Introducción a la Programación de Restricciones y Optimización

Tutorial

Septiembre 2005

BUAP, Puebla, México

John Hooker

Carnegie Mellon University

Agenda

- Examinaremos **5 problemas ejemplo** y los usaremos para ilustrar algunas ideas básicas de **programación de restricciones** y **optimización**.
  - Transferencia de carga
  - Problema del agente viajero
  - Optimización global continua
  - Configuración de producto
  - Asignación de máquinas

Agenda

- Ejemplo: **Transferencia de carga**
  - Propagación de cotas
  - Consistencia de cotas
  - Cortes “knapsack” (mochila)
  - Relajación lineal
  - Búsqueda por ramificación (“branching search”)

Agenda

- Ejemplo: **Problema del agente viajero**
  - Filtración por restricciones **"todas diferentes"**
  - Relajación de la condición **"todas diferentes"**
  - Arco-consistencia

Agenda

- Ejemplo: **Optimización global continua**
  - Propagación no lineal de cotas
  - Factorización de funciones
  - División por intervalos
  - Propagación lagrangeana
  - Fijación de variable por costo reducido

Agenda

- Ejemplo: **Configuración de producto**
  - Indices variables
  - Filtración por restricción de **elemento**
  - Relajación de la restricción **elemento**
  - Relajación de una disjunción de sistemas lineales

Agenda

- Ejemplo: **Asignación de máquinas**
  - "Edge finding"
  - Descomposición de Benders y "nogoods"

Ejemplo: Transferencia de Carga

- Propagación de cotas
- Consistencia de cotas
- Cortes “knapsack” (mochila)
- Relajación
- Búsqueda por ramificación (“branching search”)

Transferencia de Carga

- Transportar 42 toneladas de carga usando a lo sumo 8 camiones.
- Los camiones son de 4 capacidades diferentes: 7,5,4 y 3 tons.
- Cuántos camiones de cada capacidad deben ser usados para minimizar costos?
- x_j = número de camiones de capacidad *j.*

Costo del camión de capacidad 4

$$\min 90x_1 + 60x_2 + 50x_3 + 40x_4$$

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

$$x_j \in \{0,1,2,3\}$$

Transferencia de Carga

- Resolveremos el problema usando "branching search".
- En cada nodo del árbol de búsqueda:
  - Reducir los dominios de las variables usando *propagación de cotas*.
  - Resolver una *relajación lineal* del problema para obtener una cota para el valor óptimo.

Propagación de Cotas

- El *dominio* de x_j es el conjunto de valores que puede tomar x_j para alguna solución factible.
  - Inicialmente el dominio de cada x_j es {0,1,2,3}.
- La *propagación de cotas* reduce los dominios.
  - Dominios de menor tamaño significan un menor número de bifurcaciones.

Propagación de Cotas

- Primero, reducir el dominio de x_1:

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

$$\Downarrow$$

$$x_1 \geq \left\lceil \frac{42 - 5x_2 + 4x_3 + 3x_4}{7} \right\rceil$$

$$\Downarrow$$

Max. elemento del dominio

$$x_1 \geq \left\lceil \frac{42 - 5 \cdot 3 - 4 \cdot 3 - 3 \cdot 3}{7} \right\rceil = \left\lceil \frac{6}{7} \right\rceil = 1$$

- Entonces, el dominio de x_1 se reduce de {0,1,2,3} a {1,2,3}.
- No es posible reducir los dominios de x_2, x_3, x_4.

Propagación de Cotas

- En general, sea $\{L_i, \ldots, U_i\}$ el dominio de x_i.
- Una desigualdad $ax \geq b$ (para $a \geq 0$) puede ser usada para aumentar L_i a

$$\max\left\{ L_i, b - \frac{\sum_{j \neq i} a_j U_j}{a_i} \right\}$$

- Una desigualdad $ax \leq b$ (para $a \geq 0$) puede ser usada para reducir U_i a

$$\min\left\{ U_i, b - \frac{\sum_{j \neq i} a_j L_j}{a_i} \right\}$$

Propagación de Cotas

- Ahora, **propagar** el dominio reducido a las otras restricciones para quizás reducir más el dominio:

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

$$\Downarrow$$

$$x_1 \leq \left\lceil \frac{8 - x_2 - x_3 - x_4}{1} \right\rceil$$

$$\Downarrow$$

$$x_1 \leq \left\lceil \frac{8 - 1 - 0 - 0}{1} \right\rceil = \left\lceil \frac{7}{1} \right\rceil = 7$$

Min elemento del dominio

- No son posibles reducciones adicionales.

Consistencia de Cotas

- Nuevamente, sea $\{L_j, \ldots, U_j\}$ el dominio de x_j
- Un conjunto de restricciones es "**consistente en cotas"** si para cada j :
  - $x_j = L_j$ pertenece a alguna solución factible y
  - $x_j = U_j$ pertenece a alguna solución factible.
- Consistencia de cotas $\Rightarrow$ no asignaremos a x_j ningún valor no factible durante la bifurcación.
- La propagación de cotas es "consistente en cotas" para el caso de una **única desigualdad**.
  - $7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$ es "consistente en cotas" cuando los dominios son $x_1 \in \{1,2,3\}$ y $x_2, x_3, x_4 \in \{0,1,2,3\}$.
- Pero no necesariamente para un **conjunto** de desigualdades.

Consistencia de Cotas

- La propagación de cotas puede no alcanzar consistencia para un conjunto.
- Consideremos el conjunto de desigualdades

$$x_1 + x_2 \geq 1$$
$$x_1 - x_2 \geq 0$$

con dominios x_1, $x_2 \in \{0,1\}$, soluciones $(x_1,x_2) = (1,0), (1,1)$.

- La propagación de cotas no tiene efecto sobre los dominios.

En $x_1 + x_2 \geq 1$:

$$x_1 \geq \lceil 1 - U_2 \rceil = \lceil 1 - 1 \rceil = 0$$
$$x_2 \geq \lceil 1 - U_1 \rceil = \lceil 1 - 1 \rceil = 0$$

En $x_1 - x_2 \geq 1$:

$$x_1 \geq \lceil 1 - U_2 \rceil = \lceil 1 - 1 \rceil = 0$$
$$x_2 \leq \lfloor 1 + L_1 \rfloor = \lfloor 1 + 0 \rfloor = 1$$

- El conjunto de restricciones no es "consistente en cotas" porque $x_1 = 0$ no es una solución factible.

Cortes "Knapsack"

- Las restricciones en desigualdad (restricciones "**knapsack**") implican **planos secantes**.
- Los planos secantes restringen la relajación lineal aun más, y su solución es una cota más restrictiva.
- Para la restricción ≥ , cada **paquete máximo** corresponde a un plano secante (**corte "knapsack"**).

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

Estos términos forman un paquete máximo porque:

(a) Por sí solos no pueden sumar ≥ 42, aun si cada x_j toma el mayor valor en su dominio (3).

(b) No existe un súper conjunto de estos términos con esta propiedad.

Cortes "Knapsack"

- El máximo paquete:

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

corresponde al corte "knapsack":

$$x_3 + x_4 \geq \left\lceil \frac{42 - 7 \cdot 3 - 5 \cdot 3}{\max\{4,3\}} \right\rceil = 2$$

Min. valor de $4x_3 + 3x_4$ requerido para satisfacer la desigualdad

Coeficientes de x_3, x_4

Cortes "Knapsack"

- En general, *J* es un paquete de $ax \geq b$ (para $a \geq 0$) si

$$\sum_{j \in J} a_j U_j < b$$

- Si *J* es un paquete de $ax \geq b$, entonces lor términos restantes deben cubrir la diferencia:

$$\sum_{j \notin J} a_j x_j \geq b - \sum_{j \in J} a_j U_j$$

- Así tenemos el corte "knapsack":

$$\sum_{j \notin J} x_j \geq \left\lceil \frac{b - \sum_{j \in J} a_j U_j}{\max_{j \notin J} \{a_j\}} \right\rceil$$

Cortes "Knapsack"

- Los cortes "knapsack" válidos de

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

son

$$x_3 + x_4 \geq 2$$

$$x_2 + x_4 \geq 2$$

$$x_2 + x_3 \geq 3$$

Relajación Lineal

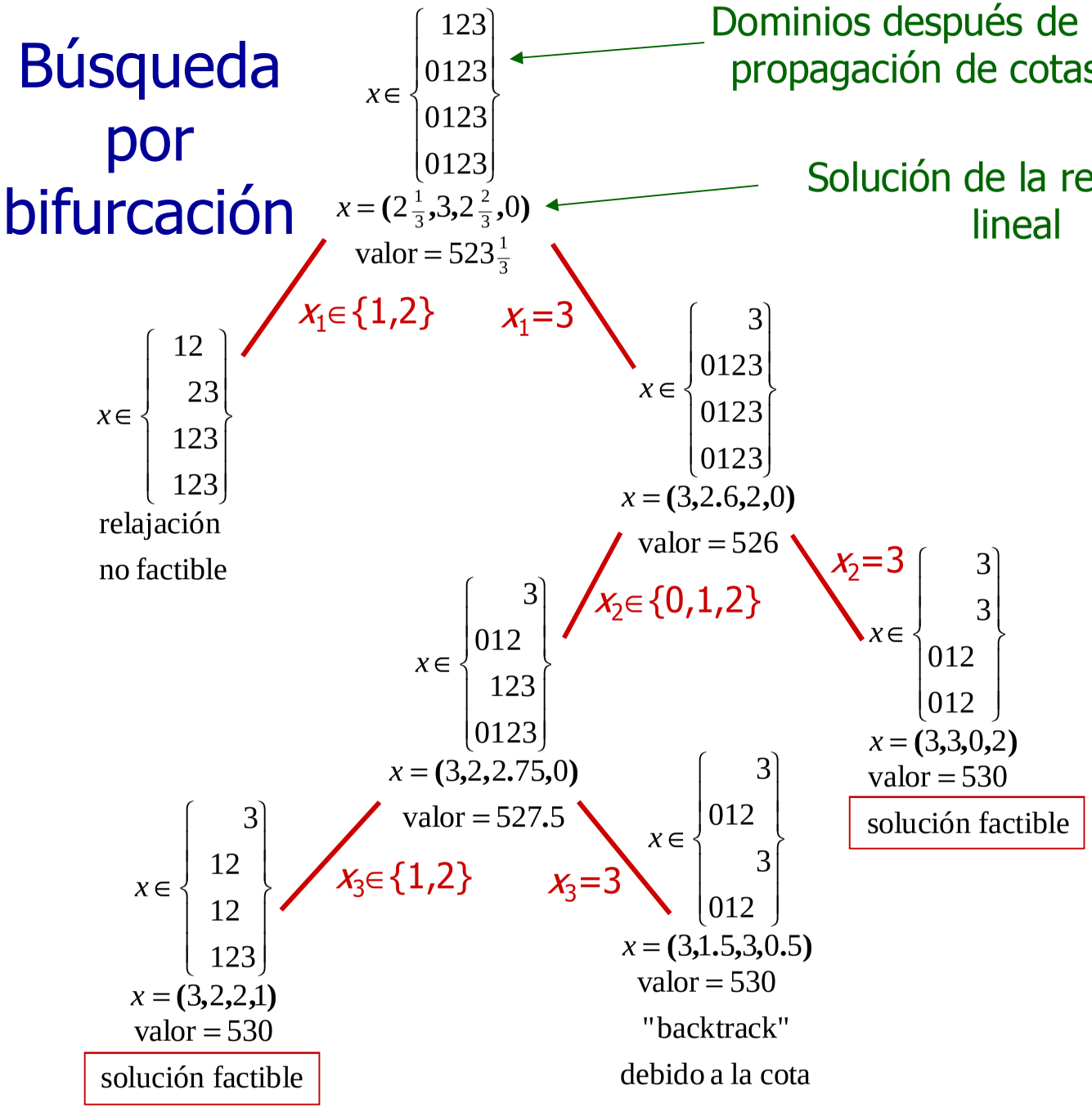
- Tenemos ahora una relajación lineal del problema de transferencia de carga:

$$\min 90x_1 + 60x_2 + 50x_3 + 40x_4$$
$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$
$$x_1 + x_2 + x_3 + x_4 \leq 8$$
$$x_3 + x_4 \geq 2$$
$$x_2 + x_4 \geq 2$$
$$x_2 + x_3 \geq 3$$
$$L_j \leq x_j \leq U_j$$

Búsqueda por Bifurcación ("Branching Search")

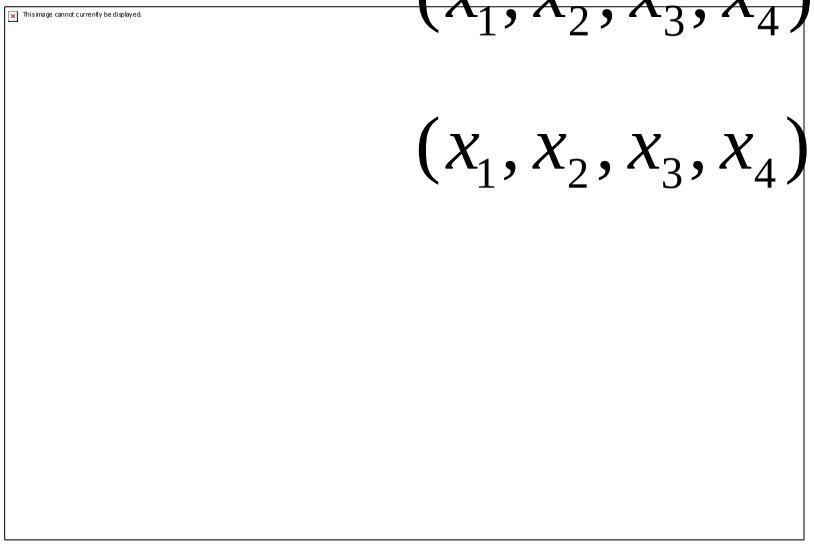
- En cada nodo del árbol de búsqueda:
  - Reducir el dominio por medio de propagación de cotas.
  - Solucionar una relajación lineal para obtener una cota inferior para cualquier solución factible en el sub-árbol que empieza en ese nodo.
- Si la relajación es no factible, o su valor óptimo no mejora la mejor solución factible encontrada hasta ahora, entonces "**backtrack**".
- De lo contrario, si la solución de la relajación es factible, guardarla y "**backtrack**".
- De lo contrario, **bifurcar** dividiendo el dominio.

Búsqueda por bifurcación



Búsqueda por Bifurcación

- Dos soluciones óptimas fueron encontradas (costo = 530):



$$(x_1, x_2, x_3, x_4) = (3,2,2,1)$$

$$(x_1, x_2, x_3, x_4) = (3,3,0,2)$$

Ejemplo: Agente Viajero

- Filtración por restricciones **todas diferentes**
- Relajación de la condición "**todas diferentes**"
- Arco-consistencia

Agente Viajero

- El agente visita cada ciudad una sola vez.
- La distancia de la ciudad i a la ciudad j es c_{ij}.
- Minimizar la distancia recorrida.
- x_i = i-ésima ciudad visitada.

Distancia desde la i-ésima ciudad visitada hasta la siguiente ciudad (ciudad n+1 = ciudad 1)

$$\min \sum_i c_{x_i x_{i+1}}$$

$$(x_1, \ldots, x_n) \text{ todas diferentes}$$

$$x_j \in \{1, \ldots, n\}$$

- Puede ser resuelto por bifurcación + reducción de dominio + relajación.

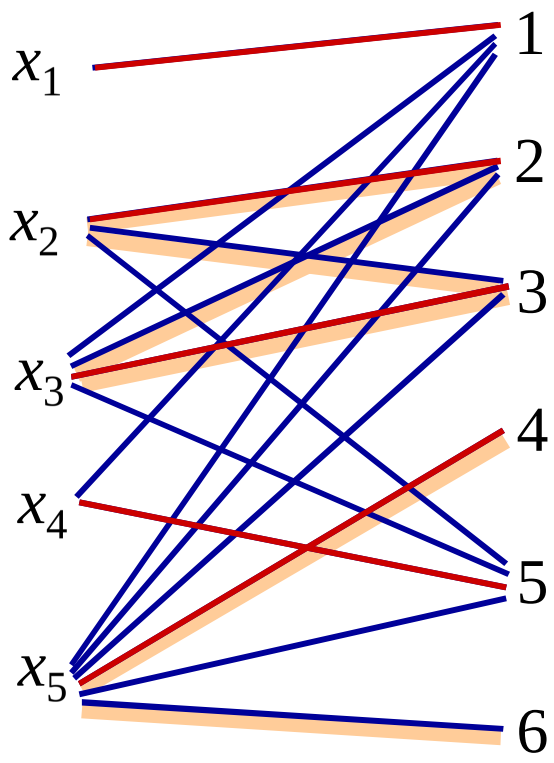
Filtración por "Todas Diferentes"

- Objetivo: **filtrar** valores no factibles en los dominios de las variables.
  - La filtración reduce el dominio y por lo tanto reduce el número de bifurcaciones necesarias.
- El mejor algoritmo conocido de filtración por "todas diferentes" está basado en el **aparemiento bipartito de máxima cardinalidad** y el teorema de Berge.

Filtración por "Todas Diferentes"

- Consideremos (x_1,x_2,x_3,x_4,x_5) todas diferentes con dominios

$$x_1 \in \{1\}$$
$$x_2 \in \{2,3,5\}$$
$$x_3 \in \{1,2,3,5\}$$
$$x_4 \in \{1,5\}$$
$$x_5 \in \{1,2,3,4,5,6\}$$



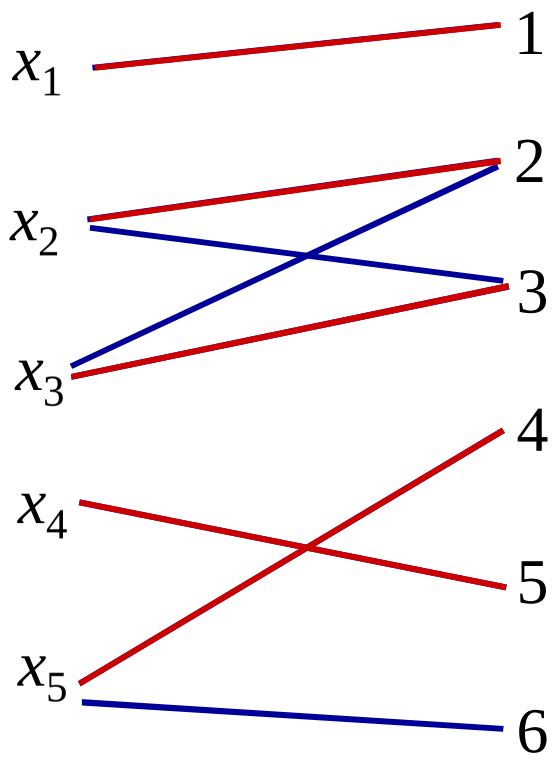
Indicar el dominio con trazos.

Encontrar el apareamiento bipartito de máxima cardinalidad.

Marcar trazos en trayectorias alternas que comienzan en vértices no cubiertos.

Marcar trazos en ciclos alternos.

Eliminar trazos no marcados o apareados.



Indicar el dominio con trazos.

Encontrar el apareamiento bipartito de máxima cardinalidad.

Marcar trazos en trayectorias alternas que comienzan en vértices no cubiertos.

Marcar trazos en ciclos alternos.

Eliminar trazos no marcados o apareados.

Filtración por "Todas Diferentes"

$$x_1 \in \{1\}$$

$$x_2 \in \{2,3,5\} \Rightarrow x_2 \in \{2,3\}$$

$$x_3 \in \{1,2,3,5\} \Rightarrow x_3 \in \{2,3\}$$

$$x_4 \in \{1,5\} \Rightarrow x_4 \in \{5\}$$

$$x_5 \in \{1,2,3,4,5,6\} \Rightarrow x_5 \in \{4,6\}$$

Relajación de "Todas Diferentes"

- (x_1,x_2,x_3,x_4) todas diferentes con dominios $x_j \in$ {1,2,3,4} tiene la relajación de **envolvente convexa**:

$$x_1 + x_2 + x_3 + x_4 = 1+2+3+4$$

$$\left\{ \begin{array}{l} x_1 + x_2 + x_3 \\ x_1 + x_2 + x_4 \\ x_1 + x_3 + x_4 \\ x_2 + x_3 + x_4 \end{array} \right\} \geq 1+2+3$$

$$x_j \geq 1$$

$$\left\{ \begin{array}{l} x_1 + x_2 \\ x_1 + x_3 \\ x_1 + x_4 \\ x_2 + x_3 \\ x_2 + x_4 \\ x_3 + x_4 \end{array} \right\} \geq 1+2$$

- Aunque esta es la relajación lineal más ajustada ("tightest") posible, puede ser demasiado débil (e involucrar demasiadas desigualdades) limitando su utilidad.

Arco-Consistencia

- El conjunto de restricciones S que contiene las variables $x_1, \ldots, x_n$ con dominios $D_1, \ldots, D_n$ es **arco-consistente** si el dominio no contiene valores no factibles.
- O sea, para todo j, y para todo $v \in D_j$, $x_j = v$, para alguna solución factible de S.
  - De hecho, se trata de **arco-consistencia generalizada** (dado que puede haber mas de 2 variables por restricción).
- La arco-consistencia se satisface filtrando todas las soluciones no factibles de los dominios.

Arco-Consistencia

- El algoritmo de apareamiento satisface arco-consistencia para "todas diferentes".
- Con frecuencia los algoritmos de filtración en la práctica no satisfacen arco-consistencia completamente ya que alcanzarla no justifica la inversión en tiempo.
- Las **herramientas básicas de programación de restricciones** son algoritmos de filtración que satisfacen o aproximan arco-consistencia.
    - Los algoritmos de filtración son análogos a los algoritmos de plano secante en programación entera.

Arco-Consistencia

- Los dominios que son reducidos a través de un algoritmo de filtración para una restricción pueden ser **propagados** a otras restricciones.
  - Similar a la propagación de cotas.
- Pero la propagación puede no satisfacer arco-consistencia para un **conjunto** de restricciones, aun cuando la arco-consistencia es satisfecha para cada restricción por separado.
  - De nuevo, similar a la propagación de cotas.

Arco-Consistencia

- Por ejemplo, consideremos el conjunto de restricciones

(x_1, x_2) todas diferentes
(x_1, x_3) todas diferentes
(x_2, x_3) todas diferentes

con dominios

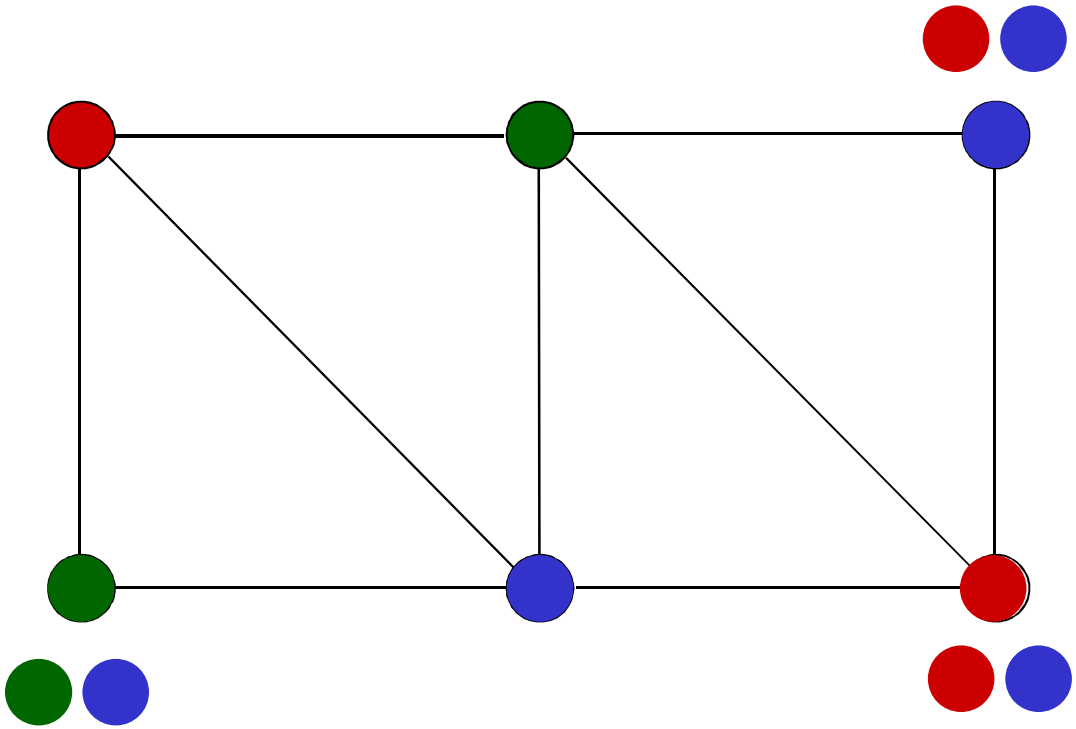
$x_1 \in \{1,2\}$
$x_2 \in \{2,3\}$
$x_3 \in \{2,3\}$

- No es posible reducir más el dominio.
  - Cada restricción es arco-consistente para sus respectivos dominios.
- Sin embargo, $x_1 = 2$ no es una solución factible para el conjunto de restricciones.
  - Por lo tanto, el conjunto de restricciones no es arco-consistente.

Arco-Consistencia

- Por otro lado, en ocasiones el mantener arco-consistencia puede resolver un problema — sin necesidad de bifurcación.
- Consideremos el problema de coloración de grafos.
  - Colorear los vértices de rojo, azul o verde de tal manera que todos los vértices adyacentes entre sí tengan colores diferentes.
  - Estas son restricciones **binarias** (contienen solo 2 variables).
  - Arbitrariamente colorear dos vértices adyacentes de rojo y verde.
  - Reducir los dominios para mantener arco-consistencia para cada restricción.

Problema de coloración de grafos que puede ser resuelto manteniendo arco-consistencia.



Ejemplo: Optimización Global Continua

- Propagación no lineal de cotas
- Factorización de funciones
- División por intervalos
- Propagación lagrangeana
- Fijación de variable por costo reducido

Optimización Global Continua

- Actualmente, los **solucionadores de programación de restricciones** (CHIP, ILOG Solver) hacen énfasis en **propagación**.
- Actualmente, los **solucionadores de programación entera** (CPLEX, Xpress-MP) hacen énfasis en **relajación**.
- Actualmente, los **solucionadores globales** (BARON, LGO) combinan propagación y relajación.
- Quizás en el futuro cercano **todos los solucionadores** combinaran propagación y relajación.

Optimización Global Continua

- Consideremos el problema de optimización global continua:

$$\max x_1 + x_2$$
$$2x_1 + x_2 \leq 2$$
$$4x_1x_2 = 1$$
$$x_1 \in [0,1], \quad x_2 \in [0,2]$$

- El conjunto factible es **no convexo**, y existe más de un óptimo local.

- Técnicas de programación **no lineal** tratan de encontrar un óptimo local.

- Técnicas de optimización global tratan de encontrar un **óptimo global**.

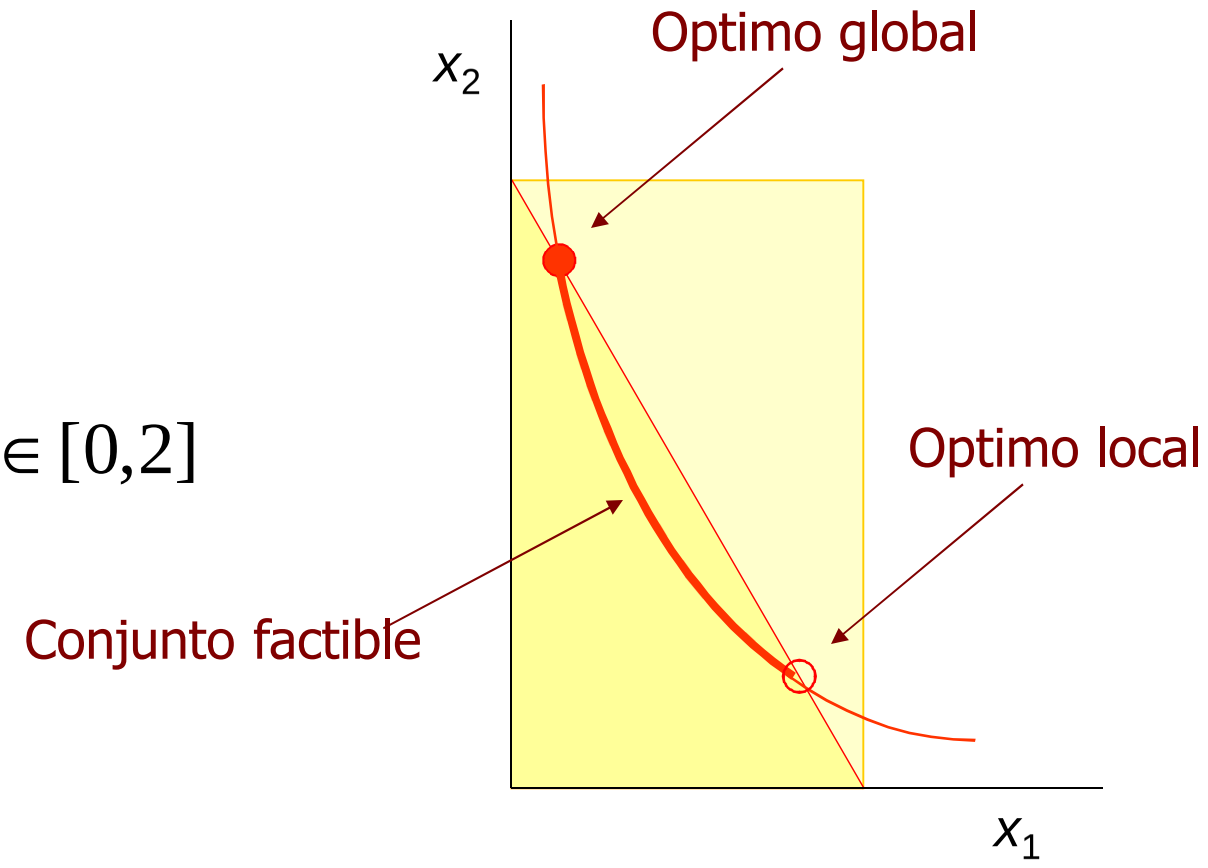
Optimización Global Continua

$$\max x_1 + x_2$$

$$4x_1x_2 = 1$$

$$2x_1 + x_2 \leq 2$$

$$x_1 \in [0,1],\ x_2 \in [0,2]$$



Optimización Global Continua

- **Bifurcar** dividiendo dominios de intervalos continuos.
- **Reducir los dominios** a través de:
  - Propagación no lineal de cotas.
  - Propagación lagrangeana (fijación de variable por costo reducido).
- **Relajar** el problema factorizando funciones.

Propagación No Lineal de Cotas

- Para propagar $4x_1x_2 = 1$, expresarla como $x_1 = 1/4x_2$ y $x_2 = 1/4x_1$. Entonces:

$$x_1 \geq \frac{1}{4U_2} = \frac{1}{4 \cdot 2} = \frac{1}{8} \qquad x_2 \geq \frac{1}{4U_1} = \frac{1}{4 \cdot 1} = \frac{1}{4}$$

- Esto implica dominios $x_1 \in [0.125,1]$, $x_2 \in [0.25,2]$.

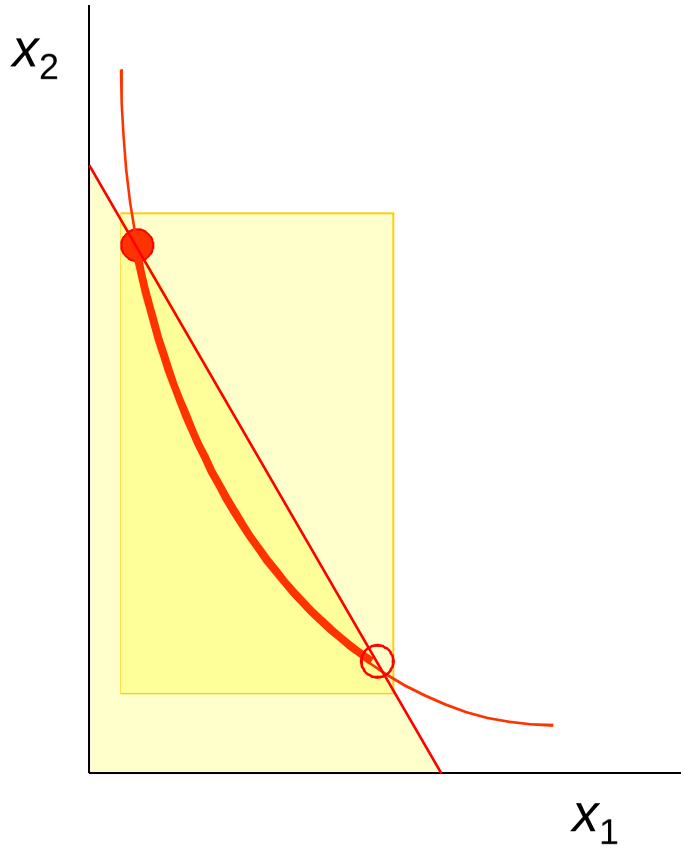
- Propagaciones adicionales de $2x_1 + x_2 \leq 2$ implican $x_1 \in [0.125,0.875]$, $x_2 \in [0.25,1.75]$.

- La iteración de las 2 restricciones converge en el punto fijo $x_1 \in [0.146,0.854]$, $x_2 \in [0.293,1.707]$.

  - En la práctica, el proceso de iteración se detiene antes de alcanzar convergencia, debido a que alcanzarla puede no compensar los costos en tiempo computacional.

Propagación No Lineal de Cotas

Propagar los intervalos
[0,1], [0,2]
a través de las
restricciones, para obtener
[1/8,7/8], [1/4,7/4]



Factorización de Funciones

- Descomponer funciones complejas en funciones elementales con relajaciones lineales conocidas.
- Expresar $4x_1x_2 = 1$ como $4y = 1$ donde $y = x_1x_2$.
  - Esto factoriza $4x_1x_2$ en la función lineal $4y$, y la función bilineal x_1x_2.
  - La relajación lineal de la función lineal $4y$ es ella misma.
  - La función bilineal $y = x_1x_2$ tiene la relajación lineal:

$$L_2x_1 + L_1x_2 - L_1L_2 \le y \le L_2x_1 + U_1x_2 - U_1L_2$$

$$U_2x_1 + U_1x_2 - U_1U_2 \le y \le U_2x_1 + L_1x_2 - L_1U_2$$

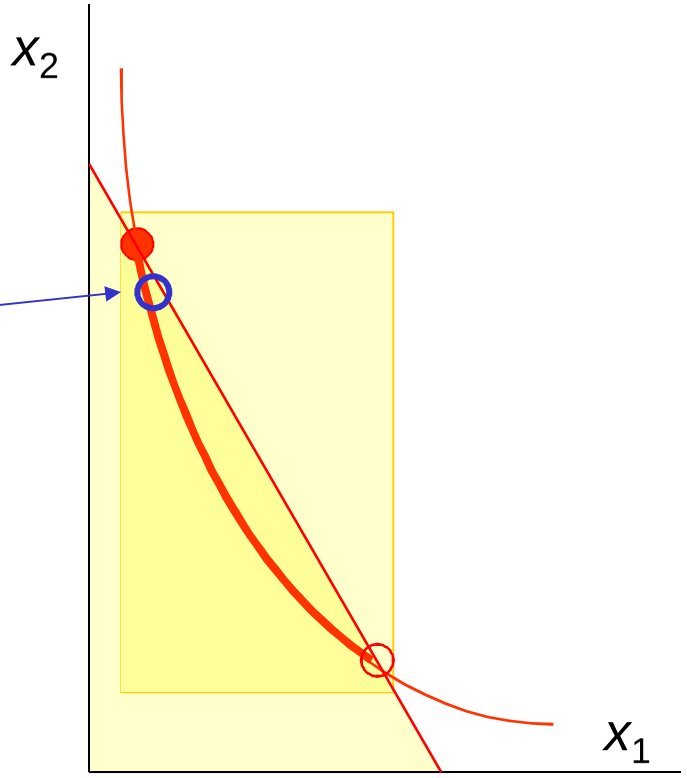
Factorización de Funciones

- Tenemos ahora la relajación lineal del problema no lineal:

$$\max x_1 + x_2$$
$$4y = 1$$
$$2x_1 + x_2 \leq 2$$
$$L_2 x_1 + L_1 x_2 - L_1 L_2 \leq y \leq L_2 x_1 + U_1 x_2 - U_1 L_2$$
$$U_2 x_1 + U_1 x_2 - U_1 U_2 \leq y \leq U_2 x_1 + L_1 x_2 - L_1 U_2$$
$$L_j \leq x_j \leq U_j$$

División de Intervalos

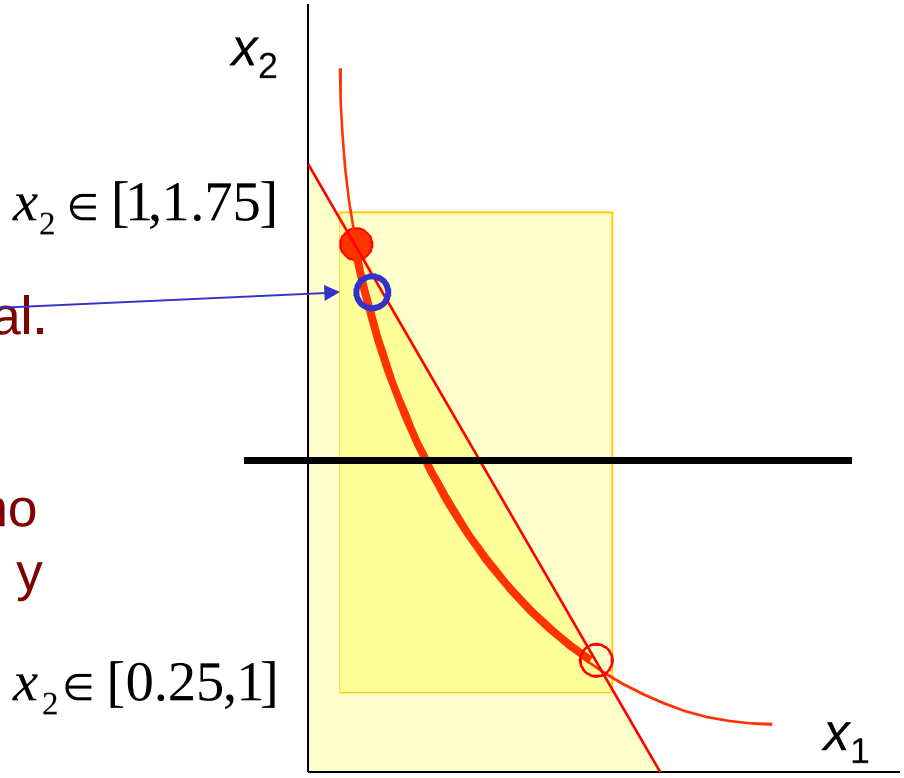
Resolver la relajación lineal.

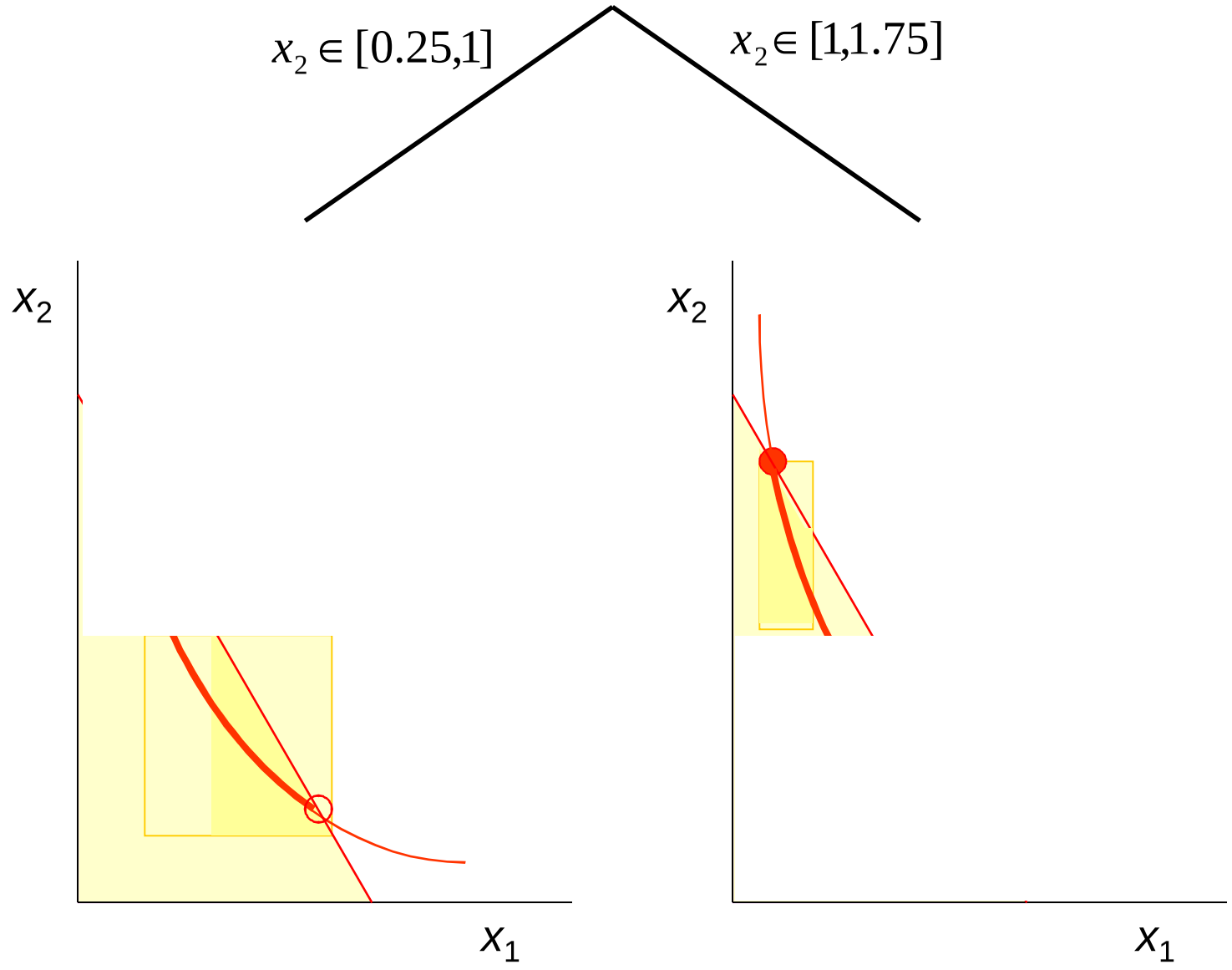


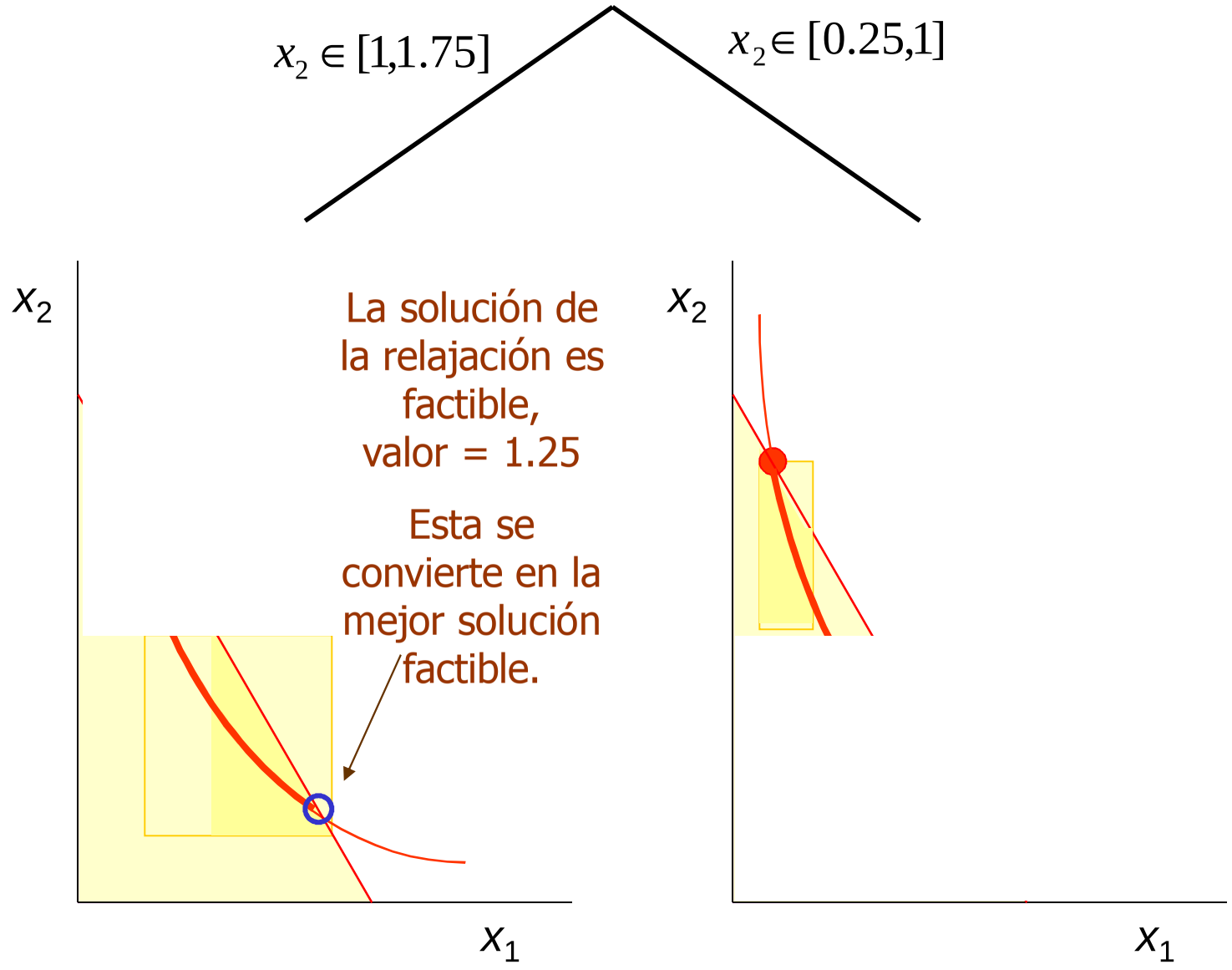
División de Intervalos

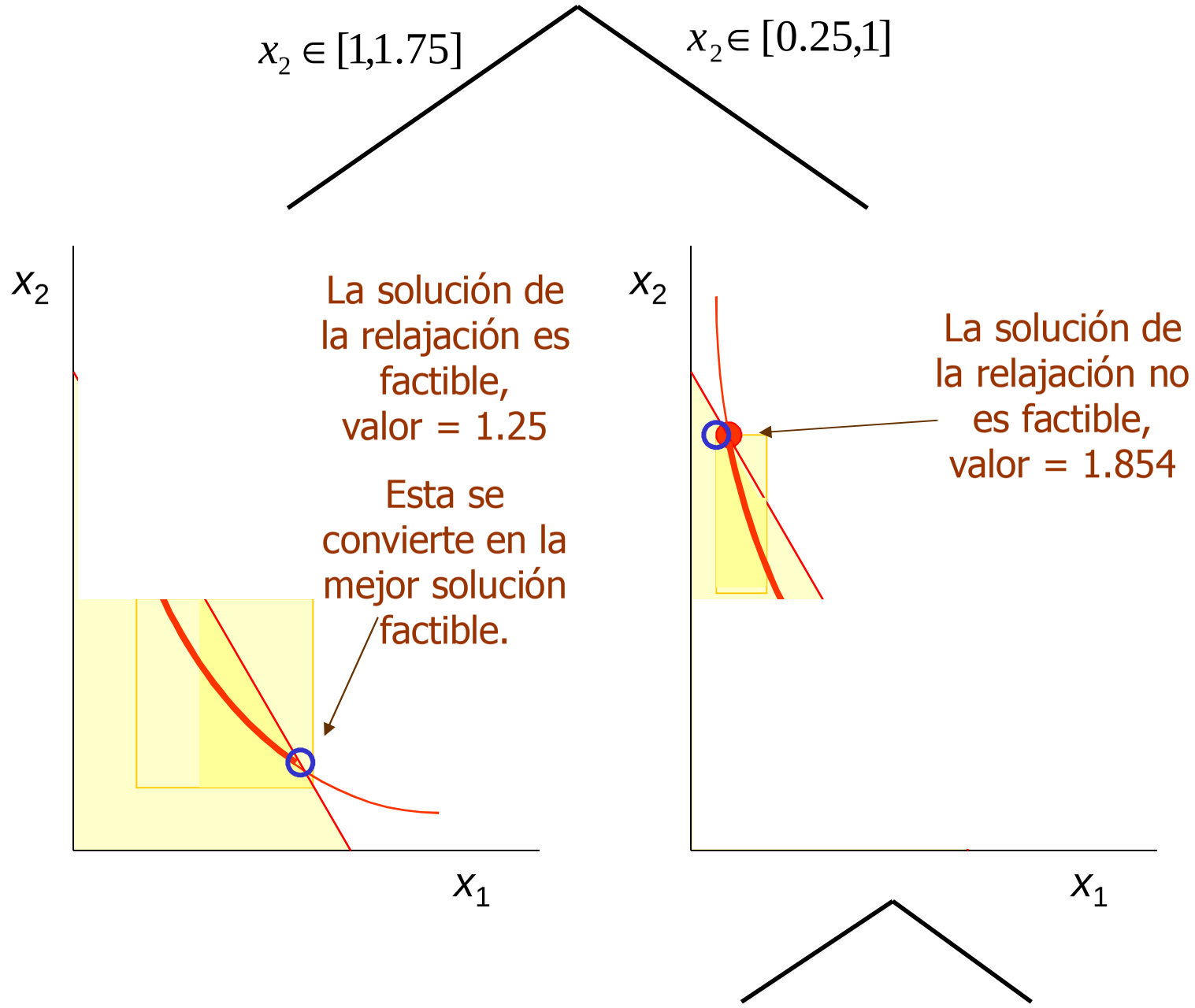
Resolver la relajación lineal.

Dado que la solución es no factible, dividir el intervalo y bifurcar.




$x_2 \in [0.25,1]$
$x_2 \in [1,1.75]$
x_2
x_1
x_2
x_1


$x_2 \in [1,1.75]$
$x_2 \in [0.25,1]$
x_2
La solución de la relajación es factible, valor = 1.25
Esta se convierte en la mejor solución factible.
x_1
x_2
x_1


$x_2 \in [1,1.75]$
$x_2 \in [0.25,1]$
x_2
La solución de la relajación es factible, valor = 1.25
Esta se convierte en la mejor solución factible.
x_1
x_2
La solución de la relajación no es factible, valor = 1.854
x_1

Propagación Lagrangeana

$$\max x_1 + x_2$$

El valor óptimo de la relajación es 1.854

$$4y = 1$$

El multiplicador de Lagrange (variable dual) en la solución de la relajación es 1.1

$$2x_1 + x_2 \le 2$$

$$L_2 x_1 + L_1 x_2 - L_1 L_2 \le y \le L_2 x_1 + U_1 x_2 - U_1 L_2$$

$$U_2 x_1 + U_1 x_2 - U_1 U_2 \le y \le U_2 x_1 + L_1 x_2 - L_1 U_2$$

$$L_j \le x_j \le U_j$$

- Cualquier reducción Δ en el lado derecho de $2x_1 + x_2 \le 2$ reduce el valor óptimo de la relajacion en 1.1 Δ.
- Entonces, cualquier reducción Δ en el lado izquierdo de la desigualdad (ahora 2) tiene el mismo efecto.

Propagación Lagrangeana

- Cualquier reducción Δ en el lado izquierdo de $2x_1 + x_2 \leq 2$ reduce el valor óptimo de la relajación (ahora 1.854) en 1.1 Δ.
- El valor óptimo no debe ser reducido por debajo del límite inferior 1.25 (valor de la mejor solución factible hasta ahora).
- Así, debemos tener $1.1\,\Delta \leq 1.854 - 1.25$, o

$$1.1[2-(2x_1+x_2)] \leq 1.854-1.25$$

o $$2x_1+x_2 \geq 2-\frac{1.854-1.25}{1.1}=1.451$$

- La desigualdad puede ser propagada para reducir los dominios.
  - Reducir el dominio de x_2 de [1,1.714] a [1.166,1.714].

Fijación de Variable por Costo Reducido

- En general, dada la restricción $g(x) \le b$ con multiplicador de Lagrange λ, la siguiente restricción es válida

Valor óptimo de la relajacion

Límite inferior en el valor óptimo del problema original

$$g(x) \ge b - \frac{v - L}{\lambda}$$

- Si $g(x) \le b$ es una restricción de no negatividad $-x_j \le 0$ con multiplicador de Lagrange λ (i.e, el costo reducido de x_j es $-\lambda$), entonces

$$-x_j \ge -\frac{v - L}{\lambda} \qquad \text{or} \qquad x_j \le \frac{v - L}{\lambda}$$

- Si x_j es una variable 0-1 y $(v - L)/\lambda < 1$, entonces podemos fijar x_j en 0. Esto se conoce como **fijación de variable por costo reducido.**

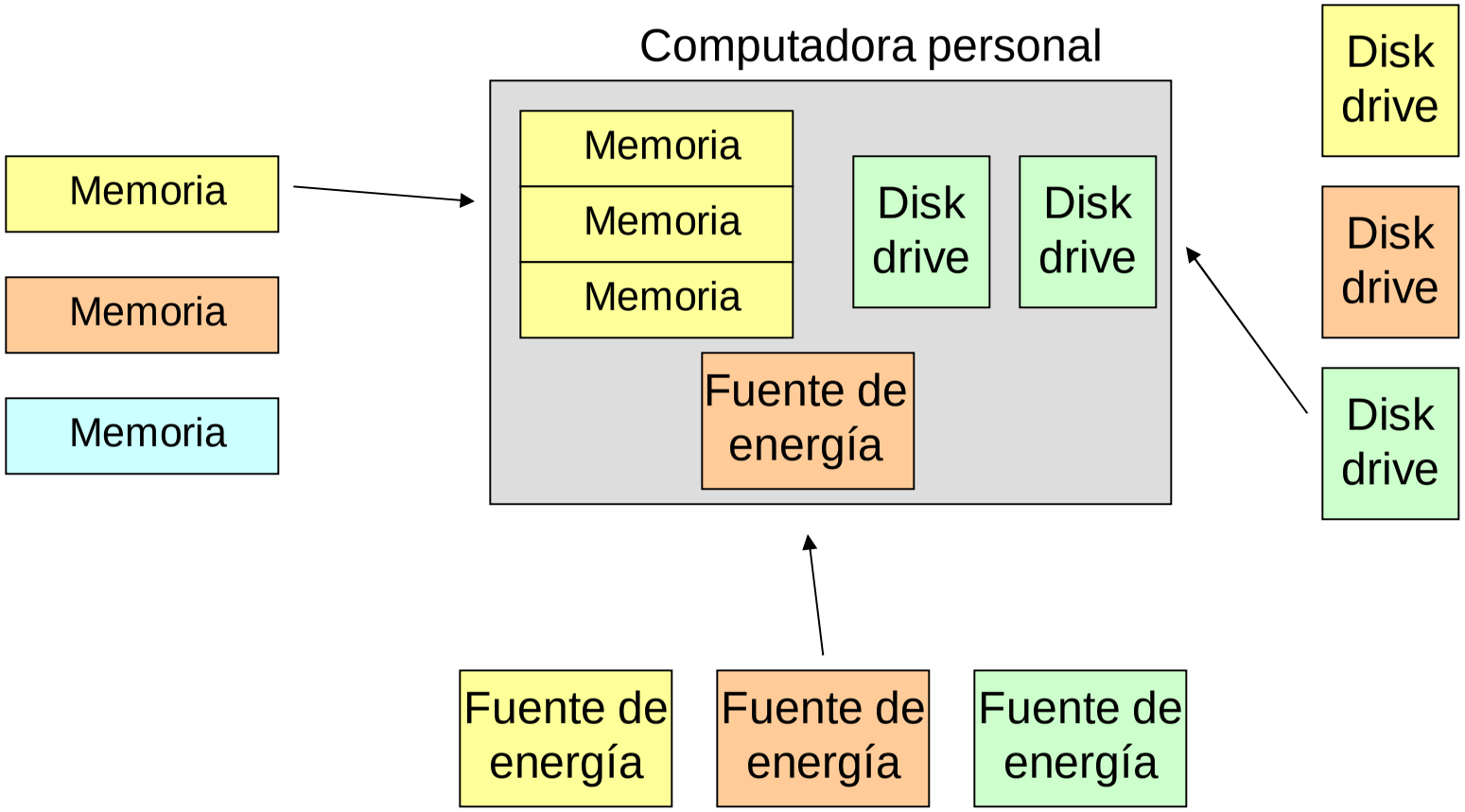
Ejemplo: Configuración de Producto

- Indices variables
- Filtración por la restricción **elemento**
- Relajación de la restricción **elemento**
- Relajación de una disjunción de sistemas lineales

Configuración de Producto

- Queremos configurar una computadora seleccionando el tipo de fuente de energía, el tipo de "disk drive" y el tipo de chips de memoria.
- También seleccionamos el número de "disk drives" y de chips de memoria.
  - Usar solamente 1 tipo de "disk drive" y 1 tipo de memoria.
- Restricciones:
  - Generar energía suficiente para los componentes.
  - El espacio de disco debe ser por lo menos 700.
  - La memoria debe ser por lo menos 850.
- Minimizar peso sujeto a las restricciones.

Configuración de Producto



Configuración de Producto

Component i	Type k	Net power generation A_{1jk}	Disk space A_{2jk}	Memory capacity A_{3jk}	Weight A_{4jk}	Max number used
1. Power supply	A	70	0	0	200	1
	B	100	0	0	250	
	C	150	0	0	350	
2. Disk drive	A	−30	500	0	140	3
	B	−50	800	0	300	
3. Memory chip	A	−20	0	250	20	3
	B	−25	0	300	25	
	C	−30	0	400	30	
Lower bound L_j		0	700	850	0	
Upper bound U_j		∞	∞	∞	∞	
Unit cost c_j		0	0	0	1	

Configuración de Producto

- Sea

  t_i = tipo del componente i instalado.

  q_i = cantidad del componente i instalado.

- Estas son las variables del problema.
- Este problema ilustrará la restricción **elemento**.

Configuración de Producto

Cantidad del atributo *j* producida (< 0 si es consumida): *memoria, energía, peso, etc.*

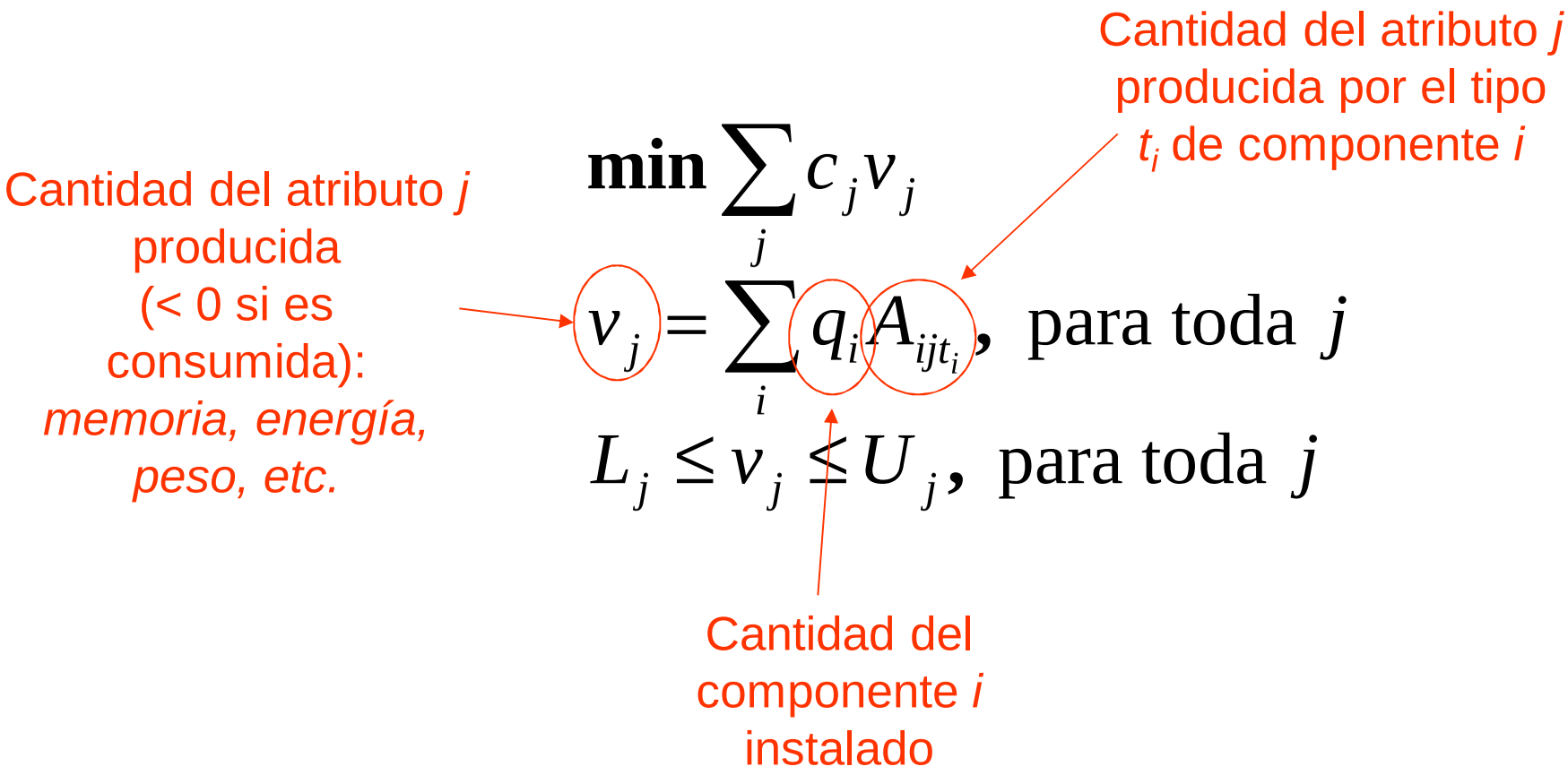
$$\min \sum_j c_j v_j$$

$$v_j = \sum_i q_i A_{ijt_i}, \text{ para toda } j$$

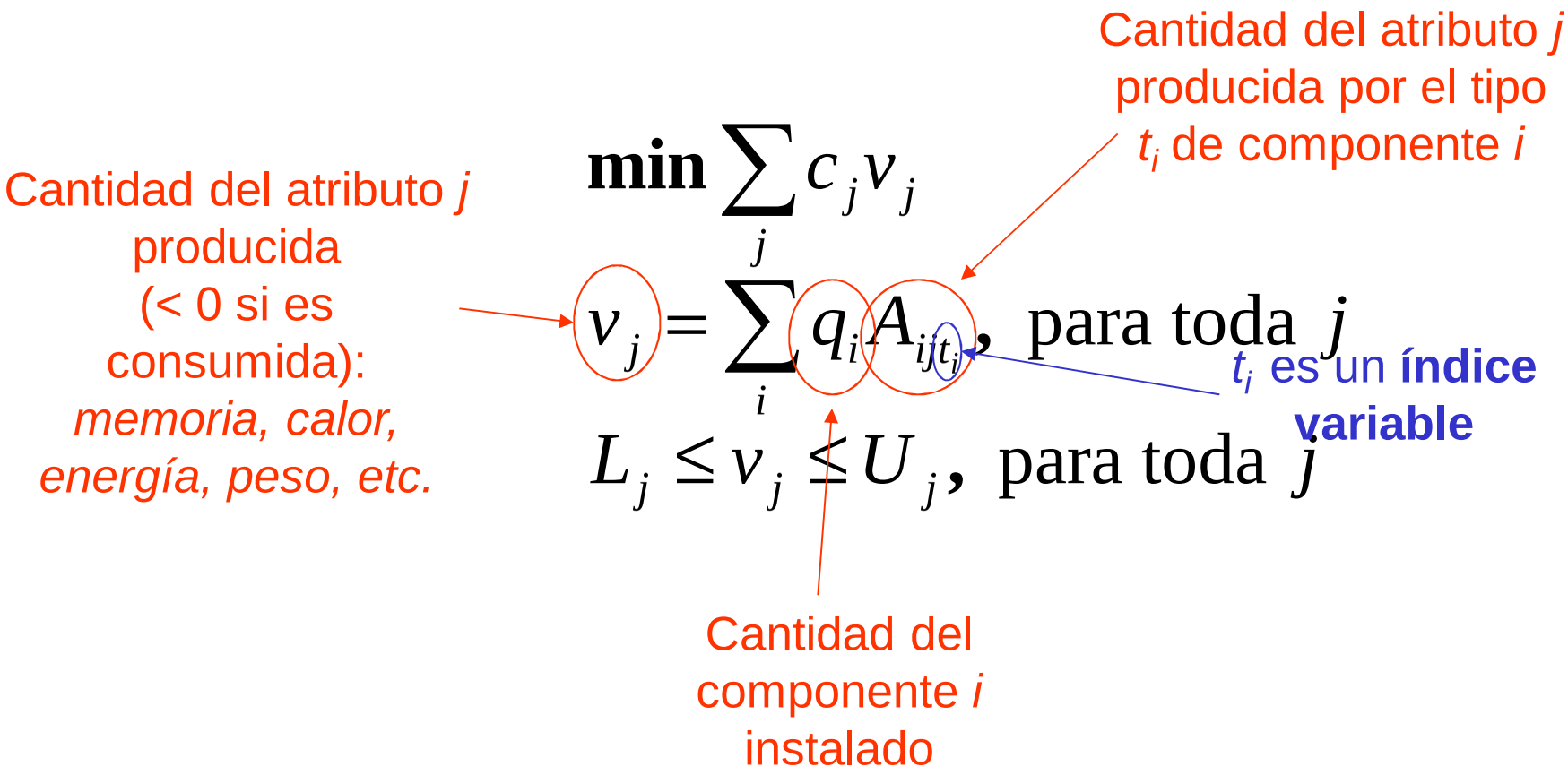
$$L_j \le v_j \le U_j, \text{ para toda } j$$

Cantidad del componente *i* instalado

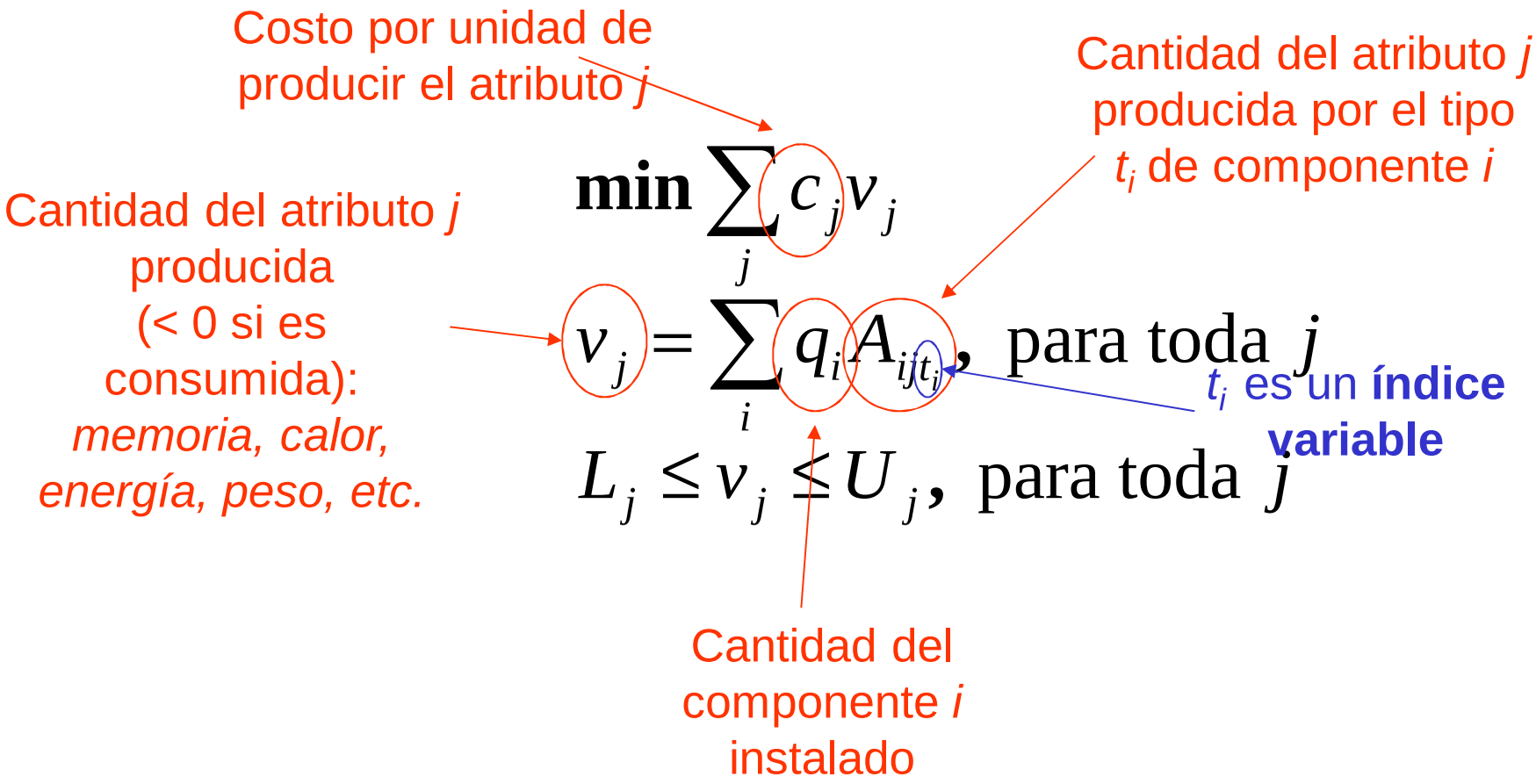
Configuración de Producto



Configuración de Producto



Configuración de Producto



Indices Variables

- Los índices variables son implementados con la restricción **elemento**.
- La y en x_y es un **índice variable**.
  - Los solucionadores de programación de restricciones implementan x_y remplazándola con una nueva variable z y agregando la restricción elemento(y,(x_1,…,x_n),z).
  - Esta restricción asigna z al y-ésimo elemento de la lista (x_1,…,x_n).
- Así, $\sum_i q_i A_{ijt_i}$ es remplazada por $\sum_i z_{ij}$ y la nueva restricción $\text{elemento}(t_i, (q_i A_{ij1}, \ldots, q_i A_{ijn}), z_{ij})$ para cada i.
- La restricción elemento puede ser propagada y relajada.

Filtración por Elemento

$$\text{elemento}(y, (x_1, \ldots, x_n), z)$$

puede ser procesada con un algoritmo de reducción de dominio que mantenga arco-consistencia.

Dominio de *z*

$$D_z \leftarrow D_z \cap \bigcup_{j \in D_y} D_{x_j}$$

$$D_y \leftarrow D_y \cap \{ j \mid D_x \cap D_{x_j} \neq \varnothing \}$$

$$D_{x_j} \leftarrow \begin{cases} D_z & \text{si } D_y = \{ j \} \\ D_{x_j} & \text{de lo contrario} \end{cases}$$

Filtración por Elemento

Ejemplo... $\text{elemento}(y,(x_1,x_2,x_3,x_4),z)$

Los dominios iniciales son:

$D_z = \{20,30,60,80,90\}$

$D_y = \{1,3,4\}$

$D_{x_1} = \{10,50\}$

$D_{x_2} = \{10,20\}$

$D_{x_3} = \{40,50,80,90\}$

$D_{x_4} = \{40,50,70\}$

Los dominios reducidos son:

$D_z = \{80,90\}$

$D_y = \{3\}$

$D_{x_1} = \{10,50\}$

$D_{x_2} = \{10,20\}$

$D_{x_3} = \{80,90\}$

$D_{x_4} = \{40,50,70\}$

Relajación de Elemento

- elemento(y,(x_1,…,x_n),z) puede ser una relajación continua.
- Esto implica la **disjunción** $\bigvee_k (z = x_k)$

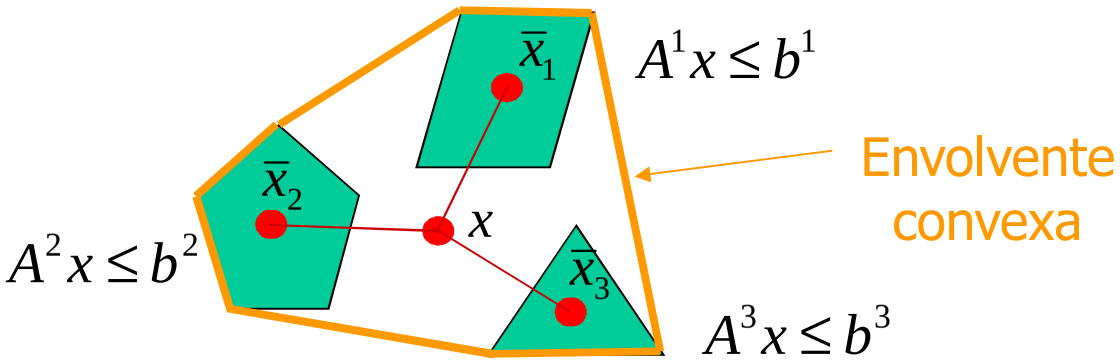
- En general, a una disjunción de sistemas lineales se le puede dar una relajación de **envolvente convexa.**
  - Esto proporciona una forma de relajar la restricción elemento.

Relajación de una Disjunción de Sistemas Lineales

- Consideremos la disjunción

$$\bigvee_k (A^k x \le b^k)$$

- Esta describe la unión de poliedros definida por $A^k x \le b^k$.
- Cada punto x en la envolvente convexa de esta unión es una combinación convexa de los puntos $\bar{x}^k$ en el poliedro.



Relajación de una Disjunción de Sistemas Lineales

- Así, tenemos $x = \sum_k \alpha_k \bar{x}^k$

$$A^k \bar{x}^k \le b^k, \text{ para toda } k$$

$$\sum_k \alpha_k = 1, \quad \alpha_k \ge 0$$

Usando el cambio de variable $x^k = \alpha_k \bar{x}^k$ obtenemos la relajación de envolvente convexa

$$x = \sum_k x^k$$

$$A^k x^k \le b^k \alpha_k, \text{ para toda } k$$

$$\sum_k \alpha_k = 1, \quad \alpha_k \ge 0$$

Relajación de Elemento

- La relajación de envolvente convexa de elemento(y,(x_1, …, x_n), z) es la relajación de envolvente convexa de la disjunción $\bigvee_k (z = x_k)$, que se reduce a

$$z = \sum_k x_{kk}$$

$$x_i = \sum_k x_{ik}, \text{ para toda } i$$

Configuración de Producto

- Volviendo al problema de configuración de producto, el modelo con la restricción de elemento es

$$\min \sum_j c_j v_j$$

$$v_j = \sum_i z_{ij}, \quad L_j \le v_j \le U_j, \quad \text{para toda } j$$

$$\text{elemento}\left(t_i, (A_{ij1}, \ldots, A_{ijm}), z_{ij}\right), \quad \text{para todas } i, j$$

con dominios

Tipo de fuente de energía — Tipo de disco — Tipo de memoria

$$t_1 \in \{A, B, C\}, \quad t_2 \in \{A, B\}, \quad t_3 \in \{A, B, C\}$$

$$v_j \in [L_j, U_j], \quad \text{all } j$$

Número de fuentes de energía — Número de discos, chips de memoria

$$q_1 \in \{1\}, \quad q_2, q_3 \in \{1,2,3\}$$

Configuración de Producto

- Podemos usar los cortes "knapsack" para filtrar los dominios.

Dado que $v_j \in [L_j, U_j]$, la restricción $v_j = \sum_j z_{ij}$ implica

$v_j^L \leq \sum_j z_{ij} \leq v_j^U$ donde cada z_{ij} toma uno de los valores $q_i A_{ij1}, \ldots, q_i A_{ijm}$.

- Lo cual implica las desigualdades "knapsack"

$$L_j \leq q_i \sum_i \max_k \{A_{ijk}\} \qquad q_i \sum_i \min_k \{A_{ijk}\} \leq U_j$$

- Estas desigualdades pueden ser usadas para reducir el dominio de q_i.

Configuración de Producto

- Usando los límites inferiores L_{ji} para energía, espacio de disco y memoria, obtenemos las desigualdades "knapsack"

$$0 \le q_1 \max\{70,100,150\} + q_2 \max\{-30,-50\} + q_3 \max\{-20,-25,-30\}$$
$$700 \le q_1 \max\{0,0,0\} + q_2 \max\{500,800\} + q_3 \max\{0,0,0\}$$
$$850 \le q_1 \max\{0,0,0\} + q_2 \max\{0,0\} + q_3 \max\{250,300,400\}$$

que se reducen a

$$0 \le 150q_1 - 30q_2 - 20q_3$$
$$700 \le 800q_2$$
$$850 \le 400q_3$$

- Propagándolas se reduce el dominio de q_3 de {1,2,3} a {3}.

Configuración de Producto

- La propagación de todas las restricciones reduce los dominios a

$q_1 \in \{1\}$	$q_2 \in \{1,2\}$	$q_3 \in \{3\}$	
$t_1 \in \{C\}$	$t_2 \in \{A, B\}$	$t_3 \in \{B, C\}$	
$z_{11} \in [150,150]$	$z_{21} \in [-75,-30]$	$z_{31} \in [-90,-75]$	
$z_{12} \in [0,0]$	$z_{22} \in [700,1600]$	$z_{32} \in [0,0]$	
$z_{13} \in [0,0]$	$z_{23} \in [0,0]$	$z_{33} \in [900,120]$	
$z_{14} \in [350,350]$	$z_{24} \in [140,600]$	$z_{34} \in [75,90]$	
$v_1 \in [0,45]$	$v_2 \in [700,1600]$	$v_3 \in [900,1200]$	$v_4 \in [565,1040]$

Configuración de Producto

- Usando la relajación de envolvente convexa de la restricción elemento obtenemos la relajación lineal del problema:

$$\min \sum_j c_j v_j$$

$$v_j = \sum_i \sum_{k \in D_{t_i}} A_{ijk} q_{ik}, \text{ para toda } j$$

$$q_i = \sum_{k \in D_{t_i}} q_{ik}, \text{ para toda } i$$

$$v_j^L \le v_j \le v_j^U, \text{ para toda } j$$

$$q_i^L \le q_i \le q_i^U, \text{ para toda } i$$

$$v_j^L \le \sum_i \max_{k \in D_{t_i}} \{A_{ijk} q_i\}, \text{ para toda } j$$

$$\sum_i \min_{k \in D_{t_i}} \{A_{ijk} q_i\} \le v_j^U, \text{ para toda } j$$

$$q_{ik} \ge 0, \text{ para todas } i, k$$

$q_{ik} > 0$ cuando el tipo k del componente i es seleccionado, y q_{ik} es la cantidad instalada

Cotas actuales de v_j

Cotas actuales de q_i

Configuración de Producto

- La solución a la relajación lineal en el nodo inicial es

$q_1 = q_{1C} = 1$ ← Usar 1 tipo C de fuente de energía (t_1 = C)

$q_2 = q_{2A} = 2$ ← Usar 2 tipos A de "disk drives" (t_2 = A)

$q_3 = q_{3B} = 3$ ← Usar 3 tipos B de chips de memoria (t_3 = B)

otros q_{ij}s $= 0$

$(v_1, \ldots, v_4) = (15, 1000, 900, 705)$ ← Mín. peso es 705.

- Ya que hay un solo $q_{ik} > 0$ para cada i, no hay necesidad de bifurcar en los t_is.
- Esta es una solución factible y por lo tanto óptima.
  - El problema es resuelto en el nodo inicial.

Ejemplo: Asignación de Máquinas

- "Edge finding"
- Descomposición de Benders y "nogoods"

Asignación de Máquinas

- Queremos asignar 5 trabajos a 2 máquinas.
  - Cada trabajo tiene tiempos de iniciación y terminación.
  - Los tiempos y costos de procesamiento varían entre las dos máquinas.
  - Queremos minimizar el costo total teniendo en cuenta las restricciones en tiempos.
- Primero estudiaremos la propagación para el problema de **1 máquina** ("**edge finding**").
- Luego resolveremos el problema de 2 máquinas usando la **descomposición de Benders** y la propagación en el problema de 1 máquina.

Asignación de Máquinas

Job j	Release time r_j	Dead-line d_j	Processing time p_{Aj}	Processing time p_{Bj}	Processing cost f_{Aj}	Processing cost f_{Bj}
1	0	10	2	2	20	30
2	0	8	3	2	20	30
3	2	7	2	2	20	30
4	2	5	4	3	30	40
5	4	7	3	2	20	30

Tiempo de procesamiento en la máquina A

Tiempo de procesamiento en la máquina B

Asignación de Máquinas

- Los trabajos deben ser realizados uno a la vez en cada máquina (**asignación disjunta**). Para esto usamos la restricción

$$\text{disjunctive}(t, p)$$

$t = (t_1, \ldots t_n)$ son tiempos de iniciación (variables)

$p = (p_{i1}, \ldots p_{in})$ son tiempos de procesamiento (constantes)

- El mejor algoritmo conocido para la restricción "disjunctive" es "**edge finding**".
  - "Edge finding" no satisface complementamente arco-consistencia o consistencia de cotas.

Asignación de Máquinas

- El problema puede ser escrito como

$$\min \sum_j f_{y_j j}$$

$$t_j + p_{y_j j} \le d_j, \text{ para toda } j$$

$$\text{disjunctive}\left((t_j \mid y_j = i), (p_{ij} \mid y_j = i)\right), \text{ para toda } i$$

Tiempo de iniciación del trabajo j

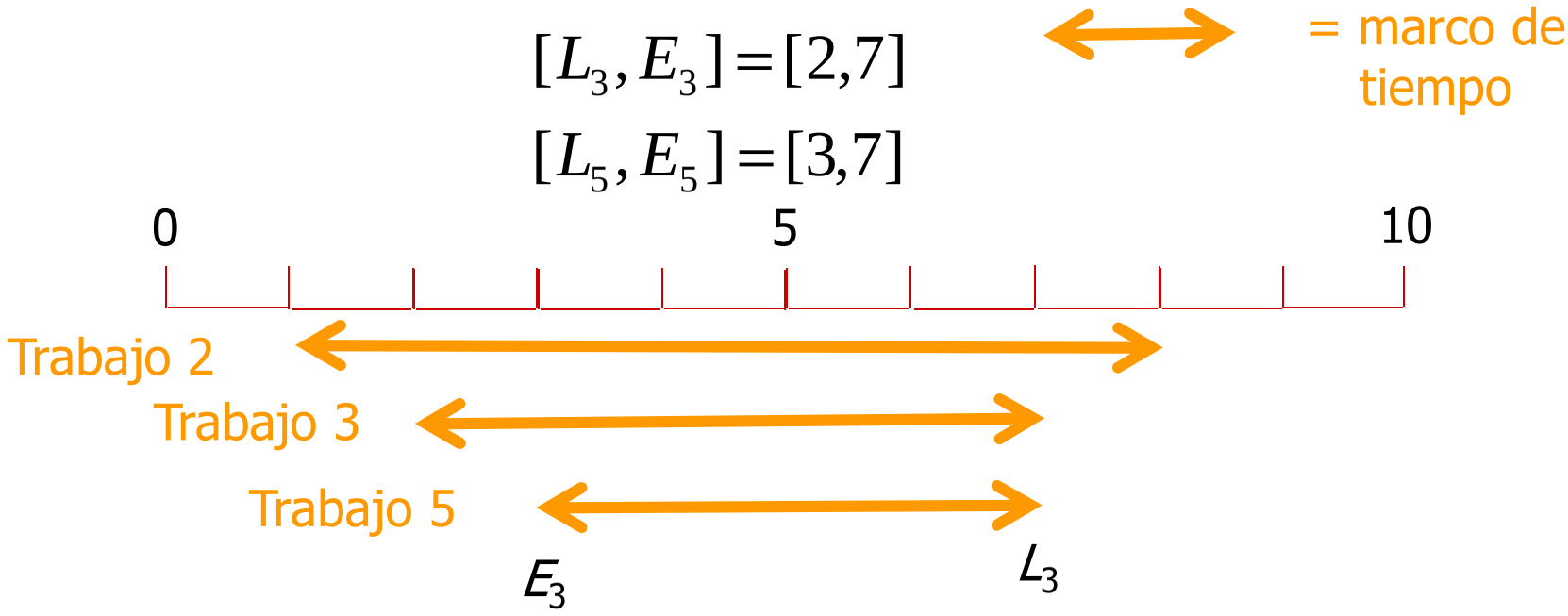
Máquina asignada al trabajo *j*

- Primero miraremos el problema de asignación de 1 máquina.

"Edge Finding"

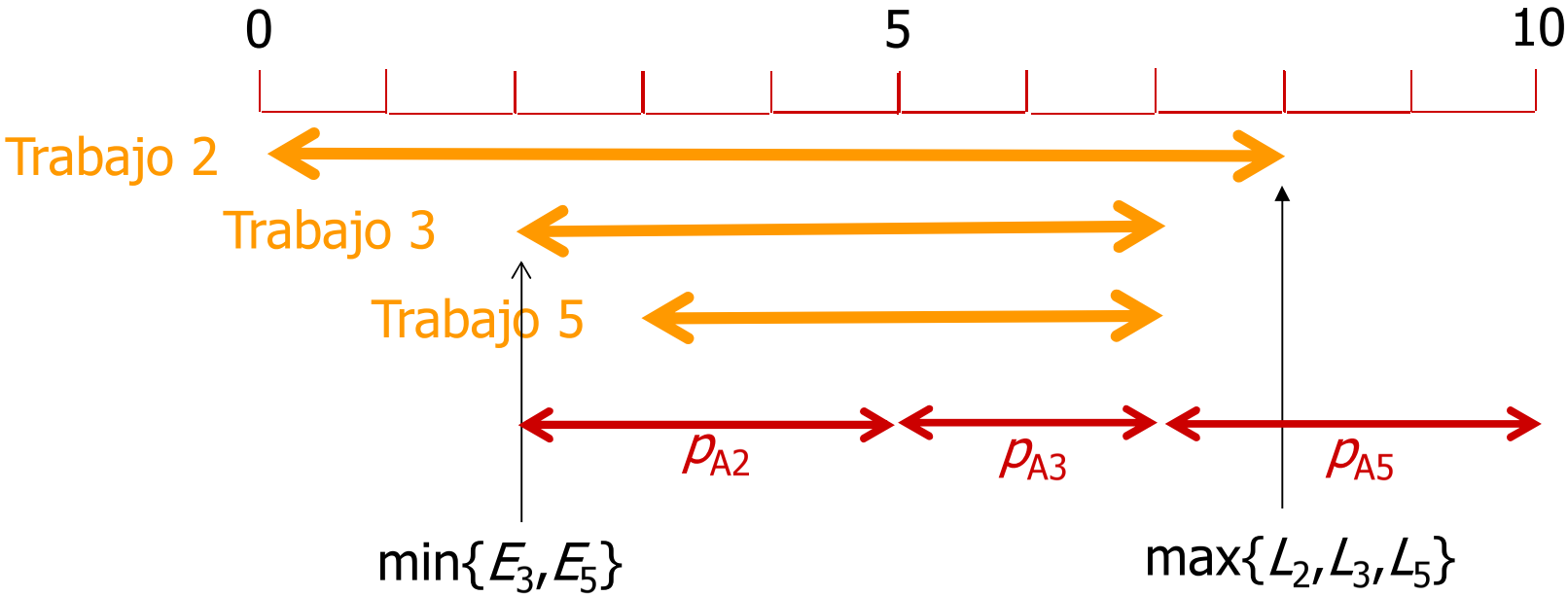
- Tratemos de asignar los trabajos 2, 3, 5 a la máquina A.
  - Inicialmente, el tiempo más temprano de iniciación E_j y el tiempo más tardío de terminación L_j son las fechas de iniciación r_j y de terminación d_j:

$$[L_2, E_2] = [0,8]$$
$$[L_3, E_3] = [2,7]$$
$$[L_5, E_5] = [3,7]$$



"Edge Finding"

- El trabajo 2 debe preceder los trabajos 3 y 5, porque:
  - Los trabajos 2, 3, 5 no se podrán realizar entre el tiempo más temprano de iniciación de 3, 5 y el tiempo más tardío de terminación de 2, 3, 5.
  - Por lo tanto, el trabajo 2 debe terminar antes de que los trabajos 3, 5 empiecen.

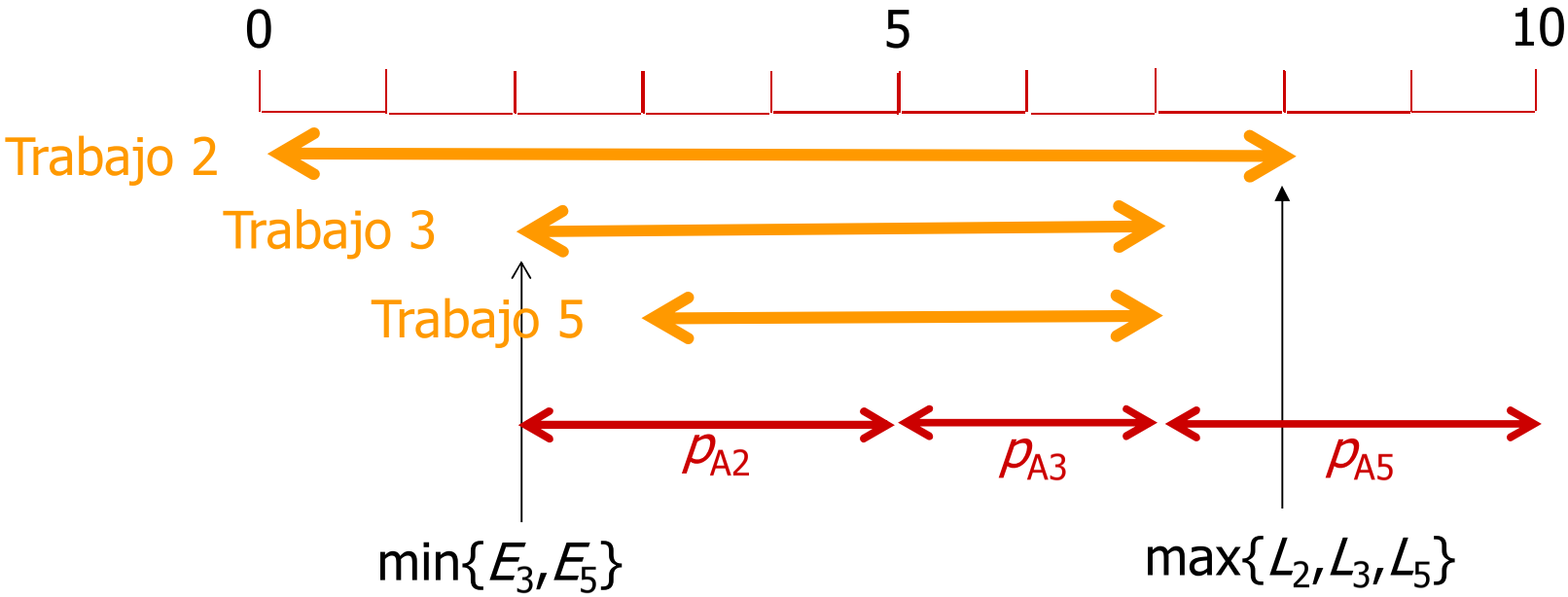


"Edge Finding"

- El trabajo 2 precede los trabajos 3, 5 porque:

$$\max\{L_2, L_3, L_5\} - \min\{E_3, E_5\} < p_{A2} + p_{A3} + p_{A5}$$

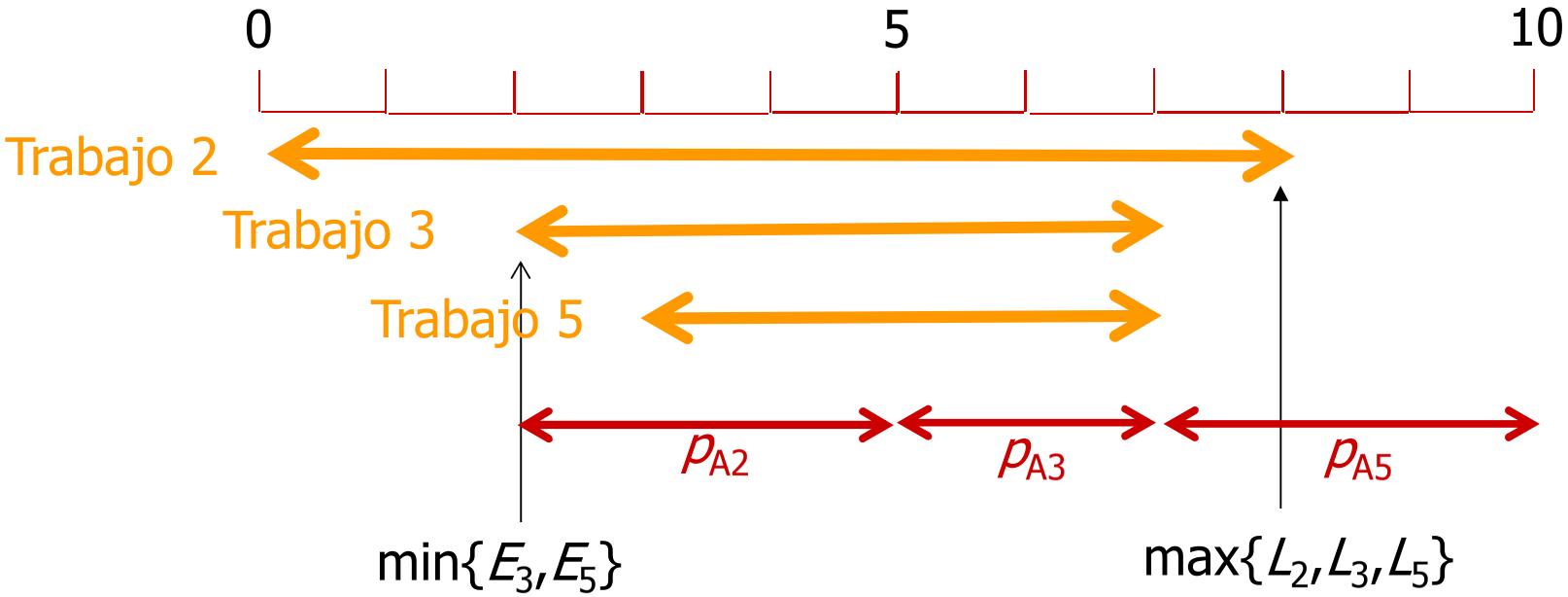
  - A esto se le llama "edge finding" porque encuentra un margen en el grafo de precedencia de los trabajos.



"Edge Finding"

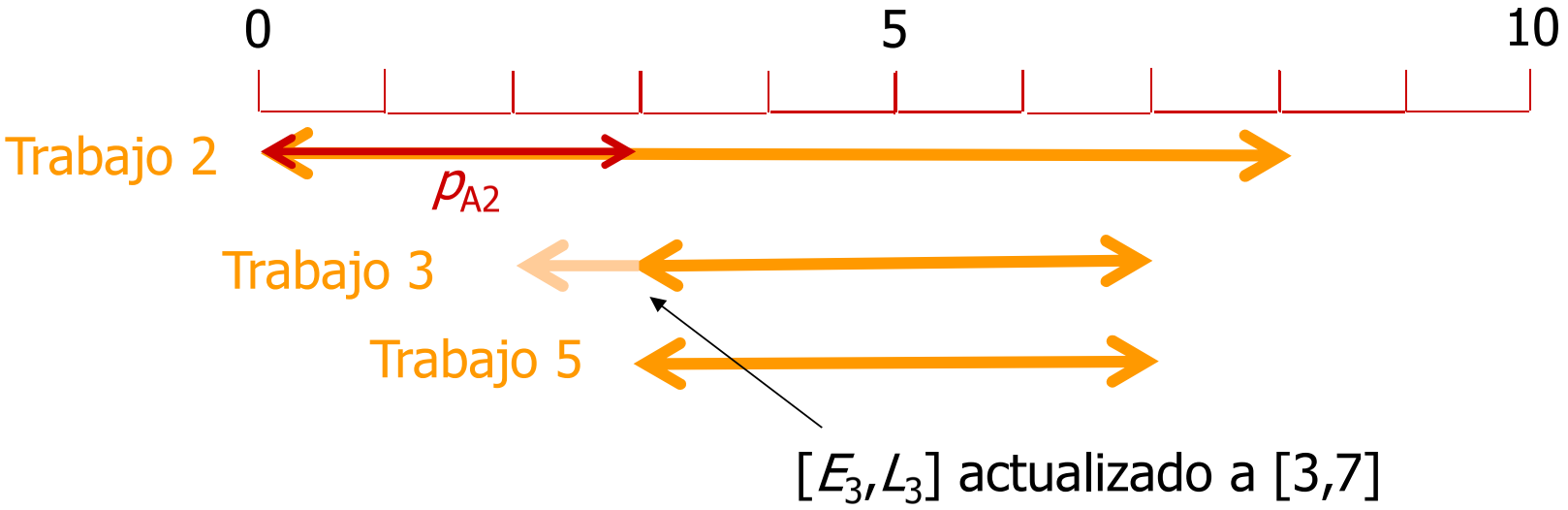
- En general, el trabajo *k* debe preceder el conjunto *S* de trabajos en la máquina *i* cuando

$$\max_{j \in S \cup \{k\}} \{L_j\} - \min_{j \in S} \{E_j\} < \sum_{j \in S \cup \{k\}} p_{ij}$$



"Edge Finding"

- Dado que el trabajo 2 debe preceder al 3 y al 4, los tiempos más tempranos de iniciación de los trabajos 3 y 4 pueden ser actualizados.
  - Ellos no pueden empezar antes de que el trabajo 2 termine.

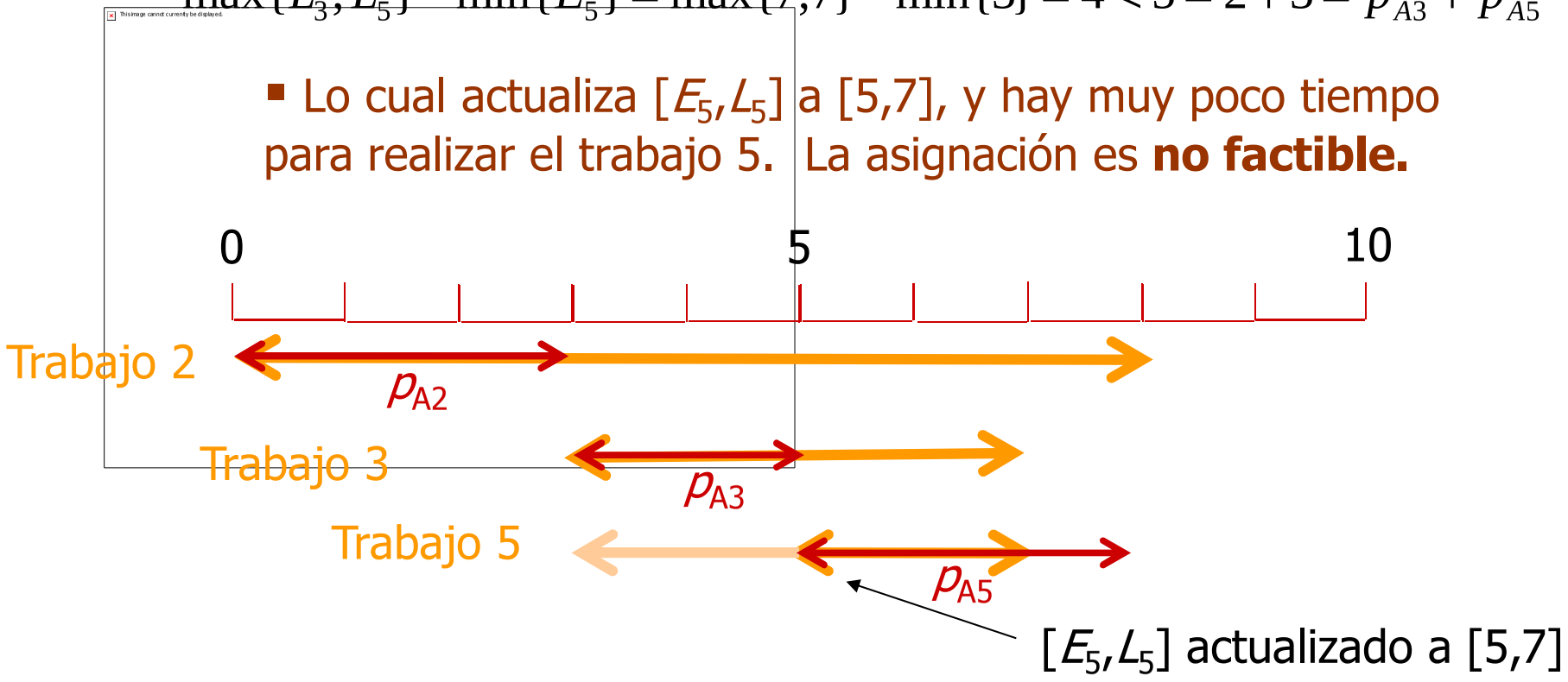


"Edge Finding"

- "Edge finding" también determina que el trabajo 3 debe preceder el trabajo 5, ya que

$$\max\{L_3, L_5\} - \min\{E_5\} = \max\{7,7\} - \min\{3\} = 4 < 5 = 2 + 3 = p_{A3} + p_{A5}$$

- Lo cual actualiza $[E_5, L_5]$ a [5,7], y hay muy poco tiempo para realizar el trabajo 5. La asignación es **no factible.**



Descomposición de Benders

- Resolveremos el problema de asignación de 2 máquinas con la **descomposición de Benders basada en lógica**.
- Primero resolveremos el problema de asignación que distribuye trabajos entre las máquinas (**problema maestro**).
- Dada esta distribución, asignaremos los trabajos distribuidos a cada máquina (**subproblema**).
- Si una máquina tiene una asignación no factible:
  - Determinar que trabajos causan la no factibilidad.
  - Generar un "**nogood**" (corte de **Benders**) que excluya la asignación de estos trabajos a esa máquina.
  - Agregar los cortes de Benders al problema maestro y solucionarlo nuevamente.
- Repetir hasta que el subproblema sea factible.

Descomposición de Benders

- El **problema maestro** (asignación de trabajos a máquinas) es:

$$\min \sum_j f_{y_j j}$$

Cortes de Benders

- Lo resolveremos como un **problema de programación entera**.
  - Sea la variable binaria $x_{ij} = 1$ cuando $y_j = i$.

$$\min \sum_{ij} f_{ij} x_{ij}$$

Cortes de Benders

$$x_{ij} \in \{0,1\}$$

Descomposición de Benders

- Agregaremos una **relajación del subproblema** al problema maestro para agilizar la búsqueda de la solución.

$$\min \sum_{ij} f_{ij} x_{ij}$$

$$p_{A3}x_{A3} + p_{A4}x_{A4} + p_{A5}x_{A5} \leq 5$$

$$p_{A2}x_{A2} + p_{A3}x_{A3} + p_{A4}x_{A4} + p_{A5}x_{A5} \leq 8$$

$$p_{A1}x_{A1} + p_{A2}x_{A2} + p_{A3}x_{A3} + p_{A4}x_{A4} + p_{A5}x_{A5} \leq 10$$

$$p_{A4}x_{A5} \leq 3$$

$$p_{A3}x_{A3} + p_{A4}x_{A4} + p_{A5}x_{A5} \leq 5$$

$$p_{B2}x_{B2} + p_{B3}x_{B3} + p_{B4}x_{B4} + p_{B5}x_{B5} \leq 8$$

Si los trabajos 3,4,5 son asignados a la máquina A, sus tiempos de procesamiento en esa máquina deben estar entre sus tiempos de iniciación más tempranos (2) y de terminación más tardíos (7).

Descomposición de Benders

- La solución del problema maestro es

$$x_{A1} = x_{A2} = x_{A3} = x_{A5} = x_{B4} = 1$$

$$\text{otras } x_{ij}\text{s} = 0$$

- Distribuir los trabajos 1,2,3,5 a la máquina A, y el trabajo 4 a la máquina B.
- El subproblema es asignar estos trabajos en las 2 máquinas.
  - Para la máquina B es trivial la asignación (solo 1 trabajo).
  - Encontramos que los trabajos 2, 3, 5 no pueden ser asignados en la máquina A.
  - Entonces los trabajos 1, 2, 3, 5 no pueden ser asignados a la máquina A.

Descomposición de Benders

- Entonces creamos un **corte de Benders** ("nogood") que excluya la asignación de los trabajos 2, 3, 5 de la máquina A.
  - Estos son los trabajos que causan la no factibilidad.
  - El corte de Benders es:

$$(1-x_{A2})+(1-x_{A3})+(1-x_{A5})\geq 1$$

- Agregar esta restricción al problema maestro.
  - La solución del problema maestro ahora es:

$$x_{A1}=x_{A2}=x_{A5}=x_{B3}=x_{B4}=1$$
$$\text{otras } x_{ij}\text{s}=0$$

Descomposición de Benders

- Entonces asignamos los trabajos 1, 2, 5 a la máquina A, y los trabajos 3, 4 a la máquina B.
  - Estas asignaciones son factibles.
  - La solución obtenida es óptima con costo mínimo de 130.