Programación lineal entera (PLE)

Qué es un problema de programación lineal entera?:

$$\text{Max } \mathbf{c}\,\mathbf{x}$$

sujeto a

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}$$

$$\mathbf{x} \in Z_+$$

Qué es un problema de programación lineal entera mixta (PLEM)?

Algunas variables son continuas y otras tienen que tomar valores enteros.

Qué es un problema de programación lineal binario (PLB)?

Las variables toman valor 0 o 1, o sea $\mathbf{x} \in \{0,1\}^n$.

Qué es un problema de optimización combinatoria?

Dado un conjunto finito N = {1,….n} con pesos c_j asignados a cada $j \in N$ y *F* una familia de conjuntos factibles de N, entonces queremos encontrar el elemento de F de peso mínimo:

$$\text{Min}_S \{ \sum_{j \in S} c_j / S \in F \}$$

(también podría ser un problema de maximización).

Muchos problemas de optimización combinatoria se pueden formular como problemas de programación lineal entera.

Cómo resolvemos un problema de programación lineal entera?.

Se parecen a los problemas de programación lineal?. Sirve redondear?.

Ejemplo:

$$\text{Max } 1.00\, x_1 + 0.64\, x_2$$

Sujeto a

$$50\, x_1 + 31\, x_2 \leq 250$$

$$3\, x_1 - 2x_2 \geq -4$$

$$x_1 , x_2 \geq 0,\ x_1 , x_2 \text{ entera}$$

Se puede ver que la solución entera es (5,0), si no pedimos que las variables sean enteras la solución del problema de programación lineal es (376/193, 950/193). Graficar.

En el caso de programación binaria la situación puede ser peor, por ejemplo una solución del problema continuo PL que sea (0.5,0.5,….0.5) no nos da información sobre cual puede ser la solución del problema binario.

Ejemplos: *escribir modelos de PLE para los siguientes casos:*

- **Problema de asignación:** Tenemos n personas para realizar n trabajos. Cada persona hace un sólo trabajo. Se asigna un costo c_{ij} a la asignación de la persona i para hacer el trabajo j de acuerdo a la capacidad de cada uno para realizar cada trabajo. Se quiere encontrar la asignación que minimice los costos.

- **Problema de la mochila (Knapsack problem)** Se tiene un presupuesto de b pesos disponible para invertir en algunos de n posibles proyectos para el próximo año. Sea a_j la inversión que requiere el proyecto j y c_j el beneficio que produce dicho proyecto. Se quiere maximizar las ganancias (sólo se puede invertir en el proyecto completo).

Problema de cubrimiento (Set Covering): se quieren cubrir un cierto número de regiones con centros de emergencia (por ejemplo estaciones de bomberos o de ambulancias) con un costo mínimo. Se conocen los costos de instalar un centro en cada región y que regiones pueden ser servidas desde el mismo.

Problema del viajante de comercio (Traveling Salesman Problem, TSP): (es el problema más famoso de Optimización Combinatoria y el más estudiado.) Un viajante de comercio quiere visitar n ciudades y volver al lugar de partida en el menor tiempo posible (o al menor costo).

Los problemas que acabamos de ver son problemas de optimización combinatoria, donde hay que buscar el subconjunto óptimo entre una familia de subconjuntos. Uno podría pensar en enumerar todos estos conjuntos y elegir el mejor (algoritmo de fuerza bruta).

Sirve este procedimiento?.

Por ejemplo en el caso del viajante de comercio, si suponemos que podemos ir de cualquier ciudad a cualquier otra, tendríamos que revisar (n-1)! recorridos posibles. Por ejemplo para un problema con n= 101 ciudades hay 9.33×10^{157} circuitos (recorridos) posibles que habría que revisar.

- Vamos a presentar algoritmos mejores que estos que permiten resolver problemas bastante grandes.

- Sin embargo muchos problemas de optimización combinatoria como por ejemplo el TSP se consideran problemas abiertos o no resueltos desde el punto de vista de la investigación. A pesar de que se resuelven cada vez problemas mayores, no conocemos métodos buenos que resuelvan estos problemas en un tiempo razonable, o sea sólo conocemos algoritmos que requieren un número de operaciones que crece en forma exponencial cuando aumenta el tamaño del problema.

- No se conocen tampoco métodos que resuelvan cualquier problema de programación lineal entera en un tiempo que crezca en forma polinomial respecto del tamaño del problema.

Otros ejemplos:

Problema de localización (UFL):

dadas n posibles ubicaciones para poner depósitos y un conjunto de m clientes, supongamos que hay un costo fijo f_j de instalar un depósito en el lugar j, y un costo c_{ij} de llevar la mercadería que requiere el cliente i desde el depósito j Se quiere instalar los depósitos de forma a minimizar los costos totales.

Este problema se puede modelar como un problema de programación entera mixta si consideramos que las variables que definen que fracción de la demanda del cliente i es llevada al depósito j, no necesitan ser enteras.

Problema de planificación de la producción (ULS):

Se quiere hacer un plan de producción de un sólo producto, para un horizonte de n periodos, con el objetivo de satisfacer la demanda en cada período minimizando costos.

Tenemos los siguientes datos:

f_t costo fijo de producir en el período t

p_t costo por unidad producida en el período t

h_t costo de almacenamiento por unidad en el periodo t

d_t demanda del periodo t.

Este problema se puede modelar como un problema de programación entera mixta si consideramos que las variables que definen cuanto producir no necesitan ser enteras.

Representación de alternativas o disyunciones:

Se puede modelar esta situación con variables binarias.

Def: Un poliedro $P \subset R^{n+p}$ es una formulación para un conjunto $X \subset Z^n \times R^p$ si $X = P \cap Z^n \times R^p$.

Puede haber varias formulaciones para un mismo problema.

Ejemplo: Dibujar dos formulaciones diferentes para el siguiente conjunto:

$X = \{ (1,1),(2,1), (3,1), (1,2),(2,2),(3,2),(2,3)\}$

Ejemplo: Sea X = {(0,0,0,0},(1,0,0,0),(0,1,0,0),(0,0,1,0),
(0,0,0,1),(0,1,0,1),(0,0,1,1)}

Las siguientes son 3 formulaciones diferentes para X

$P1 = \{ x \in R^4, 0 \le x \le 1, 83\,x_1 + 61\,x_2 + 49\,x_3 + 20\,x_4 \le 100\}$

$P2 = \{ x \in R^4, 0 \le x \le 1, 4\,x_1 + 3\,x_2 + 2\,x_3 + x_4 \le 4\}$

$P3 = \{ x \in R^4, 0 \le x \le 1,$

$$\begin{aligned} 4\,x_1 + 3\,x_2 + 2\,x_3 + x_4 &\le 4 \\ x_1 + x_2 + x_3 &\le 1 \\ x_1 + x_4 &\le 1\} \end{aligned}$$

Se puede probar fácilmente a partir de estas ecuaciones que
$P1 \supset P2 \supset P3$ y $P3 = conv(X)$

Formulaciones alternativas para los problemas de localización (UFL) y planificación de la producción (ULS).

Dado $X \subset R^n$, el problema

$$IP : \{\max cx / x \in X\}$$

es equivalente al problema de programación lineal

$$\{\max cx / x \in \text{conv}(X)\}$$

Como ya vimos conv(X) es un poliedro y sus puntos extremos están en X.

- *Cuándo decimos entonces que una formulación es mejor que otra?.*
- *Cuál sería la formulación ideal para un problema de programación lineal entera?*

- Si la formulación es una descripción de la cápsula convexa del conjunto de soluciones enteras entonces podemos resolver el problema como un problema de programación lineal (continua, por ejemplo con el método simplex). Todos los vértices del poliedro serán enteros y por lo tanto el óptimo de PL será entero.

- En la mayoría de los casos es muy difícil encontrar la cápsula convexa del conjunto de soluciones enteras, y además muchas veces la descripción de la cápsula convexa requiere de un número exponencial de desigualdades.

Cotas y relajaciones

Sea el problema **IP** de PLE

$$z^* = \max \{ cx / x \in X \subset Z^n \}$$

Dada una solución factible x, cómo podemos probar que es óptima?.

- Querríamos, por ejemplo, encontrar una cota inferior $\mathbf{z\_}$ de z y una cota superior $\mathbf{z^{--}}$ tal que $\mathbf{z\_ = z^{--}}$.

- O sea nos serviría por ejemplo tener un algoritmo que vaya generando una sucesión creciente de valores de $\mathbf{z\_}$ y una sucesión decreciente de valores $\mathbf{z^{--}}$ y esto es lo que tratan de hacer los métodos que se usan para resolver problemas de IP.

Cómo hacer para obtener cotas inferiores (cotas primales) de z?.*

- Cualquier solución factible, $x \in X$, nos provee de una cota inferior $cx \leq z^*$.

- A veces obtener una solución factible es tan difícil como resolver el problema de optimización.

- Para obtener soluciones factibles en la práctica se usan frecuentemente algoritmos heurísticos. Estos algoritmos tratan de obtener las mejores soluciones posibles pero no garantizan obtener el óptimo.

Cómo hacer para obtener cotas superiores (cotas duales) de z?*

Relajación: reemplazar el problema **IP** por uno más simple que sabemos resolver, por ejemplo:

i) "agrandar" el conjunto de soluciones factibles,

ii) o reemplazar la función objetivo por una cuyo valor sea mayor o igual a cx en todas las soluciones factibles.

Prop: si **RP** es una relajación de **IP** y $\mathbf{z^R}$ es el valor óptimo de **RP** entonces $\mathbf{z^R \geq z^*}$

Relajación lineal:

Si PLE es

$$\mathbf{z^*} = \max \{ cx / Ax \leq b, x \in Z_+^n \}$$

la relajación lineal RLP de PLE se define como:

$$\mathbf{z^{LP}} = \max \{ cx / Ax \leq b, x \in R_+^n \}$$

Queremos tener la mejor formulación del PLE posible.

Prop:

i) si RLP es no factible, entonces PLE es no factible.

ii) Si x^* es óptimo para RLP y $x^* \in X$ entonces x^* es la solución óptima de PLE.

Estas ideas se pueden aplicar también a otras relajaciones diferentes de PLE.

Ejemplo: la relajación lineal del siguiente problema

$$\max\ 7\,x1 + 4\,x2 + 5\,x3 + 2\,x4$$

sujeto a

$$3\,x1 + 3\,x2 + 4\,x3 + 2\,x4 \leq 6$$

$$x \in \{0,1\}^4$$

tiene solución $x^* = (1,1,0,0)$

Qué conclusión podemos sacar?.

Relajaciones combinatorias:

Problemas que podemos resolver más fácilmente que PLE, se obtienen por ejemplo eliminando algunas de las restricciones de PLE.

- Por ejemplo si tenemos el problema del viajante de comercio TSP asimétrico, el problema de asignación es una relajación del TSP.

- Si tenemos el problema de la mochila, tenemos

$$X = \{x \in X \subset Z_+^n \,/\, \sum_j a_j x_j \leq b\}$$

Obtenemos una relajación reemplazando X por

$$X' = \{x \in X \subset Z_+^n \,/\, \sum_j \lfloor a_j \rfloor x_j \leq \lfloor b \rfloor\}$$

Relajación Lagrangeana: en este caso en vez de simplemente ignorar las restricciones “difíciles”, las incluímos en la función objetivo.

Prop: Si tenemos $z^* = \max \{ cx / Ax \leq b, x \in X \subset Z_+^n \}$ y ponemos

$$z(u) = \max \{ cx + u (b - Ax) , x \in X \subset Z_+^n \}$$

entonces, para todo $u \geq 0$, $z(u) \geq z^*$.

Dualidad:

En el caso de programación lineal entera NO se verifica el mismo resultado de dualidad que vimos en PL. O sea no es cierto que si formulamos el problema dual en forma similar, el óptimo del primal y el dual sean iguales.

- Se dice en este caso que se cumple una condición de dualidad débil cuando el óptimo del "primal" es menor o igual al del dual, *(problema de maximización)* y de dualidad fuerte cuando como en el caso de PL los óptimos del primal y el dual son iguales.

Por ejemplo si tenemos el PLE

$$z^* = \max \{ cx / Ax \leq b, x \in Z_+^n \}$$

podemos decir que el problema

$$w^{LP} = \min \{ yb / yA \geq c, y \in R_+^n \}$$

es un dual débil porque

$$cx \leq yb$$

para todo par de soluciones x,y, y por lo tanto también

$$z^* \leq w^{LP}$$

Prop: si IP: max { c(x) / $x \in X$ } y

D= min { w(u) / $u \in U$ }

son un par de problemas duales débiles entonces:

i) Si D es no acotado IP es no factible

ii) Si $x^* \in X$ y $u^* \in U$ cumplen $c(x^*) = w(u^*)$ entonces x^* y u^* son óptimas para IP y D respectivamente.

Ejemplo: el problema de determinar un matching máximo en un grafo y el problema de determinar el recubrimiento mínimo de ejes por nodos son duales débiles.

Se puede probar usando teoría de grafos o también formulándolos como problemas de programación lineal entera, usando como matriz A la matriz de incidencia del grafo.

El problema de matching máximo quedaría:

$$z = \max \{\mathbf{1}x / Ax \leq 1, x \in Z_+^n\}$$

Y el de determinar un recubrimiento mínimo de los ejes por nodos quedaría:

$$z = \min \{\mathbf{1}x / A^t x \geq 1, x \in Z_+^n\}$$

Se puede ver con un ejemplo que estos dos problemas no son duales fuertes.

Problemas “fáciles” o “bien resueltos”

En la práctica decimos a veces que un problema de programación programación lineal entera es “fácil”, si desde el punto de vista de las aplicaciones nos sirven los resultados que obtenemos redondeando.

Ejemplo: cutting stock *(ejemplo además de un método de generación de columnas, se dio en clase, ver libro de Chvatal)*

Pero si hablamos de soluciones exactas del problema tenemos que tener definiciones más precisas y formales……..

Problemas “fáciles” o “bien resueltos”

Cuándo podemos decir que un problema de programación lineal entera es “fácil”?. (en este caso queremos resolverlo en forma exacta)

i) Hay un algoritmo polinomial para resolverlo

ii) Hay un dual fuerte que permite definir condiciones de optimalidad fácilmente verificables.

iii) Hay un algoritmo polinomial para resolver el problema de “separación” .

iv) Tenemos una descripción completa y compacta de la cápsula convexa del problema.

PLE con matrices totalmente unimodulares

Sea $z = \max \{ cx / Ax \leq b, x \in Z_+^n \}$ con A y b con todos sus elementos enteros.

Prop: Si para la base óptima B se verifica que det(B) = +/-1 entonces la solución óptima básica correspondiente de la relajación lineal RLP del PLE es entera, y por lo tanto el óptimo de RLP es igual a el óptimo z* del PLE.

Dem: usando la regla de Cramer para resolver sistemas lineales para calcular el óptimo x_B^* en el simplex revisado, se ve que si los elementos de B y b son enteros y det (B) = +/-1, x_B^* es entero.

Def: Decimos que una matriz A es **totalmente unimodular (TU)** si el determinante de A y de todas sus submatrices cuadradas es 1, -1, ó 0.

Prop: Si A es totalmente unimodular entonces todas las soluciones básicas de la relajación lineal RLP del PLE son enteras, y por lo tanto el óptimo de RLP es igual a el óptimo z* del PLE.

Cuál es la importancia de este resultado?.

Si la matriz del PLE es totalmente unimodular entonces podemos obtener las solución resolviendo su relajación lineal, por ejemplo usando el método Simplex.

Cómo reconocer una matriz totalmente unimodular?

Prop: Son equivalentes:

i) A es TU

ii) A^T es TU

iii) La matriz (A,I) es TU

Prop: Una matriz es TU si se verifica que:

i) $a_{ij} \in \{1,-1,0\}$ para todo i,j

ii) Cada columna contiene a lo sumo dos coeficientes $\neq 0$.

iii) Existe una partición (M_1,M_2) del conjunto de filas M tal que para toda columna j que tiene dos elementos distintos de o, se satisface que $\sum_{i \in M1} a_{ij} - \sum_{i \in M2} a_{ij} = 0$

Prop: El problema de PL max $\{cx / Ax \leq b, x \in R_+^n\}$ tiene solución entera para todo vector b para la cuál existe solución finita si y sólo si A es TU.

Si A es TU entonces :

i) vale la propiedad de dualidad fuerte

ii) Se conoce la cápsula convexa.

iii) El problema de separación se puede resolver eficientemente.

Ejemplos de problemas “fáciles”:

- Problema de asignación. *(la matriz del problema es TU).*
- Problema de encontrar el flujo máximo en una red. *(idem anterior)*
- Problema de transporte *(el conocido método para resolver este problema es una adaptación del método simplex).*

Métodos exactos para resolver problemas de PLE

- Branch and bound (“ramificación y poda”)
- Métodos de cortes o de planos de corte
- Métodos “Branch and Cut”: *(combinan las ideas de los dos anteriores y son los que han permitido resolver los problemas de programación lineal entera u optimización combinatoria más grandes).*
- Métodos de generación de columnas *(se usan cuando tenemos un número muy grande de variables, que nos impide tener guardadas en la memoria de la computadora todas las columnas de A).*

Para obtener el óptimo estos métodos pueden requerir un número muy grande de iteraciones, (el número de operaciones crece exponencialmente con el tamaño del problema a resolver).

Branch and bound

Tenemos el PLE $z = \max \{ cx / x \in X \subset Z^n \}$

Queremos dividir el problema en problemas más pequeños que sean más fáciles de resolver.

O sea poner $X = X_1 \cup \ldots\ldots \cup X_h$, entonces si

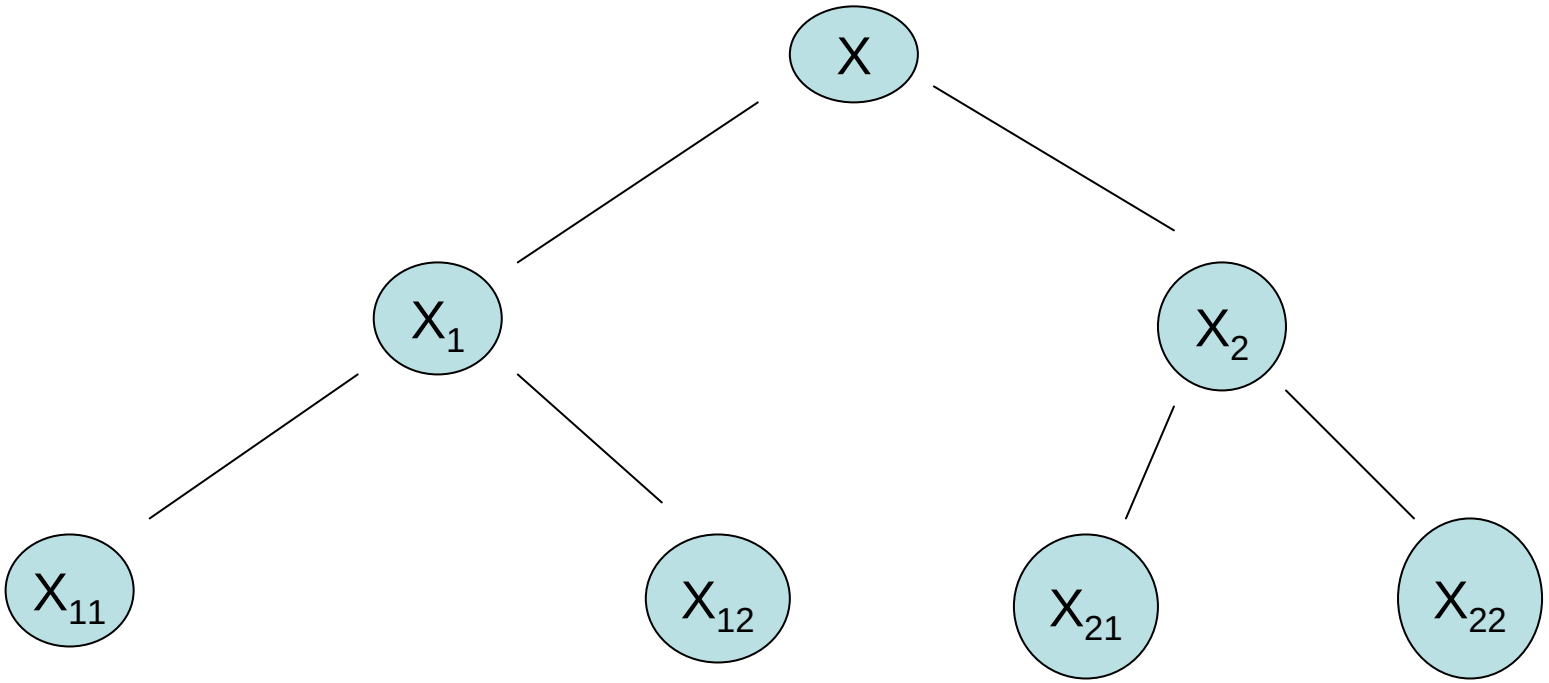
$$z^k = \max \{ cx / x \in X_k \}$$

tendremos que $z = \max_k \{z^k\}$

Para construir un método de branch and bound usamos estas ideas en forma iterativa.

Usamos un árbol de enumeración para representar el algoritmo branch and bound. En cada nodo resolvemos un problema menor de optimización.

Ejemplo:



Acá $X = X_1 \cup X_2$, $X_1 = X_{11} \cup X_{12}$, etc.

Ejemplo: árbol de enumeración para $S \subset \{0,1\}^3$

Ejemplo: árbol de enumeración para las soluciones del TSP. Ejemplo para 4 ciudades.

Enumeración implícita

Cómo hacemos para no tener que recorrer el árbol completo?.

Prop.: Si tenemos una descomposición del conjunto de soluciones factibles $X = = X_1 \cup \ldots\ldots \cup X_h$, y $z^k = \max \{ cx / x \in X_k \}$, y si $z^{k\_}$ es una cota superior de z^k y $z^k\_$ es una cota inferior de z^k, entonces $z^{\_} = \max_k z^{k\_}$ es una cota superior de z^* y $z\_ = \max_k z^k\_$ es una cota inferior de z^*.

En la mayoría de los casos se resuelve en cada nodo del árbol la relajación lineal del problema z^k, usando el método simplex.

Criterios de poda:

- por optimalidad
- por valor de las cotas
- por infactibilidad.

Cómo se pueden obtener estas cotas?.

Otras decisiones a tomar para implementar un B&B:

- *Cómo dividir el conjunto de soluciones?. En cuantas partes?*
- *En que orden analizamos los subproblemas abiertos.(nodos del árbol)*

Ejemplo:

$$\text{Max } 4\,x_1 - x_2$$

Sujeto a

$$7\,x_1 - 2\,x_2 \leq 14$$

$$x_2 \leq 3$$

$$2\,x_1 - 2\,x_2 \leq 3$$

$$x \in Z^2_+$$

- Resolviendo la relajación lineal del problema original tenemos la solución x= (20/7,3) con un valor de $\bar{z}$ =59/7.

- En este caso tenemos una sola variable, x_1 que es fraccionaria. Dividimos el problema entonces agregando para un lado la restricción $x_1 \leq 2$ y para el otro $x_1 \geq 3$. O sea

  $S_1 = S \cap \{x / x_1 \leq 2\}$ y $S_2 = S \cap \{x / x_1 \geq 3\}$

- Elegimos ahora resolver S_1. Agregamos la nueva restricción que es sólo una cota de una variable, $x_1 \leq 2$ y resolvemos usando el método simplex a partir de la última solución básica. Obtenemos la solución óptima de S_1 x = (2,1/2) con valor $\bar{z}_1$ = 15/2.

- Tenemos de nuevo una sola variable que no es entera. Partimos S_1 usando las restricciones $x_2 \leq 0$ y $x_2 \geq 1$ y tenemos dos nuevos nodos

$S_{11} = S \cap \{x / x_1 \leq 2 \text{ y } x_2 = 0\}$ y

$S_{12} = S \cap \{x / x_1 \leq 2 \text{ y } x_2 \geq 1\}$

- Tenemos ahora tres nodos “abiertos” para continuar: S_2 S_{11} y S_{12} . Elegimos arbitrariamente resolver el problema lineal correspondiente a S_2 y vemos que es un problema no factible.

- Elegimos ahora $S_{12} = S \cap \{x / x_1 \leq 2 \text{ y } x_2 \geq 1\}$ y obtenemos una solución x= (2,1) con valor $z^{12} = 7$. Esta es la mejor solución entera hasta el momento y tenemos entonces una cota inferior z_ = 7 . Por esta rama no tenemos que abrir más nodos.

- Elegimos el único nodo que nos queda,

$S_{11} = S \cap \{x / x_1 \leq 2 \text{ y } x_2 = 0\}$. Al resolverlo obtenemos x = (3/2, 0) con valor $\bar{z} = 6$. Como ya teníamos una cota z_ = 7, podemos podar esta rama también y la solución óptima es (2,1) con valor 7.

- *Qué trabajo requiere desarrollar un programa de un método tipo Branch and Bound basado en programación lineal?.*

- Obtener las mejores cotas posibles
- Reglas de branching. Qué variable elegir?.
- Reglas para decidir que nodo “abierto” procesar.
- Cómo resolver el problema de PL en cada nodo?.
- Cómo almacenar el árbol?. Qué almacenar? (*poca información versus muchos cálculos).*

Si la cantidad de nodos abiertos en el algoritmo de branch and bound es muy grande podemos tener problemas de falta de memoria en la computadora.

No hay una solución mejor para todos los problemas

Qué ofrecen los paquetes comerciales de PLE?

- Prepocesamiento
- Elección entre varias estratégias de pivotaje. Uso de métodos de punto interior en el nodo raíz.
- Elección entre varias estrategias de branching.

(GUB branching, Strong branching)

- Definición de prioridades de las variables.
- Fijar por costo reducido
- Heuristicas primales
- El usuario puede proveer una buena solución factible, es importante pasar el valor para usarlo como cota.

Preprocesamiento

Los programas comerciales también incluyen una etapa inicial de preprocesamiento donde se intenta detectar rápidamente si se pueden eliminar restricciones o variables que sean redundantes o mejorar las cotas de algunas variables.

- Reglas generales

Ejemplo: usando reglas de preprocesamiento podemos reducir el sistema

$$\text{Max } 2x_1 + x_2 - 2x_3$$

s. a

$$5x_1 - 2x_2 + 8x_3 \leq 15$$
$$8x_1 + 3x_2 - x_3 \geq 9$$
$$x_1 + x_2 + x_3 \leq 6$$
$$0 \leq x_1 \leq 3$$
$$0 \leq x_2 \leq 1$$
$$1 \leq x_3$$

al sistema

$$\text{Max } 2x_1 + x_2 - 2x_3$$

s.a

$$5x_1 - 2x_2 + 8x_3 \leq 15$$
$$8x_1 + 3x_2 - x_3 \geq 9$$
$$7/8 \leq x_1 \leq 9/5$$
$$0 \leq x_2 \leq 1$$
$$1 \leq x_3 \leq 101/64$$

Ejemplo: usando solamente reglas lógicas resolver el sistema booleano

$$7x_1 + 3x_2 - 4x_3 - 2x_4 \leq 1$$
$$-2x_1 + 7x_2 + 3x_3 + x_4 \leq 6$$
$$-2x_2 - 3x_3 - 6x_4 \leq -5$$
$$3x_1 - 2x_3 \geq -1$$
$$x \in B^4$$

Algoritmos de Planos de Corte

•**Problema:**

$$\max \{cx / x \in X\}$$

con $X = \{x / Ax \leq b, x \in Z^n_+\}$

•**Proposición:** conv (X) es un poliedro que puede entonces escribirse como

$$\text{conv}(X) = \{x / \underline{A}x \leq \underline{b}, x \geq 0\}$$

• Lo mismo ocurre para un problema de programación lineal entera mixta.

- Situación ideal: tenemos una descripción completa de la cápsula convexa. (*problemas fáciles).*

- Si el problema es NP-Hard no tenemos esperanza de poder tener una buena descripción de conv(X).

Cómo podemos obtener una aproximación de la cápsula convexa?

- **Def:** Una desigualdad $\pi x \leq \pi_0$ es una desigualdad válida para $X \subset R^n$ si se verifica que $\pi x \leq \pi_0$ para todo $x \in X$.

- *Qué desigualdades válidas son "buenas" o útiles?*
- *Cómo usarlas en un problema en particular?*

Ejemplos: Desigualdades "lógicas" como las que ya vimos en los ejemplos de preprocesamiento, redondeo entero, características del problema, etc. (ver ejemplos Wosley pág 114-117).

- *Cómo generar desigualdades válidas?*
- *Cómo probar que son válidas?*

Proposición: $\pi x \leq \pi_0$ es una desigualdad válida para $P = \{x / Ax \leq b, x \geq 0\} \neq \emptyset$ si y sólo si $\exists\ u \geq 0, v \geq 0$ tal que $uA - v = \pi$ y $ub \leq \pi_0$ o $\exists\ u \geq 0$ tal que $uA \geq \pi$ y $ub \leq \pi_0$

Proposición: Si $X = \{ y \in Z^1 / y \leq b \}$ entonces $y \leq \lfloor b \rfloor$ es una desigualdad válida para X.

Ejemplos: desigualdades válidas para Matching y otros problemas (Wosley, pág 118).

Método de Chvátal- Gomory para construir una desigualdad válida.

Sea $X = P \cap Z^n$, $P = \{x \in R^n / Ax \leq b\}$, A matriz de mxn

$\{a_1, a_2 \ldots\ldots\ldots a_n\}$ columnas de A, $u \in R^m_+$.

i) La desigualdad $\sum_j u\, a_j\, x_j \leq ub$ es válida para P

ii) $\sum_j \lfloor u\, a_j \rfloor x_j \leq ub$ es válida para P.

iii) $\sum_j \lfloor u\, a_j \rfloor x_j \leq \lfloor u\, b \rfloor$ es valida para X.

- Las desigualdades originalmente propuestas por Gomory, que son una clase particular de estas y que fueron las que dieron origen a este método, se pueden expresar a partir de la base óptima de la relajación lineal, incorporar al último sistema y reoptimizar a partir de allí. Ese es el método que Gomory propuso a comienzos de los 60.

- **Teorema:** Toda desigualdad válida para X puede ser obtenida aplicando C-G un número finito de pasos.

- *Porqué esto no es suficiente para considerar resuelto el problema de PLE?.*

- Más en general si podemos determinar una familia de desigualdades válidas para el problema podemos agregarlas a la formulación y usar un paquete comercial para branch and bound.

Ejemplos: Problema de localización sin restricciones de capacidad. Problema de Lost-Sizing con restricciones de capacidad (Wosley pág 122)

Algoritmos de planos de corte

Supongamos que $X = P \cap Z^n$ y que conocemos una familia F de desigualdades válidas para X, $\pi x \leq \pi_0 \in F$.

- F puede tener un número muy grande de desigualdades como para agregarlas directamente a la formulación.

En estos casos usamos un algoritmo de planos de corte para intentar resolver el problema $IP = \max \{cx: x \in X\}$.

- Para una función objetivo específica el objetivo no es encontrar la cápsula convexa completa, sólo necesitamos una buena aproximación "cerca" de la solución óptima.

- **Problema de separación para F**: dada una solución x^* de la relajación lineal $\underline{z} = \max \{cx: x \in P\}$ encontrar una desigualdad $\pi^t x \leq \pi^t_0$ de F, tal que $\pi^t x^* > \pi^t_0$, o sea una desigualdad que corta x^*

Algoritmo:

1. **Inicialización**: $t= 0$, $P^0 = P$
2. **Iteración t**. Resolver el problema de programación lineal.

$$\underline{z}^t = \max \{cx: x \in P^t\}$$

Sea x^t una solución óptima.

- Si $x^t \in Z^n$ parar. x^t es óptima para IP
- Si $x^t \notin Z^n$ resolver el *problema de separación* para x^t y la familia de desigualdades válidas F.
  - Si se encuentra una desigualdad $\pi^t x \leq \pi^t_0$ en F, tal que $\pi^t x^t > \pi^t_0$, o sea una desigualdad que corta x^t ponemos $P^t = P \cap \{x / \pi^t x \leq \pi^t_0\}$, $t = t+1$.
  - Sino, parar

- *Qué implica resolver el problema de separación ?*

- *Cómo se implementa esto?* Se puede usar un paquete comercial de programación lineal, que permita incorporar nuevos cortes y algoritmos de separación.

- En la práctica es a menudo mejor agregar varios cortes violados por vez y no uno por iteración.

- *Qué pasa si el algoritmo termina sin solución entera para IP?*

$P^t = P \cap \{x / \pi^i x \leq \pi^i_0, i = 1 \ldots t\}$ es una formulación más ajustada del problema que puede ser usar para iniciar un branch and bound.

Algoritmo de planos de corte de Gomory

- Tenemos el problema

$$\max \{cx / \underline{A}x = \underline{b},\ x \geq 0,\ x \text{ entero}\}$$

- Supongamos que tenemos una solución óptima de la relajación lineal

$$x_{Bi} + \sum_{j \in N} \underline{a}_{ij}\, x_j = \underline{a}_{i0} \text{ para } i=1,\ldots.m \quad (*)$$

- Si la solución básica no es entera existe $\underline{a}_{i0} \notin Z$ para alguna fila i.

$$x_{Bi} + \sum_{j \in N} \lfloor \underline{a}_{ij} \rfloor\, x_j \leq \lfloor \underline{a}_{i0} \rfloor \quad (**)$$

- Restando (*) y (**) queda $\sum_{j \in N} (\underline{a}_{ij} - \lfloor \underline{a}_{ij} \rfloor)\, x_j \geq \underline{a}_{i0} - \lfloor \underline{a}_{i0} \rfloor$

- Ponemos $f_{ij} = \underline{a}_{ij} - \lfloor \underline{a}_{ij} \rfloor$ para $j \in N$ y $f_{i0} = \underline{a}_{i0} - \lfloor \underline{a}_{i0} \rfloor$.

- Esta desigualdad corta la solución óptima x*

- Definimos la slack $s = - f_{j0} + \sum_{j \in N} f_{ij}\, x_j$, agregamos la nueva ecuación y reoptimizamos.

- S es entera y nonegativa.

- Estas ideas se pueden aplicar también a problemas de Programación Lineal Entera Mixta.

- El primer algoritmo de corte que se reportó es un caso del TSP de 1954 (Dantzig, Fulkerson, Johnson). En este caso el problema de separación se resolvió detectando los cortes violados mirando la solución.

- En la práctica esto no es suficiente para resolver problemas de programación lineal entera. Se requiere derivar otro tipo de desigualdades “más fuertes” y en general ad-hoc para cada problema.

Ej: Wosley (pág 125)

Proposición: Sea β_i una fila de B^{-1} y $q_i = \beta_i - \lfloor \beta_i \rfloor$ para i = 1….m

El corte de Gomory $\sum_{j \in N} f_{ij} x_j \geq f_{j0}$ es una desigualdad de Chvatal- Gomory que se puede escribir en terminos de las variables originales como

$$\sum \lfloor q\, a_j \rfloor x_j \leq \lfloor q\, b \rfloor$$

Desigualdades válidas fuertes

- Qué es una desigualdad válida fuerte para un problema de PLE?
- Describir una familia interesante de desigualdades fuertes para un problema particular puede ser muy difícil.
- Dada una familia de desigualdades fuertes el problema de separación puede ser difícil de resolver.
- Para problemas difíciles hay que usar estas desigualdades en el marco de un Branch and Bound (algoritmos Branch and Cut).
- Si el problema es NP-Hard no hay esperanza (salvo que sea P = NP) de obtener una descripción explícita completa de conv(X)
- Si tenemos $X = X_1 \cap X_2$ y el problema sobre X_2 está en P, se puede intentar determinar conv(X_2)
- Si tenemos $X = X_1 \cap X_2$ con ambos probellmas NP-hard, puede de todos modos ser más fácil determinarn desigualdades válidas separados para X_1 y X_2

- Una desigualdad válida $\pi x \leq \pi_0$ para P domina a otra desigualdad válida $\mu x \leq \mu_0$ si existe $v > 0$ tal que $\pi \geq v\,\mu$ y $\pi_0 \leq v\,\mu_0$, $(\pi, \pi_0) \neq (v\,\mu, v\,\mu_0)$. Esto implica que

$$\{x \in R^n_+ / \pi x \leq \pi_0\} \subset \{x \in R^n_+ / \mu x \leq \mu_0\}$$

Ej : $2x_1 + 4x_2 \leq 9$ es dominada por $x_1 + 3x_2 \leq 4$

(poner v = 0.5)

- Una desigualdad válida $\pi x \leq \pi_0$ para P es redundante si existen $k \geq 2$ desigualdades válidas para P tal que una combinación lineal de ellas con pesos positivos domina a $\pi x \leq \pi_0$.

Para que sirve saber si una desigualdad es redundante?

- **Teorema:** Si P es un poliedro de dimensión completa entonces tiene una única descripción minimal

$$P = \{x \in R^n / a^i x \leq b_i \text{ para } i = 1,\ldots m \}$$

donde cada desigualdad es única (salvo múltiplos positivos)

- Todas estas desigualdades son entonces ***necesarias*** para la descripción de la cápsula convexa. Si se suprime alguna de ellas el poliedro ya no es el mismo P.

- Si una desigualdad válida para P, no es múltiplo de alguna de las que describen P es redundante.

- Se puede demostrar que la dimensión de P es el número máximo número de puntos afinmente independientes en P menos 1.
- Si P es de dimensión completa dim (P) = n una desigualdad válida $\pi x \leq \pi_0$ es necesaria en la descripción de P si y sólo si es una faceta de P.
- Entonces si P es de dimensión completa, $\pi x \leq \pi_0$ define una faceta de P si y sólo si hay n puntos afinmente independientes que la satisfacen por igualdad.

Ej: Wosley pág 143.

Cómo probar que una desigualdad es una faceta?

Cómo probar que se tiene la descripción completan de la cápsula convexa de X?

Cómo probar que una desigualdad es una faceta?

Suponemos conv(X) acotado y de dimensión completa.

- Encontrar n puntos afinmente independientes que verifiquen

  $\pi x = \pi_0$

- i) Determinar $t \geq n$ puntos $x^1 \ldots\ldots\ldots x^t \in X$ que satisfagan

$\pi x = \pi_0$

ii) Suponemos que todos los puntos están en un hiperplano $\mu x = \mu_0$

Resolvemos el sistema lineal

$$\sum \mu_j x^k_j = \mu_0 \text{ para } k = 1 \ldots t$$

donde μ, μ_0 son las incógnitas.

iii) Si la única solución es $(\mu, \mu_0) = \lambda (\pi, \pi_0)$ para $\lambda \neq 0$ entonces la desigualdad $\pi x \leq \pi_0$ define una faceta. E*stamos verificando en forma indirecta que los puntos son afinmente independientes.*

Ej: Wosley pág 144.

Cómo intentar probar que se tiene la descripción completa de la cápsula convexa de X?.

- Aprovechar la estructura especial del problema.

Por ejemplo cuando la matriz es unimodular.

- Mostrar que los puntos fraccionales no son extremos de P.

- Probar que para todo $c \in R^n$ la solución óptima de

$$z^{LP} = \max \{cx / Ax \leq b\}$$

es entera.

- Probar que para todo $c \in R^n$ existe un punto $x^* \in X$ y una solución factible u^* del dual

$$w^{LP} = \min \{ub / uA = c, c \geq 0\}$$

tal que $c\, x^* = u^* b$.

- Probar que si $\pi x \leq \pi_0$ es una faceta de conv(X) entonces es una de las desigualdades que definen P.

- Verificar que $b \in Z^n$ y mostrar que para todo $c \in Z^n$ el valor óptimo del dual $w^{LP} = \min \{ub/\ uA = c,\ c \geq 0\}$ es entero

- Supongamos que $Q \subset R^n \times R^p$ es un poliedro tal que

  $P = proy_x(Q) = \{x \in R^n / (x,w) \in Q \text{ para algún } w \in R^p\}$.
  Mostrar que para todo $c \in R^n$ $\max \{cx/\ (x,w) \in Q\}$ tiene solución óptima con $x \in X$.

Desigualdades para restricciones mochila 0-1

Sea $X = \{x \in B^n / \sum_j a_j x_j \leq b\}$

Asumimos que b y los a_j son positivos (eventualmente se hace un cambio de variables)

- Un conjunto $C \subset N$ es un cubrimiento si $\sum_j a_j > b$. es un cubrimento minimal si $C \setminus \{j\}$ no es cubrimiento para ningún $j \in C$

- Si C es un recubrimiento, la desigualdad de cubrimiento (*cover)*

$$\sum_{j \in C} x_j \leq |C| - 1 \text{ es válida para X.}$$

Ej: $X = \{x \in B^7 / 11x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \leq 19\}$

Cómo podemos "reforzar" esta desigualdades?

- Si C es un recubrimiento, la desigualdad extendida de cubrimiento

$$\sum_{j \in E(C)} x_j \leq |C| - 1$$

donde $E(C) = C \cup \{j / a_j \geq a_i \text{ para todo } i \in C\}$

Ej: Poner C = {3,4,5,6} en el ejemplo anterior

Cómo podemos dar un procedimiento para mejorar esta desigualdad aún más?.

Ej: ver continuación del ejemplo anterior, pág 148.

En el caso general queremos encontrar los mejores valores posibles para α_j , $j \in N \backslash C$ tal que la desigualdad

$$\sum_{j \in C} x_j + \sum \alpha_j x_j \leq |C| - 1$$

es válida para X.

Se puede formular un procedimiento general de “lifting” para obtener desigualdades que definen facetas a partir de las desigualdades minimales de cover. (Wosley, pág 149)

Separación para las desigualdades de cover

Problema: dado un punto x^*, $0 \leq x^*_j \leq 1$, con algún x^*_j no entero, queremos saber si x^* satisface o no todas las desigualdades de cover.

- Podemos escribir las desigualdades de cover como:

$$\sum_{j \in C} (1 - x_j) \geq 1$$

- Entonces queremos determinar si existe un conjunto $C \subset N$ tal que $\sum_{j \in C} a_j > b$ para el cual $\sum_{j \in C} (1 - x^*_j) \geq 1$.

- O sea queremos saber si existe

$$\xi = \min_{C \subset N} \{\sum_{j \in C} (1 - x^*_j) \geq 1 / \sum_{j \in C} a_j > b\} < 1$$

- Cómo no conocemos C, podemos reformular esto como un PLE agregando las variables binarias z_j:

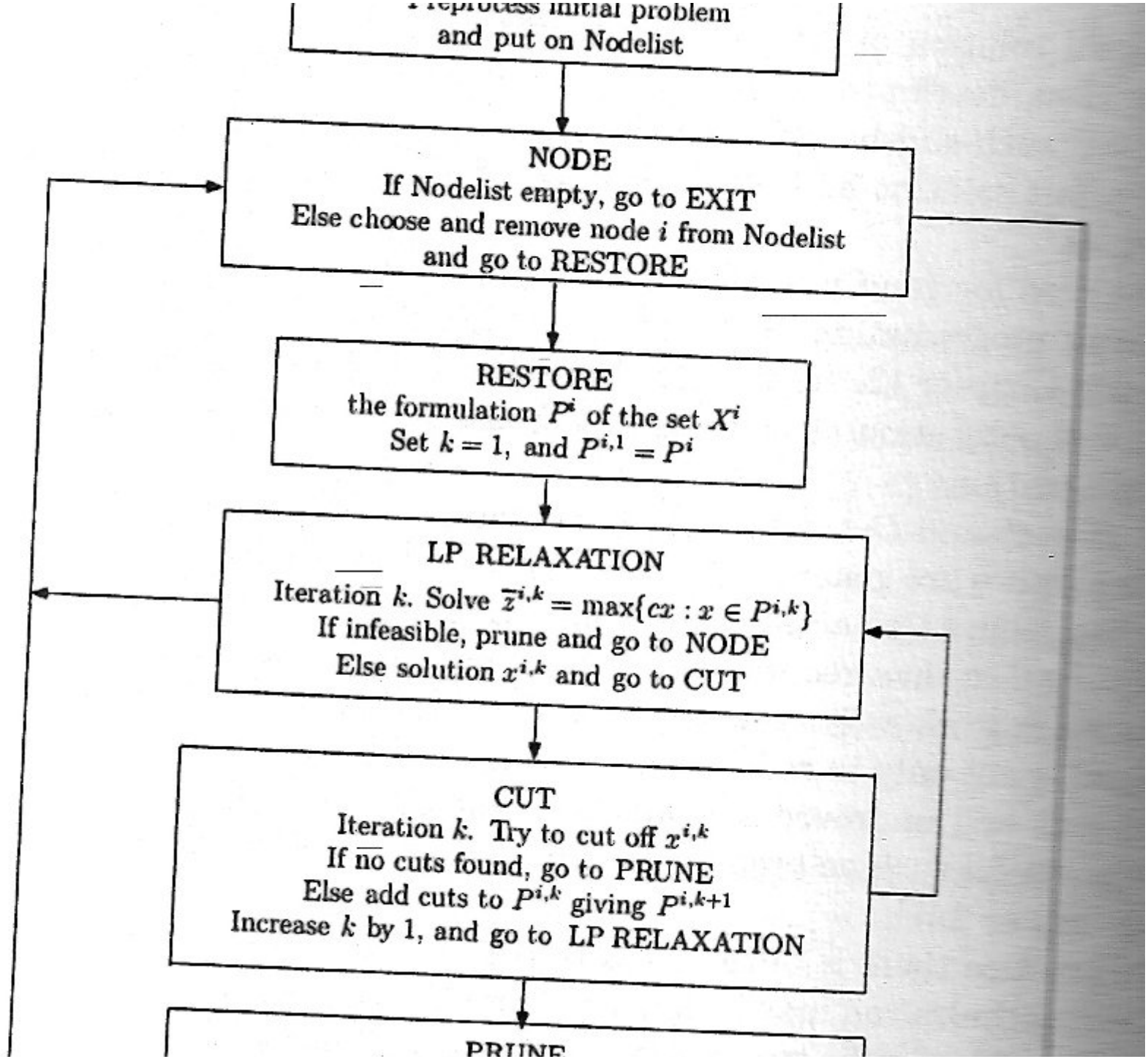
$$\text{Es } \xi = \min \{ \sum_{jN} (1 - x^*_j) z_j \geq 1 / \sum_{j \in C} a_j z_j > b, z \in B^n \} < 1?$$

- Si $\xi \geq 1$, x* satisface todas las desigualdades de cover.
- Si $\xi < 1$ con solución óptima z^R la desigualdad de cover $\sum_{j \in R} (1 - x_j) \geq 1$ corta a x*.

Ej: Wosley (pág 150)

Métodos branch and cut

- Son algoritmos branch and bound en el cual se generan planos de corte en cada nodo del árbol.
- Combinan todas las ideas que vimos para resolver un problema PLE particular.
- Requiere un trabajo previo para determinar desigualdades válidas, facetas, etc. para el problema específico a tratar.
- En cada nodo se trata de obtener la mejor cota dual posible
- Hay que usar preprocesamiento, heurísticas primales, etc.
- Se almacenan cortes en un pool de cortes.


Preprocess initial problem
and put on Nodelist
NODE
If Nodelist empty, go to EXIT
Else choose and remove node i from Nodelist
and go to RESTORE
RESTORE
the formulation P^i of the set X^i
Set $k = 1$, and $P^{i,1} = P^i$
LP RELAXATION
Iteration k. Solve $\overline{z}^{i,k} = \max\{cx : x \in P^{i,k}\}$
If infeasible, prune and go to NODE
Else solution $x^{i,k}$ and go to CUT
CUT
Iteration k. Try to cut off $x^{i,k}$
If no cuts found, go to PRUNE
Else add cuts to $P^{i,k}$ giving $P^{i,k+1}$
Increase k by 1, and go to LP RELAXATION
PRUNE

Algunos programas de programación lineal y lineal entera:

- LINDO: tiene versión para estudiantes que se puede bajar libremente de http://www.lindo.com.
  Es muy sencillo de usar.

CPLEX : es considerado el mejor programa comercial y académico de programación lineal y lineal entera. Es el standard en la literatura académica y para las empresas. Su primera versión fue puesta en público en 1988 y desde entonces siguen mejorándolo. Se puede bajar una demo de http:
//www-01.ibm.com/software/integration/optimization/cplex-optimizer/.
Se puede usar en forma sencilla ingresando la formulación del problema mediante un lenguaje de modelado. Para problemas más complejos, incorporación de cortes y rutinas de separación, etc., se usa una biblioteca de rutinas. Es gratuito para uso académico.

- GUROBI: http://www.gurobi.com/: es un programa académico y comercial para programación lineal y entera, relativamente nuevo, que está mostrando resultados competitivos con CPLEX

- GAMS: es un programa que usan mucho los economistas, trae muchos ejemplos de aplicaciones. Se puede bajar una versión limitada que sirve para estudiantes de http://www.gams.de

- XPRESS: es otro programa del tipo del CPLEX, pero que tiene un poco menos de difusión, y aparentemente no tan eficiente en problemas difíciles. http://www.dashoptimization.com

- LP- Solve programa de acceso libre.
- *Hay otros programas de software libre.*