restricción de $u\}$.
Si $\#A_u < n-1$ entonces elegimos convenientemente una rama $(i,j) \in E$ y generamos dos hijos de u, uno agregando a u la restricción $(i,j) \in \mathcal{C}$ y el otro agregando a u la restricción $(i,j) \notin \mathcal{C}$.
Si, en cambio, $\#A_u = n-1$ entonces no generamos ningún hijo de u.
Veamos en un ejemplo cuál es la manera en que se eligen las ramas (i,j) que definen las restricciones de los subproblemas, cómo calcular el costo de cada vértice del árbol que se genera y la razón por la cual el problema se traduce en encontrar una hoja de mínimo costo.

En lo que sigue, por número entenderemos un número real o infinito. Sea G el grafo completo dirigido con los $n = 7$ vértices $1, 2, \dots 7$. Supongamos que el costo de cada rama (i,j) de G $(1 \leq i,j \leq 7)$ está dado por la matriz

$$||c_{ij}|| = \begin{pmatrix} \infty & 3 & 93 & 13 & 33 & 9 & 57 \\ 4 & \infty & 77 & 42 & 21 & 16 & 34 \\ 45 & 17 & \infty & 36 & 16 & 28 & 25 \\ 39 & 90 & 80 & \infty & 56 & 7 & 91 \\ 28 & 46 & 88 & 33 & \infty & 25 & 57 \\ 3 & 88 & 18 & 46 & 92 & \infty & 7 \\ 44 & 26 & 33 & 27 & 84 & 39 & \infty \end{pmatrix}$$

Paso 1. Definimos la raíz s del árbol como el problema de hallar un circuito $\mathcal{C}$ tal que $c(\mathcal{C})$ sea mínimo.

Paso 2. Calculamos el costo y generamos los hijos de cada nodo u que sea una hoja del subárbol que tenemos construído hasta ahora y que satisfaga que $\#A_u < n-1$.
El subárbol que tenemos construído hasta ahora tiene un solo nodo (la raíz s), que es una hoja. Como $\#A_s = 0 < 6 = n-1$ pues $A_s = \emptyset$, para calcular $c(s)$ y generar los hijos de s procedemos de la siguiente manera:

Asociamos a s la matriz $||c_{ij}(s)|| = ||c_{ij}||$ y consideremos la matriz D_1 que se obtiene restando un número no negativo k_1 a cada coeficiente de la fila 1 de $||c_{ij}(s)||$ de manera que cada coeficiente de la nueva matriz sea no negativo. Como cada circuito $\mathcal{C}$ pasa una sola vez por el vértice 1 entonces existe un único j, $1 \leq j \leq 7,\ j \neq 1$ tal que la rama $(1,j) \in \mathcal{C}$. Luego el costo de cualquier circuito $\mathcal{C}$ calculado utilizando la nueva matriz de costos D_1 es igual a $c(\mathcal{C}) - k_1$ ya que la única rama cuyo costo sufrió una modificación es la rama $(1,j)$ cuyo costo fue disminuido en k_1.
Hacemos esto con el máximo valor posible de k_1, en este caso $k_1 = 3$, para que la nueva matriz D_1 tenga al menos un coeficiente nulo en la primera fila. Ahora consideremos la matriz D_2 que se obtiene restando un número no negativo k_2 a cada coeficiente de la fila 2 de D_1 de manera que cada coeficiente de la nueva matriz sea no negativo. Como cada circuito $\mathcal{C}$ pasa una sola vez por el vértice 2 entonces existe un único j, $1 \leq j \leq 7,\ j \neq 2$ tal que la rama $(2,j) \in \mathcal{C}$. Luego el costo de cualquier circuito $\mathcal{C}$ calculado utilizando la nueva matriz de costos D_2 es igual al costo de $\mathcal{C}$ calculado usando la matriz de costos D_1 menos k_2, lo que es igual a $c(\mathcal{C}) - k_1 - k_2$.
Hacemos esto con el máximo valor posible de k_2, en este caso $k_2 = 4$. De esta manera D_2 tiene al menos un coeficiente nulo en cada una de las dos primeras filas.

Repitiendo este procedimiento para las restantes filas obtenemos la matriz

$$D_n = D_7 = \begin{pmatrix} \infty & 0 & 90 & 10 & 30 & 6 & 54 \\ 0 & \infty & 73 & 38 & 17 & 12 & 30 \\ 29 & 1 & \infty & 20 & 0 & 12 & 9 \\ 32 & 83 & 73 & \infty & 49 & 0 & 84 \\ 3 & 21 & 63 & 8 & \infty & 0 & 32 \\ 0 & 85 & 15 & 43 & 89 & \infty & 4 \\ 18 & 0 & 7 & 1 & 58 & 13 & \infty \end{pmatrix}$$

que tiene al menos un coeficiente nulo en cada fila. El costo de cualquier circuito $\mathcal{C}$ calculado utilizando la nueva matriz de costos D_n es $c(\mathcal{C}) - k_1 - k_2 - \cdots - k_7$, donde $k_1 = 3$, $k_2 = 4$, $k_3 = 16$, $k_4 = 7$, $k_5 = 25$, $k_6 = 3$ y $k_7 = 26$. Ahora, para cada j, $(1 \leq j \leq 7)$ restamos un número no negativo r_j a cada coeficiente de la columna j de D_n de manera que cada coeficiente de la nueva matriz sea no negativo. Eligiendo r_j el máximo valor posible, en este caso $r_1 = 0$, $r_2 = 0$, $r_3 = 7$, $r_4 = 1$, $r_5 = 0$, $r_6 = 0$ y $r_7 = 4$, obtenemos la matriz

$$||c'_{ij}(s)|| = \begin{pmatrix} \infty & 0 & 83 & 9 & 30 & 6 & 50 \\ 0 & \infty & 66 & 37 & 17 & 12 & 26 \\ 29 & 1 & \infty & 19 & 0 & 12 & 5 \\ 32 & 83 & 66 & \infty & 49 & 0 & 80 \\ 3 & 21 & 56 & 7 & \infty & 0 & 28 \\ 0 & 85 & 8 & 42 & 89 & \infty & 0 \\ 18 & 0 & 0 & 0 & 58 & 13 & \infty \end{pmatrix}$$

que tiene al menos un coeficiente nulo en cada fila y en cada columna. El costo de cualquier circuito $\mathcal{C}$ calculado utilizando la nueva matriz de costos $||c'_{ij}(s)||$ definido por

$$c'(\mathcal{C}, s) = \sum_{(i,j) \in \mathcal{C}} c'_{ij}(s)$$

satisface $c'(\mathcal{C}, s) = c(\mathcal{C}) - k_1 - \cdots - k_7 - r_1 - \cdots - r_7$. Definimos $c(s) = k_1 + \cdots + k_n + r_1 + \cdots + r_n$. En este caso, $c(s) = 96$. De esta manera, $c(\mathcal{C}) = c'(\mathcal{C}, s) + c(s) = c'(\mathcal{C}, s) + 96$.

Observación 5.1. Para cualquier circuito $\mathcal{C}$ se verifica que $c(\mathcal{C}) \geq c(s)$. En efecto, como los coeficientes de $||c'_{ij}(s)||$ son no negativos entonces $c'(\mathcal{C}, s) \geq 0$. Luego, $c(\mathcal{C}) = c'(\mathcal{C}, s) + c(s) \geq c(s)$.

Ahora generamos dos hijos u_1 y v_1 de s, eligiendo una rama $(i_1, j_1) \in E$ tal que el coeficiente $c'_{i_1 j_1}(s)$ correspondiente a la fila i_1 y a la columna j_1 de $||c'_{ij}(s)||$ sea nulo, y tomando como u_1 el subproblema que resulta de agregar al problema s la condición de que (i_1, j_1) pertenezca al circuito y como v_1 el que resulta de agregar a s la condición de que (i_1, j_1) no pertenezca al circuito. Supongamos que en este caso elegimos $(i_1, j_1) = (4, 6)$. Entonces el subproblema u_1 es el de hallar un circuito de mínimo costo que contenga a la rama $(4, 6)$ y v_1 el de hallar un circuito de mínimo costo que no la contenga.

Paso 3. Calculamos el costo y generamos los hijos de cada nodo u que sea una hoja del subárbol que tenemos construído hasta ahora y que satisfaga que $\#A_u < n - 1$.
El subárbol que tenemos construído hasta ahora consiste de la raíz s y sus dos hijos, u_1 y v_1. Las hojas son, por lo tanto, u_1 y v_1.
Como $\#A_{u_1} = 1 < 6 = n - 1$ pues $A_{u_1} = \{(i_1, j_1)\}$, para calcular $c(u_1)$ y luego generar sus hijos le asociamos al problema u_1 la matriz de costos $||c_{ij}(u_1)||$ que se obtiene reemplazando en $||c'_{ij}(s)||$ el coeficiente $c'_{j_1 i_1}(s)$ por ∞ y eliminando la fila i_1 y la columna j_1.
Si $\mathcal{C}$ es un circuito que contiene a la rama (i_1, j_1) entonces para todo $j \neq j_1$ la rama (i_1, j) no puede pertenecer a $\mathcal{C}$, para todo $i \neq i_1$ la rama (i, j_1) no puede pertenecer a $\mathcal{C}$. Como además $c'_{i_1 j_1} = 0$ eso significa que no necesitamos guardar la información sobre la fila i_1 y la columna j_1 de $||c'_{ij}(s)||$ y por eso

son eliminadas. Como además la rama (j_1, i_1) no puede pertenecer al circuito, para evitar que esto pudiera ocurrir reemplazamos el coeficiente $c'_{j_1 i_1}(s)$ por ∞.
Como eliminaremos una fila y una columna, utilizaremos una tabla de doble entrada en lugar de la notación matricial con el objeto de no perder la información de a cuáles ramas corresponden los distintos costos (el costo de cada rama (i, j) según $||c_{ij}(u_1)||$ es el valor correspondiente a la fila i y la columna j).

En nuestro ejemplo la matriz $||c_{ij}(u_1)||$ correspondiente a u_1 es

	col 1	*col* 2	*col* 3	*col* 4	*col* 5	*col* 7
fila 1	∞	0	83	9	30	50
fila 2	0	∞	66	37	17	26
fila 3	29	1	∞	19	0	5
fila 5	3	21	56	7	∞	28
fila 6	0	85	8	∞	89	0
fila 7	18	0	0	0	58	∞

En este caso, el costo de la rama $(5, 7)$ según esta matriz es 28, es decir, $c_{57}(u_1) = 28$.
Ahora generamos la matriz $||c'_{ij}(u_1)||$ que se obtiene restando a cada coeficiente de la fila i de $||c_{ij}(u_1)||$ el máximo número no negativo k_i y luego restando a cada coeficiente de la columna j el máximo número no negativo r_j, de manera que cada coeficiente de la nueva matriz resulte ser no negativo y definimos $c(u_1) = c(s) + \sum k_i + \sum r_j$. Luego $c(u_1) \geq c(s)$. Notemos que $||c'_{ij}(u_1)||$ tiene un coeficiente nulo en cada fila y cada columna.
En nuestro caso resulta que $k_5 = 3$, $k_i = 0$ para $i \neq 5$ y $r_j = 0$ para todo j, de donde $||c'_{ij}(u_1)||$ es la matriz

	col 1	*col* 2	*col* 3	*col* 4	*col* 5	*col* 7
fila 1	∞	0	83	9	30	50
fila 2	0	∞	66	37	17	26
fila 3	29	1	∞	19	0	5
fila 5	0	18	53	4	∞	25
fila 6	0	85	8	∞	89	0
fila 7	18	0	0	0	58	∞

y $c(u_1) = c(s) + 3 = 96 + 3 = 99$.

Si $\mathcal{C}$ es un circuito que contiene a la rama (i_1, j_1) (es decir, una solución factible de u_1) entonces

$$\begin{aligned} c(\mathcal{C}) &= c'(\mathcal{C}, s) + c(s) = \\ &= \sum_{(i,j)\in\mathcal{C}} c'_{ij}(s) + c(s) = \\ &= \sum_{\substack{(i,j)\in\mathcal{C} \\ (i,j)\neq(i_1,j_1)}} c'_{ij}(s) + c(s) \end{aligned}$$

ya que (i_1, j_1) fue elegido verificando $c'_{i_1 j_1}(s) = 0$.
Sean

$$c(\mathcal{C}, u_1) = \sum_{\substack{(i,j)\in\mathcal{C} \\ (i,j)\neq(i_1,j_1)}} c_{ij}(u_1)$$

y

$$c'(\mathcal{C}, u_1) = \sum_{\substack{(i,j)\in\mathcal{C} \\ (i,j)\neq(i_1,j_1)}} c'_{ij}(u_1)$$

Notemos que $c(\mathcal{C}, u_1)$ y $c'(\mathcal{C}, u_1)$ están bien definidos pues si $\mathcal{C}$ es un circuito que contiene a la rama (i_1, j_1) entonces para todo $(i, j) \in \mathcal{C}$, $(i, j) \neq (i_1, j_1)$ vale que $i \neq i_1$, que $j \neq j_1$ y que $(i, j) \neq (j_1, i_1)$.

Como $||c_{ij}(u_1)||$ fue construída eliminando la fila i_1 y la columna j_1 de $||c'_{ij}(s)||$ y reemplazando $c'_{j_1 i_1}(s)$ por ∞ entonces para todo $(i,j) \in \mathcal{C}$, $(i,j) \neq (i_1, j_1)$ existen los coeficientes $c_{ij}(u_1)$ y $c'_{ij}(u_1)$ y vale

$$\sum_{\substack{(i,j)\in\mathcal{C} \\ (i,j)\neq(i_1,j_1)}} c'_{ij}(s) = \sum_{\substack{(i,j)\in\mathcal{C} \\ (i,j)\neq(i_1,j_1)}} c_{ij}(u_1) = c(\mathcal{C}, u_1)$$

Luego, $c(\mathcal{C}) = c(\mathcal{C}, u_1) + c(s)$.
Con el mismo razonamiento utilizado para ver que $c'(\mathcal{C}, s) = c(\mathcal{C}) - 96$ puede verse que $c'(\mathcal{C}, u_1) = c(\mathcal{C}, u_1) - 3$ (esto se debe a que $||c'_{ij}(u_1)||$ fue construída a partir de $||c_{ij}(u_1)||$ con un procedimiento análogo al utilizado para construír $||c'_{ij}(s)||$ a partir de $||c_{ij}(s)|| = ||c_{ij}||$).
Luego

$$\begin{aligned} c(\mathcal{C}) &= c(\mathcal{C}, u_1) + c(s) = c'(\mathcal{C}, u_1) + 3 + c(s) = c'(\mathcal{C}, u_1) + 3 + 96 = \\ &= c'(\mathcal{C}, u_1) + 99 = c'(\mathcal{C}, u_1) + c(u_1) \end{aligned}$$

Por lo tanto, $c(\mathcal{C}) = c'(\mathcal{C}, u_1) + c(u_1)$. Además, como $A_{u_1} = \{(i_1, j_1)\}$ entonces $c'(\mathcal{C}, u_1) = \sum_{\substack{(i,j)\in\mathcal{C} \\ (i,j)\notin A_{u_1}}} c'_{ij}(u_1)$.

Ahora cosideremos el problema v_1. Como $\#A_{v_1} = 0 < 6 = n - 1$ pues $A_{v_1} = \emptyset$, para calcular $c(v_1)$ y luego generar sus hijos le asociamos al problema v_1 la matriz de costos $||c_{ij}(v_1)||$ que se obtiene reemplazando en $||c'_{ij}(s)||$ el coeficiente $c'_{i_1 j_1}(s)$ por ∞. Esto garantiza que la rama (i_1, j_1) no pertenecerá al circuito (recordemos que v_1 es el problema de hallar un circuito de mínimo costo que no contenga esa rama).

En nuestro ejemplo, la matriz $||c_{ij}(v_1)||$ correspondiente a v_1 es

	col 1	*col* 2	*col* 3	*col* 4	*col* 5	*col* 6	*col* 7
fila 1	∞	0	83	9	30	6	50
fila 2	0	∞	66	37	17	12	26
fila 3	29	1	∞	19	0	12	5
fila 4	32	83	66	∞	49	∞	80
fila 5	3	21	56	7	∞	0	28
fila 6	0	85	8	∞	89	∞	0
fila 7	18	0	0	0	58	13	∞

Ahora generamos la matriz $||c'_{ij}(v_1)||$ que se obtiene restando a cada coeficiente de la fila i de $||c_{ij}(v_1)||$ el máximo número no negativo k_i y luego restando a cada coeficiente de la columna j el máximo número no negativo r_j, de manera que cada coeficiente de la nueva matriz resulte ser no negativo y definimos $c(v_1) = c(s) + \sum k_i + \sum r_j$. Luego, $c(v_1) \geq c(s)$. Notemos que $||c'_{ij}(v_1)||$ tiene un coeficiente nulo en cada fila y cada columna.

En nuestro caso resulta que $k_4 = 32$, $k_i = 0$ para $i \neq 4$ y $r_j = 0$ para todo j, de donde $||c'_{ij}(v_1)||$ es la matriz

	col 1	*col* 2	*col* 3	*col* 4	*col* 5	*col* 6	*col* 7
fila 1	∞	0	83	9	30	6	50
fila 2	0	∞	66	37	17	12	26
fila 3	29	1	∞	19	0	12	5
fila 4	0	51	34	∞	17	∞	48
fila 5	3	21	56	7	∞	0	28
fila 6	0	85	8	∞	89	∞	0
fila 7	18	0	0	0	58	13	∞

y $c(v_1) = c(s) + 32 = 96 + 32 = 128$.

Si $\mathcal{C}$ es un circuito que no contiene a la rama (i_1, j_1) (es decir, una solución factible de v_1) entonces para todo $(i,j) \in \mathcal{C}$ vale que $(i,j) \neq (i_1, j_1)$ de donde $c'_{ij}(s) = c_{ij}(v_1)$ para todo $(i,j) \in \mathcal{C}$ (ya que $||c_{ij}(v_1)||$ se obtuvo reemplazando en $||c'_{ij}(s)||$ el coeficiente $c'_{i_1 j_1}(s)$ por ∞).
Sean

$$c(\mathcal{C}, v_1) = \sum_{(i,j)\in\mathcal{C}} c_{ij}(v_1)$$

y

$$c'(\mathcal{C}, v_1) = \sum_{(i,j)\in\mathcal{C}} c'_{ij}(v_1)$$

Ahora se tiene que $c'(\mathcal{C}, v_1) = c(\mathcal{C}, v_1) - 32$ y como $c'_{ij}(s) = c_{ij}(v_1)$ para todo $(i,j) \in \mathcal{C}$ entonces

$$\begin{aligned} c(\mathcal{C}) &= c'(\mathcal{C}, s) + c(s) = \sum_{(i,j)\in\mathcal{C}} c'_{ij}(s) + c(s) = \\ &= \sum_{(i,j)\in\mathcal{C}} c_{ij}(v_1) + c(s) = c(\mathcal{C}, v_1) + c(s) = c'(\mathcal{C}, v_1) + 32 + c(s) = \\ &= c'(\mathcal{C}, v_1) + 32 + 96 = c'(\mathcal{C}, v_1) + 128 = c'(\mathcal{C}, v_1) + c(v_1) \end{aligned}$$

Luego, $c(\mathcal{C}) = c'(\mathcal{C}, v_1) + c(v_1)$ y, como $A_{v_1} = \emptyset$, entonces $c'(\mathcal{C}, v_1) = \sum_{\substack{(i,j)\in\mathcal{C} \\ (i,j)\notin A_{v_1}}} c'_{ij}(v_1)$.

Ahora generamos los hijos de u_1 a partir de la matriz $||c'_{ij}(u_1)||$. Elegimos una rama (i_2, j_2) de G tal que el coeficiente correspondiente a la fila i_2 y a la columna j_2 de $||c'_{ij}(u_1)||$ sea nulo, por ejemplo, $(i_2, j_2) = (3, 5)$, y generamos dos hijos u_2 y v_2 de u_1, el primero será el subproblema que resulta de agregar al problema u_1 la condición de que (i_2, j_2) pertenezca al circuito y el segundo el que resulta de agregar a u_1 la condición de que (i_2, j_2) no pertenezca al circuito.
Finalmente, generamos los hijos de v_1 a partir de la matriz $||c'_{ij}(v_1)||$. Elegimos una rama (i_3, j_3) de G tal que el coeficiente correspondiente a la fila i_3 y a la columna j_3 de $||c'_{ij}(v_1)||$ sea nulo, por ejemplo, $(i_3, j_3) = (2, 1)$ y generamos dos hijos u_3 y v_3 de v_1, el primero será el subproblema que resulta de agregar al problema v_1 la condición de que (i_3, j_3) pertenezca al circuito y el segundo el que resulta de agregar a v_1 la condición de que (i_3, j_3) no pertenezca al circuito.

Paso 4. Calculamos el costo y generamos los hijos de cada nodo u que sea una hoja del subárbol que tenemos construído hasta ahora y que satisfaga que $\#A_u < n - 1$.
El subárbol que tenemos construído hasta ahora consiste de la raíz s, sus dos hijos u_1 y v_1, los hijos u_2 y v_2 de u_1 y los hijos u_3 y v_3 de v_1. Las hojas son, por lo tanto, u_2, v_2, u_3 y v_3.
Comencemos por los hijos de u_1, que son u_2 y v_2. Como $\#A_{u_2} = 2 < 6 = n-1$ pues $A_{u_2} = \{(i_1, j_1), (i_2, j_2)\}$, para calcular $c(u_2)$ y luego generar sus hijos le asociamos a u_2 la matriz de costos $||c_{ij}(u_2)||$ que se obtiene reemplazando en $||c'_{ij}(u_1)||$ el coeficiente $c'_{j_2 i_2}(u_1)$ por ∞ y eliminando la fila i_2 y la columna j_2 y como $\#A_{v_2} = 1 < 6 = n - 1$ pues $A_{v_2} = \{(i_1, j_1)\}$, para calcular $c(v_2)$ y luego generar sus hijos le asociamos a v_2 la matriz de costos $||c_{ij}(v_2)||$ que se obtiene reemplazando en $||c'_{ij}(u_1)||$ el coeficiente $c'_{i_2 j_2}(u_1)$ por ∞.

En nuestro caso $||c_{ij}(u_2)||$ es la matriz

	col 1	*col* 2	*col* 3	*col* 4	*col* 7
fila 1	∞	0	83	9	50
fila 2	0	∞	66	37	26
fila 5	0	18	∞	4	25
fila 6	0	85	8	∞	0
fila 7	18	0	0	0	∞

y $||c_{ij}(v_2)||$ es la matriz

	col 1	*col* 2	*col* 3	*col* 4	*col* 5	*col* 7
$fila$ 1	∞	0	83	9	30	50
$fila$ 2	0	∞	66	37	17	26
$fila$ 3	29	1	∞	19	∞	5
$fila$ 5	0	18	53	4	∞	25
$fila$ 6	0	85	8	∞	89	0
$fila$ 7	18	0	0	0	58	∞

Restando primero a cada coeficiente de la fila i de $||c_{ij}(u_2)||$ el máximo número no negativo k_i y luego a cada coeficiente de la columna j el máximo número no negativo r_j , de manera que cada coeficiente de la nueva matriz sea no negativo, obtenemos la matriz $||c'_{ij}(u_2)||$ que tiene un coeficiente nulo en cada fila y cada columna y como antes definimos $c(u_2) = c(u_1) + \sum k_i + \sum r_j$. Luego $c(u_2) \geq c(u_1)$.
En nuestro ejemplo $k_i = 0 = r_j$ para todo i, j de donde $||c'_{ij}(u_2)|| = ||c_{ij}(u_2)||$ y por lo tanto se tiene que $c(u_2) = c(u_1) + 0 = c(u_1) = 99$.

Dejamos a cargo del lector verificar que si $\mathcal{C}$ es un circuito que contiene a las ramas (i_1, j_1) e (i_2, j_2) (es decir, una solución factible de u_2), definiendo

$$c'(\mathcal{C}, u_2) = \sum_{\substack{(i,j)\in\mathcal{C} \\ (i,j)\neq(i_1,j_1) \\ (i,j)\neq(i_2,j_2)}} c'_{ij}(u_2)$$

resulta que $c(\mathcal{C}) = c'(\mathcal{C}, u_2) + c(u_2)$ y $c'(\mathcal{C}, u_2) = \sum_{\substack{(i,j)\in\mathcal{C} \\ (i,j)\notin A_{u_2}}} c'_{ij}(u_2)$.

De manera análoga, restando primero a cada coeficiente de la fila i de $||c_{ij}(v_2)||$ el máximo número no negativo k_i y luego a cada coeficiente de la columna j el máximo número no negativo r_j , de manera que cada coeficiente de la nueva matriz sea no negativo, obtenemos la matriz $||c'_{ij}(v_2)||$ que tiene un coeficiente nulo en cada fila y cada columna y definimos $c(v_2) = c(u_1) + \sum k_i + \sum r_j$. Luego $c(v_2) \geq c(u_1)$.
En nuestro caso, $k_3 = 1$, $k_i = 0$ para $i \neq 3$, $r_5 = 17$ y $r_j = 0$ para $j \neq 5$ de donde $||c'_{ij}(v_2)||$ es la matriz

	col 1	*col* 2	*col* 3	*col* 4	*col* 5	*col* 7
$fila$ 1	∞	0	83	9	13	50
$fila$ 2	0	∞	66	37	0	26
$fila$ 3	28	0	∞	18	∞	4
$fila$ 5	0	18	53	4	∞	25
$fila$ 6	0	85	8	∞	72	0
$fila$ 7	18	0	0	0	41	∞

y $c(v_2) = c(u_1) + 18 = 99 + 18 = 117$.

Nuevamente dejamos a cargo del lector verificar que si $\mathcal{C}$ es un circuito que contiene a la rama (i_1, j_1) y no contiene a (i_2, j_2) (es decir, una solución factible de v_2), definiendo

$$c'(\mathcal{C}, v_2) = \sum_{\substack{(i,j)\in\mathcal{C} \\ (i,j)\neq(i_1,j_1)}} c'_{ij}(v_2)$$

resulta que $c(\mathcal{C}) = c'(\mathcal{C}, v_2) + c(v_2)$ y $c'(\mathcal{C}, v_2) = \sum_{\substack{(i,j)\in\mathcal{C} \\ (i,j)\notin A_{v_2}}} c'_{ij}(v_2)$.

Ahora consideremos los hijos de v_1, que son u_3 y v_3. Como $\#A_{u_3} = 1 < 6 = n - 1$ pues $A_{u_3} = \{(i_3, j_3)\}$, para calcular $c(u_3)$ y luego generar sus hijos le asociamos a u_3 la matriz de costos $||c_{ij}(u_3)||$ que se obtiene reemplazando en $||c'_{ij}(v_1)||$ el coeficiente $c'_{j_3 i_3}(v_1)$ por ∞ y eliminando la fila i_3 y la columna j_3 y como $\#A_{v_3} = 0 < 6 = n-1$ pues $A_{v_3} = \emptyset$, para calcular $c(v_3)$ y luego generar sus hijos le asociamos a v_3 la matriz de costos $||c_{ij}(v_3)||$ que se obtiene reemplazando en $||c'_{ij}(v_1)||$ el coeficiente $c'_{i_3 j_3}(v_1)$ por ∞.

En nuestro ejemplo, $||c_{ij}(u_3)||$ es la matriz

	col 2	*col* 3	*col* 4	*col* 5	*col* 6	*col* 7
fila 1	∞	83	9	30	6	50
fila 3	1	∞	19	0	12	5
fila 4	51	34	∞	17	∞	48
fila 5	21	56	7	∞	0	28
fila 6	85	8	∞	89	∞	0
fila 7	0	0	0	58	13	∞

y $||c_{ij}(v_3)||$ es la matriz

	col 1	*col* 2	*col* 3	*col* 4	*col* 5	*col* 6	*col* 7
fila 1	∞	0	83	9	30	6	50
fila 2	∞	∞	66	37	17	12	26
fila 3	29	1	∞	19	0	12	5
fila 4	0	51	34	∞	17	∞	48
fila 5	3	21	56	7	∞	0	28
fila 6	0	85	8	∞	89	∞	0
fila 7	18	0	0	0	58	13	∞

Restando primero a cada coeficiente de la fila i de $||c_{ij}(u_3)||$ el máximo número no negativo k_i y luego a cada coeficiente de la columna j el máximo número no negativo r_j, de manera que cada coeficiente de la nueva matriz sea no negativo, obtenemos la matriz $||c'_{ij}(u_3)||$ que tiene un coeficiente nulo en cada fila y cada columna y definimos $c(u_3) = c(v_1) + \sum k_i + \sum r_j$. Luego $c(u_3) \geq c(v_1)$.
En nuestro caso, $k_1 = 6$, $k_4 = 17$, $k_i = 0$ para $i \neq 1, 4$, y $r_j = 0$ para todo j de donde $||c'_{ij}(u_3)||$ es la matriz

	col 2	*col* 3	*col* 4	*col* 5	*col* 6	*col* 7
fila 1	∞	77	3	24	0	44
fila 3	1	∞	19	0	12	5
fila 4	34	17	∞	0	∞	31
fila 5	21	56	7	∞	0	28
fila 6	85	8	∞	89	∞	0
fila 7	0	0	0	58	13	∞

y $c(u_3) = c(v_1) + 23 = 128 + 23 = 151$

Si $\mathcal{C}$ es un circuito que no contiene a la rama (i_1, j_1) y sí contiene a (i_3, j_3) (es decir, una solución factible de u_3), definiendo

$$c'(\mathcal{C}, u_3) = \sum_{\substack{(i,j)\in\mathcal{C} \\ (i,j)\neq(i_3,j_3)}} c'_{ij}(u_3)$$

se tiene que $c(\mathcal{C}) = c'(\mathcal{C}, u_3) + c(u_3)$ y $c'(\mathcal{C}, u_3) = \displaystyle\sum_{\substack{(i,j)\in\mathcal{C} \\ (i,j)\notin A_{u_3}}} c'_{ij}(u_3)$

Análogamente, restando primero a cada coeficiente de la fila i de $||c_{ij}(v_3)||$ el máximo número no negativo k_i y luego a cada coeficiente de la columna j el máximo número no negativo r_j , de manera que cada coeficiente de la nueva matriz sea no negativo, obtenemos la matriz $||c'_{ij}(v_3)||$ que tiene un coeficiente nulo en cada fila y cada columna y definimos $c(v_3) = c(v_1) + \sum k_i + \sum r_j$. Luego $c(v_3) \geq c(v_1)$.
En nuestro caso, $k_2 = 12$, $k_i = 0$ para $i \neq 2$, y $r_j = 0$ para todo j de donde $||c'_{ij}(v_3)||$ es la matriz

	col 1	*col* 2	*col* 3	*col* 4	*col* 5	*col* 6	*col* 7
fila 1	∞	0	83	9	30	6	50
fila 2	∞	∞	54	25	5	0	14
fila 3	29	1	∞	19	0	12	5
fila 4	0	51	34	∞	17	∞	48
fila 5	3	21	56	7	∞	0	28
fila 6	0	85	8	∞	89	∞	0
fila 7	18	0	0	0	58	13	∞

y $c(v_3) = c(v_1) + 12 = 128 + 12 = 140$.

Además, si $\mathcal{C}$ es un circuito que no contiene a las ramas (i_1, j_1) e (i_3, j_3) definiendo

$$c'(\mathcal{C}, v_3) = \sum_{(i,j)\in\mathcal{C}} c'_{ij}(v_3)$$

resulta que $c(\mathcal{C}) = c'(\mathcal{C}, v_3) + c(v_3)$ y $c'(\mathcal{C}, v_3) = \sum_{\substack{(i,j)\in\mathcal{C} \\ (i,j)\notin A_{v_3}}} c'_{ij}(v_3)$.

Ahora deberíamos generar los hijos de u_2, v_2, u_3 y v_3 e ir al paso 5, cosa que no haremos porque calculamos que a estas alturas el lector ya ha comprendido el procedimiento y su paciencia está a punto de agotarse.

Resumiendo: hasta ahora, en cada paso calculamos el costo y generamos los hijos de cada vértice u que sea una hoja del subárbol que se tiene hasta ese momento y que satisfaga que $\#A_u < n - 1$, asociando a u una matriz $||c'_{ij}(u)||$ que tiene un cero en cada fila y en cada columna y satisface:
Si $A_u = \{e \in E /\ \text{“}e \in \mathcal{C}\text{”}$ es una restricción de $u\}$, para cualquier solución factible $\mathcal{C}$ de u se tiene que

$$c(\mathcal{C}) = c'(\mathcal{C}, u) + c(u)$$

donde

$$c'(\mathcal{C}, u) = \sum_{\substack{(i,j)\in\mathcal{C} \\ (i,j)\notin A_u}} c'_{ij}(u)$$

Además, el costo $c(u)$ de u que calculamos satisface que $c(u) \le c(v)$ para cada hijo v de u.
Finalmente, veamos ahora cómo procedemos cuando tenemos una hoja u del subárbol generado hasta el momento que satisface $\#A_u = n-1$. En este caso, no generamos ningún hijo de u y para calcular $c(u)$ primero determinamos si u es o no es factible. Si es factible entonces calculamos $c(u)$ con el mismo procedimiento de antes. En cambio, si no es factible, ponemos $c(u) = \infty$.

De esta manera obtenemos un árbol binario con raíz donde cada nodo u tiene asignado un costo $c(u)$ en forma tal que vale $c(u) \le c(v)$ si (u, v) es una rama del árbol. Las hojas de este árbol son los vértices h tales que $\#A_h = n - 1$. Además, para cada hoja h del árbol que sea un problema factible tenemos definida una matriz $||c'_{ij}(h)||$ que tiene un cero en cada fila y cada columna y satisface:
Si $A_h = \{e \in E /\ \text{“}e \in \mathcal{C}\text{”}$ es una restricción de $h\}$, para cualquier solución factible $\mathcal{C}$ de h se tiene que

$$c(\mathcal{C}) = c'(\mathcal{C}, h) + c(h)$$

donde

$$c'(\mathcal{C}, h) = \sum_{\substack{(i,j)\in\mathcal{C} \\ (i,j)\notin A_h}} c'_{ij}(h)$$

Notemos que determinar si una hoja h del árbol es factible es fácil porque si $\#A_h = n - 1$ entonces hay $n - 1$ ramas que necesariamente deben pertenecer a cualquier solución factible de h. Por la foma en que fuimos procediendo (cada vez que agregamos una restricción del tipo "$(i,j) \in \mathcal{C}$" eliminamos la fila i y la columna j), no puede haber en A_h dos ramas con la misma punta ni dos ramas con la misma cola. Luego, los conjuntos $\{i \,/\, (i,j) \in A_h\}$ y $\{j \,/\, (i,j) \in A_h\}$ tienen $n-1$ elementos cada uno. Por lo tanto, existe un único i_0 y un único j_0 tales que $i_0 \notin \{i \,/\, (i,j) \in A_h\}$ y $j_0 \notin \{j \,/\, (i,j) \in A_h\}$. Notemos que si $\mathcal{C}$ es una solución factible de h entonces tiene n ramas, verifica que para cada i entre 1 y n hay una y sólo una rama de $\mathcal{C}$ cuya cola sea i y una y sólo una rama de $\mathcal{C}$ cuya punta sea i y las $n-1$ ramas de A_h deben ser ramas de $\mathcal{C}$. Esto muestra que el único camino $\mathcal{P}$ que podría ser una solución factible de h es aquél cuyas ramas son las $n-1$ ramas pertenecientes a A_h y la rama (i_0, j_0) (ya que si agregamos una rama (i,j) con $i \neq i_0$ o $j \neq j_0$ a las ramas pertenecientes a A_h ese camino no tendría ninguna rama cuya punta es i_0 o ninguna rama cuya cola es j_0). Por otra parte, como $\mathcal{P}$ no contiene dos ramas con la misma punta ni dos ramas con la misma cola, entonces es un circuito (en cuyo caso es la única solución factible de h) o bien consiste de dos o más subcircuitos (en cuyo caso h no es factible), cosa que puede chequearse fácilmente.

Veamos ahora que la solución al problema del viajante es la hoja de mínimo costo.

Si h es una hoja factible entonces $\#A_h = n-1$. Como cada vez que agregamos una restricción del tipo "$(i,j) \in \mathcal{C}$" eliminamos una fila y una columna y para obtener h hemos agregado $n-1$ de estas restricciones, entonces $||c'_{ij}(h)||$ es una matriz de 1×1 que tiene un cero en cada fila y cada columna. Luego debe ser $||c'_{ij}(h)|| = (\,0\,)$ de donde $c'(\mathcal{C}, h) = 0$. Por lo tanto, si $\mathcal{C}$ es una solución factible de h resulta que $c(\mathcal{C}) = c'(\mathcal{C}, h) + c(h) = c(h)$.

Como las soluciones factibles del problema del viajante son las soluciones factibles de cada una de las hojas del árbol y como el costo de dicha solución factible coincide con el costo de su correspondiente hoja entonces nuestro problema se traduce en encontrar una hoja de mínimo costo. Además, como vimos antes, si h es la hoja de mínimo costo entonces tiene una única solución factible que puede hallarse fácilmente y que, por lo dicho, resulta ser la solución al problema del viajante.

Para hallar una hoja de mínimo costo, utilizamos el método de branch and bound y, como en el caso de programación lineal entera, en lugar de generar el árbol y luego aplicar el algoritmo, en cada paso calculamos sólo los vértices y los costos que necesitamos.

Ejemplo 5.3. Resolvamos el problema del viajante para el grafo completo de $n = 4$ vértices que tiene como matriz de costos a la matriz

$$||c_{ij}|| = \begin{pmatrix} \infty & 4 & 99 & 23 \\ 8 & \infty & 86 & 55 \\ 6 & 24 & \infty & 20 \\ 12 & 87 & 22 & \infty \end{pmatrix}$$

Iniciamos el algoritmo poniendo $c = \infty$.

Primero calculamos $||c'_{ij}(s)||$ y $c(s)$ y obtenemos

$$||c'_{ij}(s)|| = \begin{pmatrix} \infty & 0 & 85 & 5 \\ 0 & \infty & 68 & 33 \\ 0 & 18 & \infty & 0 \\ 0 & 75 & 0 & \infty \end{pmatrix}$$

y $c(s) = 54 < \infty = c$.

Ahora generamos uno de los hijos de s, que denotaremos por p_1, agregando la restricción $(4,1) \in \mathcal{C}$. Se tiene entonces que $||c_{ij}(p_1)||$ es la matriz

	col 2	*col* 3	*col* 4
fila 1	0	85	∞
fila 2	∞	68	33
fila 3	18	∞	0

de donde $||c'_{ij}(p_1)||$ es

	col 2	*col* 3	*col* 4
$fila$ 1	0	50	∞
$fila$ 2	∞	0	0
$fila$ 3	18	∞	0

y $c(p_1) = 122 < \infty = c$.

Ahora generamos uno de los hijos de p_1, al que denotaremos por p_2, obtenido agregándole a p_1 la restricción $(2,3) \in \mathcal{C}$. Se tiene entonces que $||c_{ij}(p_2)||$ es la matriz

	col 2	*col* 4
$fila$ 1	0	∞
$fila$ 3	∞	0

de donde $||c'_{ij}(p_2)|| = ||c_{ij}(p_1)||$ y $c(p_2) = 122 < \infty = c$.

Ahora generamos uno de los hijos de p_2, al que denotaremos por p_3, obtenido agregándole a p_2 la restricción $(1,2) \in \mathcal{C}$. Como $\#A_{p_3} = 3 = n-1$ entonces para calcular su costo primero debemos ver si es factible. Como $A_{p_3} = \{(4,1),(2,3),(1,2)\}$ entonces $\{i \,/\, (i,j) \in A_{p_3}\} = \{4,2,1\}$ y $\{j \,/\, (i,j) \in A_{p_3}\} = \{1,3,2\}$. Por lo tanto, el único i_0 entre 1 y 4 tal que $i_0 \notin \{i \,/\, (i,j) \in A_{p_3}\}$ es $i_0 = 3$ y el único j_0 entre 1 y 4 tal que $j_0 \notin \{j \,/\, (i,j) \in A_{p_3}\}$ es $j_0 = 4$. Luego, el único camino que podría ser una solución factible de p_3 es aquél cuyas ramas son $(4,1)$, $(2,3)$, $(1,2)$ y $(3,4)$. Como este camino es un circuito entonces p_3 es factible y su única solución es el circuito $4 \longrightarrow 1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 4$.
Calculemos $c(p_3)$. Como $||c_{ij}(p_3)||$ es la matriz

	col 4
$fila$ 3	0

entonces $||c'_{ij}(p_3)|| = ||c_{ij}(p_3)||$ y $c(p_3) = 122 < \infty = c$. Como además p_3 es una hoja entonces actualizamos h y c poniendo $h = p_3$ y $c = 122$.

Ahora hacemos un backtracking y volvemos a p_2 para generar su otro hijo, que denotaremos por p_4, obtenido agregando a p_2 la restricción $(1,2) \notin \mathcal{C}$. Como $||c_{ij}(p_4)||$ es la matriz

	col 2	*col* 4
$fila$ 1	∞	∞
$fila$ 3	∞	0

entonces $||c'_{ij}(p_4)||$ es la matriz

	col 2	*col* 4
$fila$ 1	0	0
$fila$ 3	∞	0

y $c(p_4) = \infty \geq 122 = c$, de modo que hacemos otro backtracking y volvemos a a p_1 para generar su otro hijo, que denotaremos por p_5, obtenido agregando a p_1 la restricción $(2,3) \notin \mathcal{C}$. Se tiene que $||c_{ij}(p_5)||$ es la matriz

	col 2	*col* 3	*col* 4
$fila$ 1	0	50	∞
$fila$ 2	∞	∞	0
$fila$ 3	18	∞	0

de donde $||c'_{ij}(p_5)||$ es la matriz

	col 2	*col* 3	*col* 4
fila 1	0	0	∞
fila 2	∞	∞	0
fila 3	18	∞	0

y $c(p_5) = 172$. Como $c(p_5) = 172 \geq 122 = c$ entonces podamos toda la descendencia de p_5 y hacemos otro backtracking para volver a s y generar su otro hijo, que denotaremos por p_6, obtenido agregando a s la restricción $(4,1) \notin \mathcal{C}$. Se tiene que $||c_{ij}(p_6)||$ es la matriz

	col 1	*col* 2	*col* 3	*col* 4
fila 1	∞	0	85	5
fila 2	0	∞	68	33
fila 3	0	18	∞	0
fila 4	∞	75	0	∞

de donde $||c'_{ij}(p_6)|| = ||c_{ij}(p_6)||$ y $c(p_6) = 54 < 122 = c$.

Ahora generamos uno de los hijos de p_6, al que denotaremos por p_7, obtenido agregándole a p_6 la restricción $(2,1) \in \mathcal{C}$. Se tiene entonces que $||c_{ij}(p_7)||$ es la matriz

	col 2	*col* 3	*col* 4
fila 1	∞	85	5
fila 3	18	∞	0
fila 4	75	0	∞

Luego, $||c'_{ij}(p_7)||$ es

	col 2	*col* 3	*col* 4
fila 1	∞	80	0
fila 3	0	∞	0
fila 4	57	0	∞

de donde $c(p_7) = 77 < 122 = c$.

Ahora generamos uno de los hijos de p_7, al que denotaremos por p_8, obtenido agregándole a p_7 la restricción $(3,2) \in \mathcal{C}$. Se tiene entonces que $||c_{ij}(p_8)||$ es la matriz

	col 3	*col* 4
fila 1	80	0
fila 4	0	∞

de donde $||c'_{ij}(p_8)|| = ||c_{ij}(p_8)||$ y $c(p_8) = 77 < 122 = c$.

Ahora generamos uno de los hijos de p_8, al que denotaremos por p_9, obtenido agregándole a p_8 la restricción $(4,3) \in \mathcal{C}$.
Como $\#A_{p_9} = 3 = n - 1$, para calcular su costo debemos ver si es factible. Dejamos a cargo del lector verificar que p_9 es factible y que su única solución factible es el circuito $4 \longrightarrow 3 \longrightarrow 2 \longrightarrow 1 \longrightarrow 4$.
Calculemos su costo. Como $||c_{ij}(p_9)||$ es la matriz

	col 4
fila 1	0

entonces $||c'_{ij}(p_9)|| = ||c_{ij}(p_9)||$ y $c(p_9) = 77 < 122 = c$. Como además p_9 es una hoja actualizamos h y c poniendo $h = p_9$ y $c = 77$.

Ahora hacemos un backtracking y volvemos a p_8 para generar su otro hijo, que denotaremos por p_{10}, obtenido agregando a p_8 la restricción $(4,3) \notin \mathcal{C}$. Como $||c_{ij}(p_{10})||$ es la matriz

	col 3	*col* 4
$fila$ 1	80	0
$fila$ 4	∞	∞

entonces $\|\|c'_{ij}(p_{10})\|\|$ es la matriz

	col 3	*col* 4
$fila$ 1	80	0
$fila$ 4	0	0

y $c(p_{10}) = \infty \geq 77 = c$, de modo que podamos toda su descendencia y hacemos otro backtracking volviendo a p_7 para generar su otro hijo, que denotaremos por p_{11}, obtenido agregando a p_7 la restricción $(3,2) \notin \mathcal{C}$. Se tiene que $\|\|c_{ij}(p_{11})\|\|$ es la matriz

	col 2	*col* 3	*col* 4
$fila$ 1	∞	80	0
$fila$ 3	∞	∞	0
$fila$ 4	57	0	∞

Luego, $\|\|c'_{ij}(p_{11})\|\|$ es la matriz

	col 2	*col* 3	*col* 4
$fila$ 1	∞	80	0
$fila$ 3	∞	∞	0
$fila$ 4	0	0	∞

y $c(p_{11}) = 134$. Como $c(p_{11}) = 134 \geq 77 = c$ entonces podamos toda la descendencia de p_{11} y hacemos otro backtracking para volver a p_6 y generar su otro hijo, que denotaremos por p_{12}, obtenido agregando a p_6 la restricción $(2,1) \notin \mathcal{C}$. Se tiene que $\|\|c_{ij}(p_{12})\|\|$ es la matriz

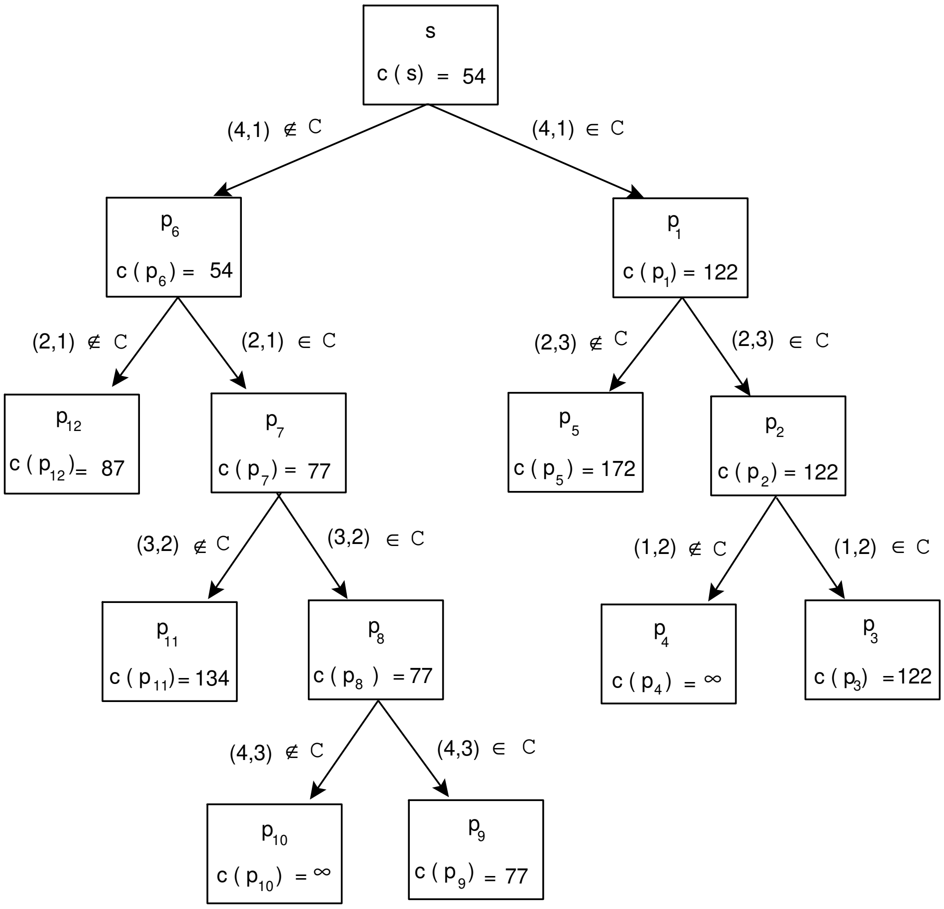
	col 1	*col* 2	*col* 3	*col* 4
$fila$ 1	∞	0	85	5
$fila$ 2	∞	∞	68	33
$fila$ 3	0	18	∞	0
$fila$ 4	∞	75	0	∞

de donde $\|\|c'_{ij}(p_{12})\|\|$ es la matriz

	col 1	*col* 2	*col* 3	*col* 4
$fila$ 1	∞	0	85	5
$fila$ 2	∞	∞	35	0
$fila$ 3	0	18	∞	0
$fila$ 4	∞	75	0	∞

y $c(p_{12}) = 87$. Como $c(p_{12}) = 87 \geq 77 = c$ entonces podamos toda la descendencia de p_{12} y como ya hemos examinado todos los hijos de todos los vértices a los que podemos llegar con backtracking, el algoritmo se detiene.

Luego, la hoja de mínimo costo es $h = p_9$ con costo $c = 77$, lo que significa que el circuito $\mathcal{C}$ de mínimo costo es $4 \longrightarrow 3 \longrightarrow 2 \longrightarrow 1 \longrightarrow 4$ con costo $c(\mathcal{C}) = 77$. El subárbol generado por el algoritmo es


s
c (s) = 54
(4,1) ∉ C
(4,1) ∈ C
p6
c (p6) = 54
p1
c (p1) = 122
(2,1) ∉ C
(2,1) ∈ C
(2,3) ∉ C
(2,3) ∈ C
p12
c (p12)= 87
p7
c (p7) = 77
p5
c (p5) = 172
p2
c (p2) = 122
(3,2) ∉ C
(3,2) ∈ C
(1,2) ∉ C
(1,2) ∈ C
p11
c (p11)= 134
p8
c (p8) = 77
p4
c (p4) = ∞
p3
c (p3) = 122
(4,3) ∉ C
(4,3) ∈ C
p10
c (p10) = ∞
p9
c (p9) = 77