CAPÍTULO 4
PROGRAMACIÓN LINEAL ENTERA

Programación Lineal Entera

Es una técnica que permite modelar y resolver problemas cuya característica principal es que el conjunto de soluciones factibles es discreto.

Modelo Lineal (que vimos en P.L.) agregando que las variables de decisión deben ser enteras.

A veces, algunas variables pueden ser enteras y otras continuas, en este caso hablamos de **programación lineal entera mixta**.

Ejemplos de estos modelos: localización de plantas.

Caso Particular: modelos lineales binarios donde las variables representan decisiones del tipo si o no (1 o 0).

El Problema

$$\begin{aligned} (\text{PE})\ \text{mín} \quad & z = c^T x \\ \text{s.a.} \quad & Ax \leq b \\ & x_i \in IN_0 \quad i = 1, \ldots, n \end{aligned}$$

Con IN_0 los naturales unión el cero.
Es claro que si el conjunto $\{x/Ax \leq b\}$ es acotado, el número de soluciones factible es finito.

Eso implica que el problema es fácil?
No, necesariamente, porque el número de soluciones puede ser exponencial en el tamaño del problema (número de variables).
Por ejemplo, el conjunto definido por

$$\begin{aligned} 0 \leq x_i \leq 1 \quad & i = 1, \ldots, n \\ x_i \in IN_0 \quad & i = 1, \ldots, n \end{aligned}$$

Es el conjunto de vértices del n-cubo que tiene 2^n vértices.

Se puede mostrar que los problemas del tipo (PE) son NP-completos, lo que significa que son difíciles de resolver.

El Problema

Problema Mixto:

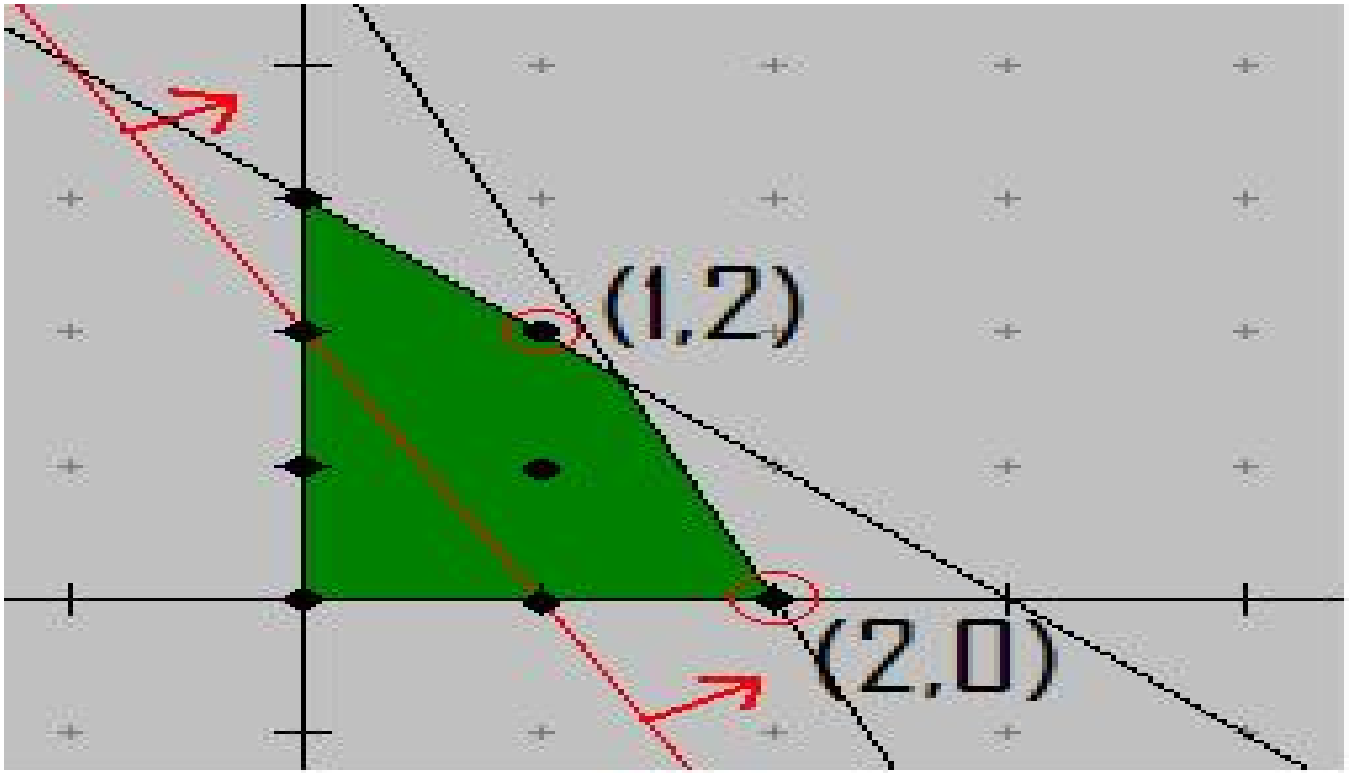
$$\begin{array}{rll} (\text{PM})\,\text{mín} & z = c^T x & \\ \text{s.a.} & Ax \leq b & \\ & x_i \in IN_0 & i \in I \subseteq \{1, \ldots, n\} \\ & x_j \geq 0 & j \in I^C \end{array}$$

(PM) es tan complicado como (PE).

Ejemplo de (PE):

$$\begin{array}{rll} \text{máx}\, z = & 2x_1 + x_2 & \\ \text{s.a.} & x_1 + x_2 \leq 3 & \\ & 5x_1 + 2x_2 \leq 10 & \\ & x_1, x_2 \geq 0, & enteros \end{array}$$

El Problema



Dejamos los métodos de soluciones para más adelante. Veremos algunos modelos de programación lineal entera.

Problema de la Mochila

Se tienen n tipos diferentes de objetos, cada uno de ellos tiene un peso w_j y un valor v_j.

Se dispone de una mochila donde se colocarán los objetos, que soporta un peso máximo de W, de manera de maximizar el valor total del contenido de la mochila.

Los objetos son indivisibles, o sea sólo se puede colocar un número entero de cada objeto.

Modelo: (Problema de la Mochila (knapsack) Entero)

$x_j \longrightarrow$ unidades del objeto j que se colocarán en la mochila.

$$\begin{array}{ll} \text{máx } z = & \sum_{j=1}^{n} v_j x_j \\ \text{s.a.} & \sum_{j=1}^{n} w_j x_j \leq W \\ & x_j \geq 0, \quad enteros \quad j = 1, \ldots, n \end{array}$$

Problema de la Mochila

Si solo existe un objeto de cada tipo entonces $x_j = 0$ ó $1 \longrightarrow$ knapsack binario ó 0-1, 1 corresponde a poner el objeto en la mochila y 0 a no ponerlo.

Otra aplicación de este problema se utiliza cuando existen mercaderías que deben ser almacenadas o transportadas considerando una disponibilidad de espacio o peso limitado, con cada mercadería con un cierto valor.

Está probado que se puede resolver (usando programación dinámica) en tiempo $O(nW)$. Pero si W no está acotado por un polinomio en n esto no es un buen resultado.

Problema de Asignación

Hay 5 profesores y 5 cursos. Cada profesor define su grado de preferencia a dictar un curso determinado con un número de 1 a 10.

Cursos \ Profesores	A	B	C	D	E
Optimización	8	9	6	10	3
Economía	6	6	10	4	9
Inv. Operativa	10	8	5	10	4
Gestión de Operaciones	10	9	4	8	5
Logística	5	4	9	3	9

Definimos el nivel total de satisfacción como la suma de las satisfacciones personales. Por ejemplo, veamos el beneficio de la siguiente asignación:

$$\begin{array}{lcll} \text{Economía} & \longleftrightarrow & B & 6 \\ \text{Optimización} & \longleftrightarrow & A & 8 \\ \text{Inv. Operativa} & \longleftrightarrow & C & 5 \\ \text{G. Operaciones} & \longleftrightarrow & D & 8 \\ \text{Logística} & \longleftrightarrow & E & 9 \\ & & total = & 36 \end{array}$$

Problema de Asignación

Se puede ver que no es la mejor.

Modelo:

$$x_{ij} = \begin{cases} 1 & \text{Sí el profesor } i \text{ se asigna al curso } j \\ 0 & \sim \end{cases}$$

Con $i, j = 1, \ldots, n$.

Veamos las restricciones:

- Cada profesor debe ser asignado a exactamente 1 curso

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

Problema de Asignación

- A cada curso debe asignarse exactamente un profesor

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

- Naturaleza de las variables

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

Finalmente la función objetivo sería:

$$\text{máx}\, z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

Con c_{ij} la preferencia del profesor i por el curso j.

Problema de Asignación

Este problema se conoce como **matching máximo**.

Si en vez de tener 2 conjuntos, ambos de n elementos, supongamos que ahora se tienen m cursos y n profesores con $m < n$.
Entonces podremos formar m pares de elementos.
Las restricciones cambian a:

$$\sum_{j=1}^{m} x_{ij} \leq 1 \quad i = 1, \ldots, n$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad j = 1, \ldots, m$$

O sea, a cada curso se le asigna un profesor pero quedan algunos profesores sin cursos.

Matching Perfecto Máximo:
Ahora podría pasar que tengamos un solo conjunto de objetos, por lo que cada par estará formado por objetos del mismo conjunto.

Problema de Asignación

Supongamos que tenemos n ingenieros en una empresa (con n par) que deben evaluar $\frac{n}{2}$ proyectos, y la empresa ha decidido armar grupos de 2 ingenieros para evaluar cada proyecto. Cada ingeniero ha manifestado en alguna escala su preferencia para trabajar con los otros (sea c_{ij} la suma de la preferencia de i para trabajar con j más la preferencia de j para trabajar con i).

Modelo:

$$x_{ij} = \begin{cases} 1 & \text{Sí } i \text{ forma equipo con } j \\ 0 & \sim \end{cases}$$

Con $i < j$, $i = 1, \ldots, n-1$.

$$\text{máx } z = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} x_{ij}$$

$$\begin{aligned} \text{s.a.} \quad & \textstyle\sum_{j=2}^{n} x_{1j} = 1 \\ & \textstyle\sum_{k=1}^{i-1} x_{ki} + \sum_{j=i+1}^{n} x_{ij} = 1 \quad i = 2, \ldots, n-1 \\ & \textstyle\sum_{k=1}^{n-1} x_{kn} = 1 \\ & x_{ij} \in \{0, 1\} \quad i = 1, \ldots, n-1; j = i+1, \ldots, n \end{aligned}$$

El Vendedor Viajero

Hay n ciudades que visitar, c_{ij} corresponde al costo de ir de la ciudad i a la ciudad j o viceversa. El problema es definir el tour óptimo.

Modelo:

$$x_{ij} = \begin{cases} 1 & \text{si el vendedor va desde } i \text{ a } j \\ 0 & \text{en caso contrario} \end{cases}$$

Veamos las restricciones:

1. Naturaleza de las variables

$$x_{ij} \in \{0, 1\} \quad (1)$$

2. El vendedor debe entrar exactamente una vez a cada ciudad

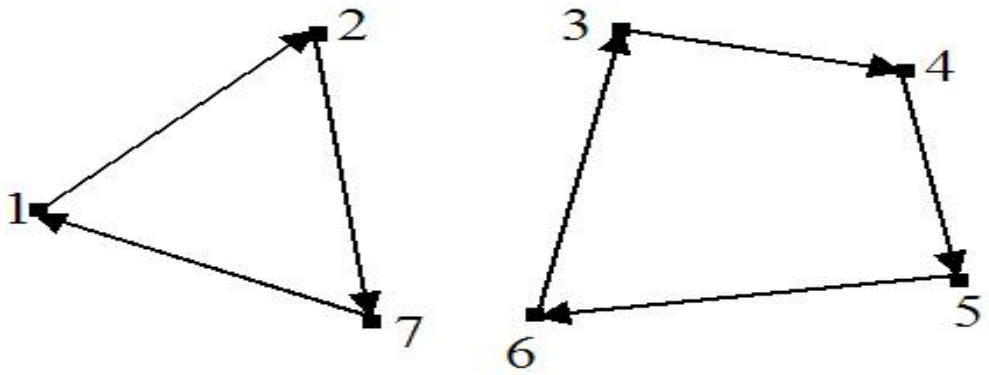
$$\sum_{i=1, i \neq j}^{n} x_{ij} = 1 \quad j = 1, \ldots, n \quad (2)$$

El Vendedor Viajero

3. El vendedor debe salir exactamente una vez de cada ciudad

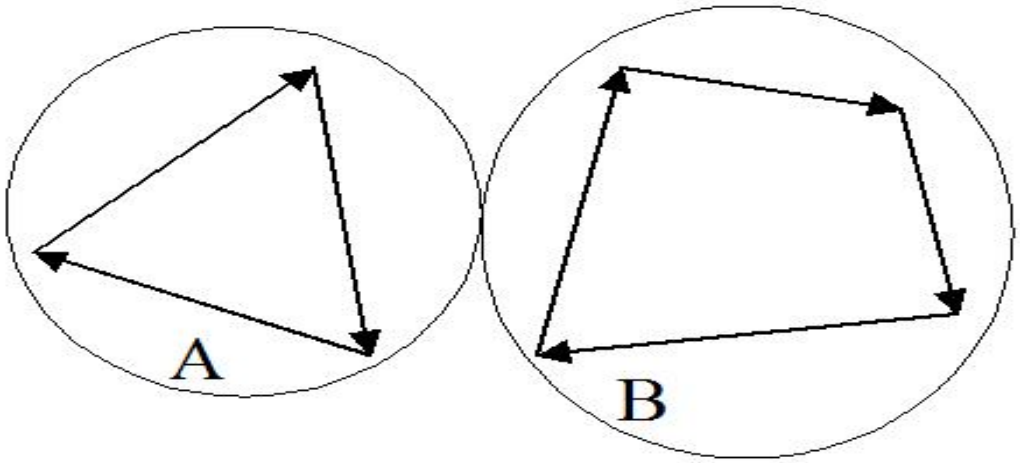
$$\sum_{j=1, j\neq i}^{n} x_{ij} = 1 \quad i = 1, \ldots, n \quad (3)$$

Basta con estas restricciones?
NO, no se evitan los subtours:



El Vendedor Viajero

Cómo evitar los subtours?



Debe existir un viaje de A a B y uno de B a A.

Para cada subconjunto de ciudades $U \subseteq \{1, \ldots, n\}$ tal que $2 \leq |U| \leq n-2$, las restricciones

$$\sum_{(i,j)/i \in U; j \in V \setminus U} x_{ij} \geq 1 \quad \forall U \subseteq V \text{ tal que } 2 \leq |U| \leq n-2 \quad (4)$$

El Vendedor Viajero

Son verificadas por todo circuito, pero cada subcircuito viola al menos una de ellas.

Otra posibilidad de escribir la misma restricción es:

$$\sum_{(i,j)/i\in U;j\in U;i\neq j} x_{ij} \leq |U| - 1 \quad \forall U \subseteq V \text{ tal que } 2 \leq |U| \leq n-2 \quad (4')$$

Luego la formulación adecuada corresponde a las restricciones:

$$(1) + (2) + (3) + (4) \text{ ó } (4')$$

Problema de la Formulación: Número de restricciones (4) o (4') $\longrightarrow O(2^n)$ lo que dificulta su resolución.

La función objetivo sería $\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$.

Este tipo de problemas se aplica en Ruteo de Vehículos, Robótica (circuito VLSI), etc.

Veamos ahora un modelo mixto.

Localización de Plantas

Sea $N = \{1, \ldots, n\}$ el conjunto de las localizaciones posibles para instalar P plantas. $M = \{1, \ldots, n\}$ corresponde al conjunto de clientes que demandan productos.

El problema consiste en determinar cuál es la ubicación de las plantas que permite satisfacer la demanda a un costo mínimo, sabiendo que se conocen los siguientes parámetros:

- c_i: Costo, en \$, de instalar la planta en la localidad i $(i = 1, \ldots, n)$.
- b_j: Demanda, en unidades, del cliente j $(j = 1, \ldots, m)$.
- h_{ij}: Costo unitario de transporte de la localidad i al cliente j.
- u_i: Capacidad de una planta instalada en la localidad i.

Modelo:

$$X_i = \begin{cases} 1 & \text{Sí se instala una planta en la localidad } i \\ 0 & \sim \end{cases}$$

Localización de Plantas

$Y_{ij} =$ Cantidad de producto a ser transportada desde la planta i al cliente j

Veamos las restricciones:

1. Satisfacción de demanda

$$\sum_{i=1}^{n} Y_{ij} = b_j \quad j = 1, \ldots, m$$

2. Capacidad de las plantas

$$\sum_{j=1}^{m} Y_{ij} \leq u_i X_i \quad i = 1, \ldots, n$$

$$(X_i = 0 \Rightarrow Y_{i1}, \ldots, Y_{im} \text{ deben ser } 0)$$

Localización de Plantas

3. Naturaleza de las variables

$$Y_{ij} \geq 0 \quad i = 1, \ldots, n; j = \ldots, m$$

$$X_i \in \{0, 1\} \quad i = 1, \ldots, n$$

Y la función objetivo será:

$$\text{mín}\, z = \underbrace{\sum_{i=1}^{n} c_i X_i}_{\text{Costo de Instalación}} + \underbrace{\sum_{i=1}^{n} \sum_{j=1}^{m} h_{ij} Y_{ij}}_{\text{Costo de Transporte}}$$

Ramificación y Acotamiento (Branch & Bound)

$$\begin{aligned} (\text{PE})\ \text{mín} \quad & z = c^T x \\ \text{s.a.} \quad & Ax \leq b \\ & x_i \in IN_0 \quad i = 1, \ldots, n \end{aligned}$$

Definición:
El problema lineal continuo que se obtiene del problema (PE) al omitir las restricciones de integralidad de las variables se denomina **relajación lineal** de (PE).

Observación: Si la solución óptima de la relajación lineal de (PE) es entera, entonces esta solución es óptima para (PE).

La estrategia de enumerar todas las soluciones factibles de un problema entero (y elegir la mejor) se denomina **enumeración explícita**. En muchas ocasiones esto es impracticable.

La idea es enumerar de forma "inteligente" de modo que no sea necesario pasar por aquellas que sabemos que no son óptimas.

Ramificación y Acotamiento

B&B tiene ese objetivo: intentar una enumeración parcial que asegure pasar por una solución óptima. Se denomina **enumeración implícita**.

La idea de este algoritmo consiste en particionar el conjunto factible de la relajación del problema en cuestión y resolver los subproblemas resultantes de esta partición (ramificación).

A partir del análisis de las soluciones obtenidas para los subproblemas y de sus respectivos valores óptimos (o de cotas para los valores óptimos) se puede establecer si es posible obtener mejores soluciones dividiendo nuevamente algún subproblema (proceso de acotamiento).

Si se concluye que determinado subproblema no puede mejorar la solución actual, no se le ramifica. En caso contrario, será uno de los candidatos a ser ramificado.

Ramificación y Acotamiento

Veamos entonces el criterio más formalmente:
Tenemos el problema

$$\begin{array}{ll} \text{mín} & c^T x \\ \text{s.a.} & Ax \leq b \\ & x_j \in IN_0 \quad j \in I \subseteq \{1, \ldots, n\} \\ & x_k \geq 0 \quad k \in I^C \end{array}$$

Y asumimos que el conjunto factible es acotado.

Sea $R_0 = \{x / Ax \leq b, x_j \geq 0 \ \forall j\}$ y sea la relajación lineal de (PE)

$$\begin{array}{ll} (\mathrm{P}_0) & z_0 = \text{mín}\, c^T x \\ \text{s.a.} & x \in R_0 \end{array}$$

Sea x^0 una solución óptima de P_0, es decir $x^0 \in R_0$ y $c^T x^0 = z_0$.

Si $x_j^0 \in IN_0 \quad \forall j \in I \Rightarrow x^0$ es solución óptima también del (PE), problema resuelto.
Si no, existe un índice $k \in I$ tal que x_k^0 no pertenece a IN_0.

Ramificación y Acotamiento

Usando este valor podemos ramificar (P_0) en dos:

$$R_0^+ = R_0 \bigcap \{x / x_k \geq \lfloor x_k^0 \rfloor + 1\}$$

$$R_0^- = R_0 \bigcap \{x / x_k \leq \lfloor x_k^0 \rfloor\}$$

Sean entonces los siguientes problemas

$$\begin{array}{ll} (\mathrm{P}_0^+) & z_0^+ = \min c^T x \\ \text{s.a.} & x \in R_0^+ \end{array}$$

$$\begin{array}{ll} (\mathrm{P}_0^-) & z_0^- = \min c^T x \\ \text{s.a.} & x \in R_0^- \end{array}$$

Obviamente: $z_0^+ \geq z_0$ y $z_0^- \geq z_0$.

Además, si x^* es solución óptima del (PE), x^* debe estar en R_0^+ o en R_0^-, pues x_k^* debe ser entera.

Ramificación y Acotamiento

Ahora se puede tomar (P_0^+) y analizar su óptimo. Sea x^1 su solución óptima. Si $x_j^1 \in IN_0 \quad \forall j \in I$ es una solución entera factible. Es óptima para (PE)?
No necesariamente; en R_0^- podría haber una solución entera mejor.
Lo que si es seguro es que es una cota superior del óptimo.

Si x^1 no es factible para (PE) se puede repetir la ramificación.

De este modo se construye un árbol donde de cada nodo se derivan dos ramas. Eventualmente, en las ramas finales estarían todas las soluciones factibles de (PE). Sin embargo, un nodo del árbol puede no requerir más ramificaciones en cuyo caso se dice que se $poda$ esa rama.

Las razones para no ramificar son:

1. El problema en el nodo resulta infactible. Obviamente, cualquier ramificación será infactible.

Ramificación y Acotamiento

2. La solución en el nodo es factible para (PE). Por lo tanto cualquier ramificación no podría tener mejor solución entera. El valor entero obtenido debe ser guardado (si no es mejor que alguno que ya se tenga no hace falta). El valor de la mejor solución entera obtenida hasta el momento se denomina **incumbente**.

3. Si en un nodo del árbol el valor óptimo es z y el valor incumbente es $\overline{z}$ con $z \geq \overline{z}$, es claro que ramificar en ese nodo sólo daría soluciones enteras peores (o iguales) a la mejor obtenida hasta el momento.

En su forma estandar, el algoritmo maneja una lista L de problemas candidatos, que deben ser examinados para eventual ramificación.

Sea $\overline{x}$ la mejor solución entera disponible y $\overline{z}$ el valor de la función objetivo en $\overline{x}$ (o sea el valor del incumbente)

Algoritmo "Branch & Bound"

L: Lista de problemas candidatos a ser ramificados.

0. $L = \{(\mathrm{P}_0)\}$; $\overline{z} = +\infty$.

1. Escoger un problema (nodo) de la lista L.
   Sea (P) el problema seleccionado.
   Si $L = \phi$, TERMINAR con las siguientes conclusiones:
   - Si $\overline{z} = +\infty$, (PE) es infactible.
   - En caso contrario, la solución óptima de (PE) es $(\overline{x}, \overline{z})$.

2. Resolver el problema (P).
   Si (P) es infactible, eliminarlo de L (podar la rama que nace de (P)). Ir a (1).

3. Si (P) es factible, sea z' el valor óptimo y x' una solución óptima.
   Si $z' \geq \overline{z}$ eliminar (P) de L e ir a (1).

4. Si $z' < \overline{z}$ y x' cumple la condición de integralidad, entonces actualizar los valores $\overline{z} \longleftarrow z'$ y $\overline{x} \longleftarrow x'$, eliminar (P) de L e ir a (1).

Algoritmo "Branch & Bound"

5. En caso contrario, sea $k \in I$ tal que x'_k no pertenece a IN_0. Ramificar el problema (P) creando dos nuevos problemas: (P^+) y (P^-).
   (P^-) se construye agregando la restricción $x_k \leq \lfloor x'_k \rfloor$.
   (P^+) se construye agregando la restricción $x_k \geq \lfloor x'_k \rfloor + 1$
   Por último se actualiza $L \longleftarrow L \bigcup \{P^-, P^+\}$ y se va al paso (1).

Ej:

$$\begin{array}{lll} (\text{PE})\,\text{mín} & z = -5x_1 - 8x_2 & \\ \text{s.a.} & x_1 + x_2 \leq 6 & \\ & 5x_1 + 9x_2 \leq 45 & \\ & x_1, x_2 \geq 0, & \textit{enteros} \end{array}$$

Hacemos $\overline{z} = +\infty$.
(P_0): Relajación lineal

$$\begin{array}{lll} \text{mín}\, z = & -5x_1 - 8x_2 & \\ \text{s.a.} & x_1 + x_2 \leq 6 & \\ & 5x_1 + 9x_2 \leq 45 & \\ & x_1, x_2 \geq 0 & \end{array}$$

Algoritmo "Branch & Bound"

Resolvemos usando SIMPLEX:

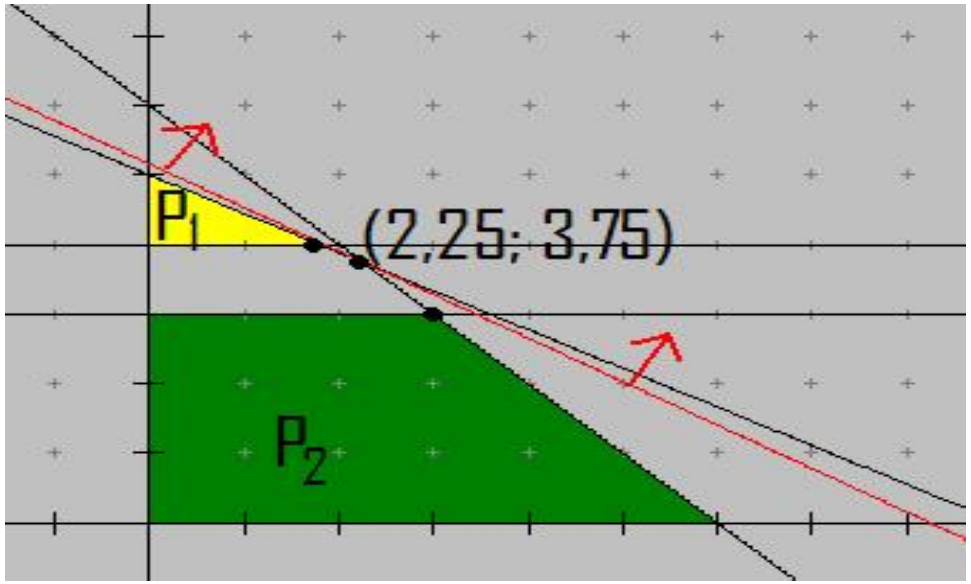
$$z_0 = -41,25 \quad x_1 = 2,25 \quad x_2 = 3,75$$

Vemos que x_1 y x_2 son no enteros.
Elegimos x_2 para ramificar, creando (P_1) y (P_2).

$$\begin{aligned} (\mathrm{P}_1)\ \text{mín}\, z = \quad & -5x_1 - 8x_2 \\ \text{s.a.} \quad & x_1 + x_2 \leq 6 \\ & 5x_1 + 9x_2 \leq 45 \\ & x_2 \geq 4 \\ & x_1, x_2 \geq 0 \end{aligned}$$

$$\begin{aligned} (\mathrm{P}_2)\ \text{mín}\, z = \quad & -5x_1 - 8x_2 \\ \text{s.a.} \quad & x_1 + x_2 \leq 6 \\ & 5x_1 + 9x_2 \leq 45 \\ & x_2 \leq 3 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Algoritmo "Branch & Bound"



Se resuelve (P_1) y se obtiene

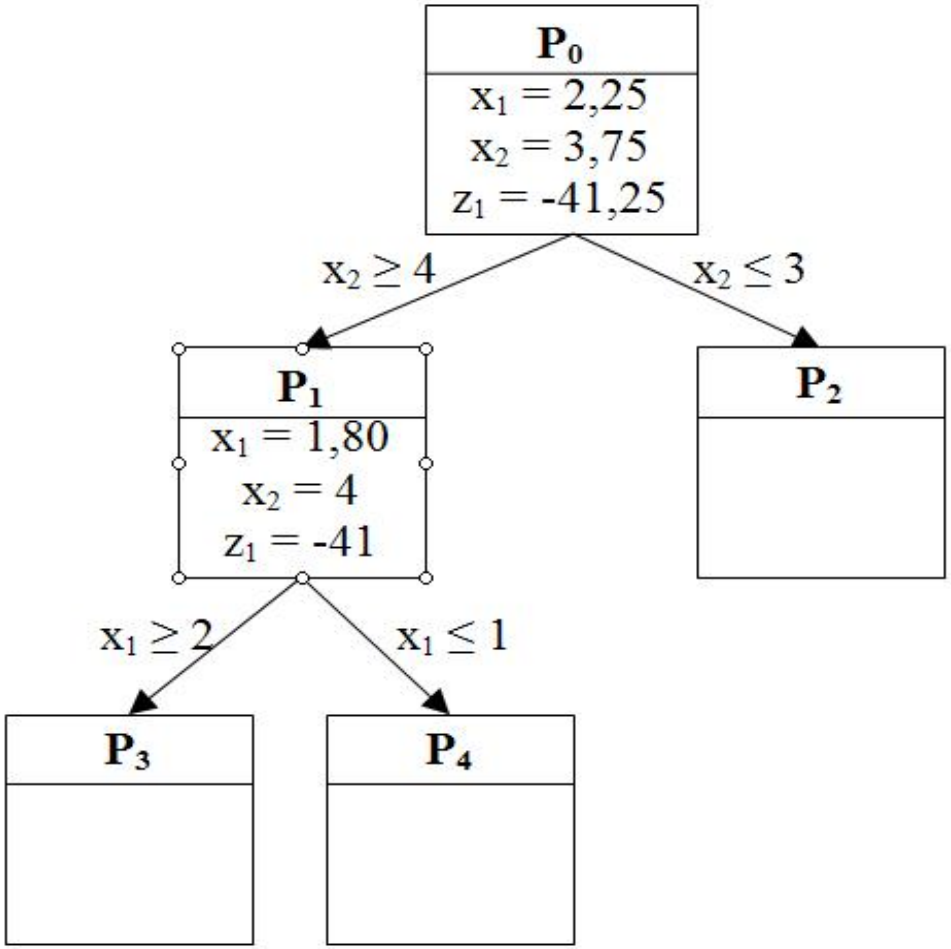
$$z_1 = -41 \quad x_1 = 1,8 \quad x_2 = 4$$

Podemos resolver (P_2) o ramificar (P_1). Hacemos esto último y ramificamos en la variable x_1.

Generamos entonces los problemas (P_3) y (P_4) agregando en un caso $x_1 \geq 2$ y en el otro $x_1 \leq 1$, respectivamente.

Algoritmo “Branch & Bound”

O sea que hasta el momento tenemos:



Algoritmo "Branch & Bound"

Hasta aquí tenemos que -41 es cota inferior para el valor óptimo del problema.

Resolvemos (P_3) y vemos que es infactible. Esto significa que el nodo (P_3) puede ser descartado y no ramificado.

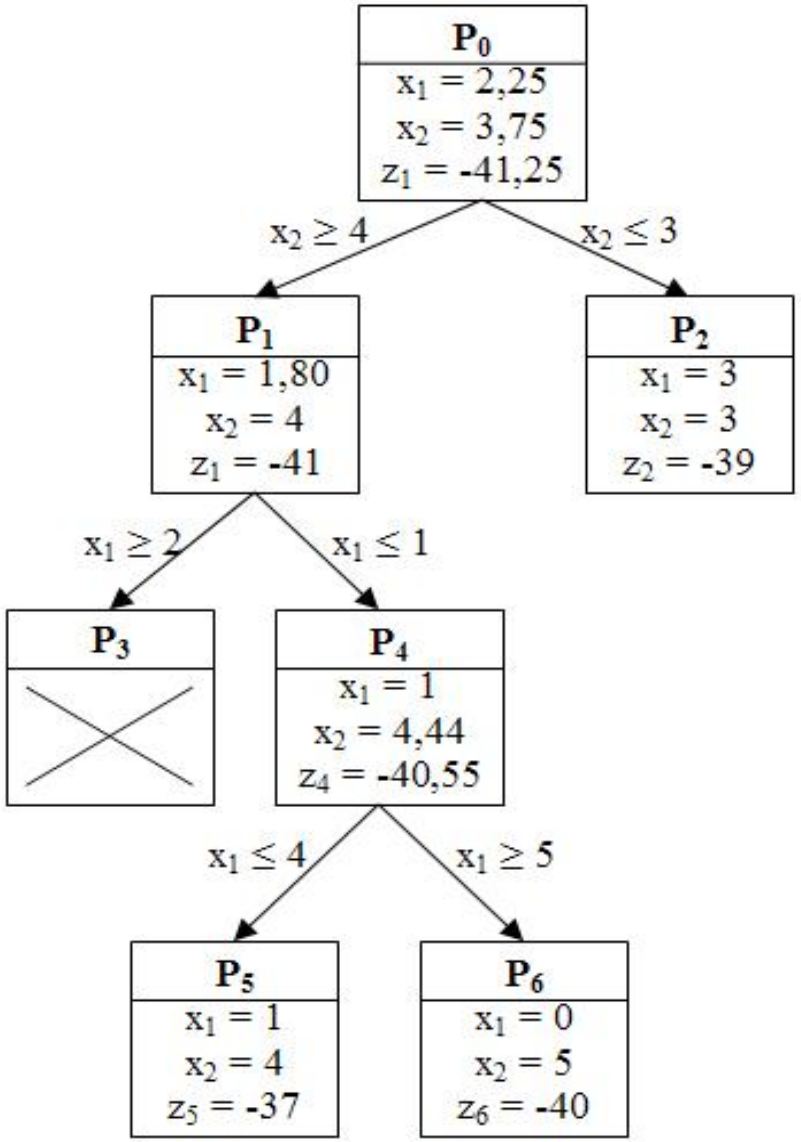
Seguimos ahora por (P_4).
La solución de (P_4) es $x_1 = 1$, $x_2 = 4,44$ y $z_4 = -40,55$. Esta solución aun no es entera por lo que subdividimos (P_4) con las restricciones $x_2 \leq 4$ y $x_2 \geq 5$, obteniendo (P_5) y (P_6), respectivamente.
El conjunto factible de (P_6) es sólo el punto $x_1 = 0$, $x_2 = 5$ y el de (P_5) es el segmento de recta que une el punto (0,4) con (1,4).

El óptimo de (P_5) da la primera solución entera ($x_1 = 1$, $x_2 = 4$, $z_5 = -37$). Tenemos ahora una cota superior del valor óptimo $\overline{z} = -37$.
Resolvemos (P_6) y tenemos $x_1 = 0$, $x_2 = 5$ y $z_6 = -40$, que mejora la solución actual. Así $\overline{z} = -40$.

Finalmente, resolvemos (P_2) y el óptimo es $x_1 = 3$, $x_2 = 3$, $z_2 = -39$.

Dado que $z_2 > \overline{z}$ y no hay más problemas para examinar, luego la solución de (P_6) es la solución general del problema.

Algoritmo “Branch & Bound”



Algunas Observaciones sobre el Algoritmo

1. Cuál variable elegir para la ramificación?

   Existen algunas técnicas que permiten estimar el grado de mejora de la función objetivo que se obtendrá al ramificar en cada una de las variables.

   Otra alternativa (heurística) es darle prioridad a aquella variable con menor c_j (mayor contribución para la mejora de la función objetivo).

2. Qué nodo elegir para ramificar?

   Podemos ramificar $a\ lo\ ancho$ o $en\ profundidad$. En principio, esta última estrategia garantiza mejorar más rápido la cota para la función objetivo.

   Igualmente, no hay forma de garantizar que la rama examinada incluya a la solución del problema.

3. Uso de una buena cota superior

   El incumbente inicia el algoritmo con un valor de $+\infty$ y va mejorando cuando se encuentran soluciones enteras factibles mejores.

Algunas Observaciones sobre el Algoritmo

Suele ser útil aplicar una heurística al problema, generalmente que devuelva una "buena"solución entera factible (y empezar con $\overline{z}$ en este valor).

4. Resolución eficiente de los problemas

   En general, los problemas lineales continuos asociados a cada nodo se resuelven sin mucho esfuerzo adicional dado que la diferencia con el nodo padre es solo una restricción adicional.

   El SIMPLEX Dual puede ser utilizado para reoptimizar a partir del nodo padre.

Programación Entera Binaria

Supongamos ahora que las variables pueden tomar sólo valores 0 ó 1.

Ahora no se tienen que agregar restricciones adicionales a los subproblemas sino determinar que una variable toma valor 0 o toma valor 1.

El número máximo de problemas a examinar es $O(2^n)$, para un problema con n variables binarias (profundidad n).

La relajación lineal se obtiene reemplazando cada restricción de binariedad por $0 \leq x_j \leq 1$ para cada variable.

Veamos un ejemplo:

$$\begin{aligned} (\text{PE})\, \text{máx}\, z = \quad & 2x_1 + x_2 + 2x_3 & \\ \text{s.a.} \quad & x_1 + 2x_2 + x_3 & \leq 3 \\ & 2x_1 + 2x_2 + 2x_3 & \leq 3 \\ & x_1 + 2x_2 + 2x_3 & \leq 4 \\ & x_1, x_2, x_3 \in \{0, 1\} & \end{aligned}$$

Programación Entera Binaria

Sea (P_0) la relajación lineal.

La solución óptima viene dada por:

$$x_1^* = 1 \quad x_2^* = 0 \quad x_3^* = \frac{1}{2} \quad z_0 = 3$$

Ramificamos en x_3 (única posibilidad):

En (P_1), agregamos $x_3 = 0$.

En (P_2), agregamos $x_3 = 1$.

Luego de resolver el problema por B&B el árbol queda:

Programación Entera Binaria