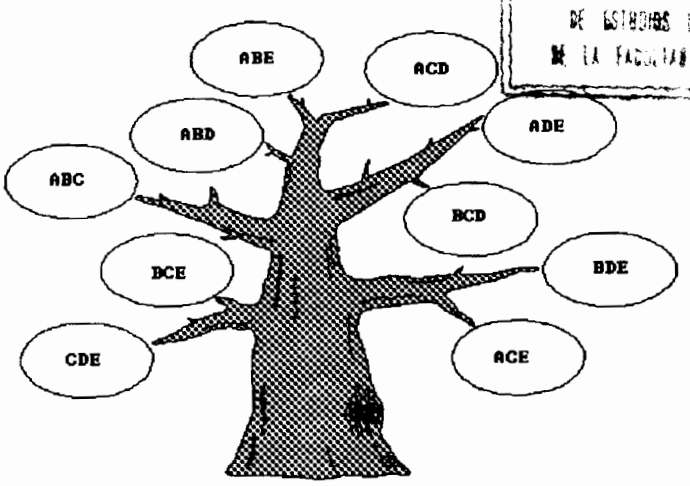
APUNTES DE PROGRAMACION ENTERA

IDALIA FLORES DE LA MOTA



Departamento de Ingeniería de Sistemas

División de Estudios de Posgrado **Facultad de Ingeniería, U.N.A.M.**

Como parte de las actividades del Departamento de Ingenieria de Sistemas de la Division de Estudios de Posgrado, Facultad de Ingenieria UNAM, nos hemos propuesto el desarrollo de una serie de apuntes que sirvan de apoyo a los distintos cursos que se imparten y, desde luego, como material de referencia o de lectura para quien asi lo desee.

En dichos apuntes se busca mantener un alto nivel, de manera que contribuyan a una solida formacion teorica del alumno, pero al mismo tiempo se intenta mantenerlo cerca de las aplicaciones de tales conocimientos, como es en sintesis la aspiracion genral del posgrado de ingenieria.

La presentación de los mismos es en siete capítulos divididos en dos partes: la primera de ellas abarca los primeros cuatro capítulos y la segunda los tres últimos.

Agradecemos al Programa de Apoyo a las Divisiones de Estudios de Posgrado (PADEP), por financiar éstos apuntes.Finalmente agradecemos a la Sra Basilisa Arroyo por el apoyo tipográfico que permitió que se concluyera este trabajo.

PROLOGO

La Programación Entera es una disciplina dentro de la Investigación de Operaciones, en particular dentro de la Programación Lineal que se ha desarrollado ampliamente en las últimas dos décadas. Junto con algunas técnicas de teoría de gráficas se han desarrollado algoritmos que se pueden ubicar también dentro de la Optimización Combinatoria como veremos en éstos apuntes. Por otro lado el desarrollo de la computación ha traido consigo un gran avance en las técnicas de la programación entera, es por ésto que es de suma importancia que el estudiante cuente con bases sólidas en ésta disciplina. Es bajo ésta perspectiva, que se presentan estos apuntes donde lo que se busca es tener un apoyo didáctico para la clase y como un complemento de la bibliografía sugerida para el curso.

Los presentes apuntes van dirigidos a estudiantes de posgrado que ya tienen conocimientos de la Programación Lineal y desean profundizar en problemas de tipo entero. No se presentan todos los algoritmos que se pueden cubrir en el curso, sin embargo se sientan bases para poder trabajar aquellos que están ausentes con cierta facilidad. El propósito de éstos apuntes es que los alumnos que cursan la materia y requieren de un cierto grado de conceptualización y una buena dosis de ejemplos tengan un documento de trabajo que les facilite el acceso a otras lecturas; se presentan también al final de cada capitulo unas notas históricas que buscan dar al lector una idea mas completa de la programación entera.

INDICE

CAPITULO 1.
CONCEPTOS BASICOS 1

1.1 INTRODUCCION 1
1.2 PROBLEMAS DE OPTIMIZACIÓN COMBINATORIA. 3
COMPLEJIDAD DE LOS PROBLEMAS. 4
1.3 MÉTODOS DE SOLUCION PARA OPTIMIZACION ENTERA Y COMBINATORIA. 7

CAPITULO 2
COMPLEJIDAD COMPUTACIONAL 11

2.1 INTRODUCCION 11
2.2 COMPLEJIDAD DE ALGORITMOS: TIEMPO Y ESPACIO . . . 12
2.3 PEOR CASO Y CASO PROBABILISTICO 16
2.4 ANALISIS ASINTOTICO DE FUNCIONES. 17
2.5 VELOCIDAD DE CRECIMIENTO Y CALCULO DE TIEMPO DE EJECUCION DE UN PROGRAMA. 23
TIEMPO DE EJECUCION DE UN PROGRAMA 25
2.5 NOTAS HISTORICAS 28

CAPITULO 3
METODOS DE PLANOS DE CORTE 29

3.1 METODOS DE PLANOS DE CORTE 29
FORMA GENERAL DEL ALGORITMO 30
DUALIDAD 31
3.2 ALGORITMO DUAL FRACCIONARIO. 34
3.3 LA NOTACION DE TABLEAU 45
3.4 EL CORTE DE GOMORY 47
3.5 ALGORITMO ENTERO PURO DE GOMORY 51

CAPITULO 4
PROPIEDADES DE LOS METODOS DE PLANOS DE CORTE 63

4.1 COMPARACIÓN ENTRE LOS CORTES GENERADOS POR EL ALGORITMO FRACCIONAL Y EL ALGORITMO ENTERO PURO DE GOMORY. 63
4.2 ALGUNAS PROPIEDADES AL AUMENTAR CORTES 69
4.2.1 CASO DUAL FRACCIONARIO 69
4.2.2 CASO DEL ALGORITMO ENTERO PURO 79
4.3 LA FUERZA RELATIVA AL AUMENTAR DESIGUALDADES . . 86
4.4 ALGORITMO DUAL FRACCIONARIO MIXTO 90
4.5 NOTAS HISTORICAS 95

CAPITULO 5
LA TECNICA DE RAMIFICACION Y ACOTAMIENTO: ENFOQUE CLASICO . 96

5.1 DESCRIPCION CONCEPTUAL DE RAMIFICACION Y ACOTAMIENTO 96
5.2 EL ARBOL DE BUSQUEDA. 98
EL ALGORITMO BASICO DE RAMIFICACION Y ACOTAMIENTO 100
CONVERGENCIA DEL ALGORITMO 100
5.3 ESTRATEGIAS OPERATIVAS EN EL ALGORITMO BASICO. . 101
RAMIFICACION ORDENADA 101
RAMIFICACION INMEDIATA DEL SUCESOR 103
RAMIFICACION DEL SUCESOR 104
ACOTAMIENTO 104
BUSQUEDA A PRIMER PROFUNDIDAD O REGLA DE LA COTA MAS RECIENTE O TAMBIEN LIFO 105
PRIMER ALCANCE O REGLA DE LA MENOR COTA O TAMBIEN FIFO 105
COSTO UNIFORME 106
DOMINACION 106
5.5 EJEMPLOS ILUSTRATIVOS. 107
EJEMPLO 5.1 (RAMIFICACIÓN) 108
EJEMPLO 5.2 PROBLEMA DE ASIGNACIÓN LINEAL (ACOTAMIENTO) 110
EJEMPLO 5.3 (EL PROBLEMA DE LAS QUATRO REINAS) . 112

CAPITULO 6
METODOS DE RAMIFICACION Y ACOTAMIENTO PARA PROGRAMACION ENTERA MIXTA 117

6.1 EL ALGORITMO DE LAND Y DOIG. 117
DESCRIPCION DEL METODO 118
REGLAS PARA AUMENTAR ARCOS Y NODOS 123
ALGORITMO 6.1 LAND-DOIG 126
6.2 ALGORITMO DE DAKIN 134
ALGORITMO 6.2: DAKIN 136
6.3 COMPARACION CON EL METODO DE LAND Y DOIG 140
6.4 ALGORITMO PARA LA SOLUCION DE PROBLEMAS DE PROGRAMACION ENTERA MIXTA DE DRIEBEEK 144
CONVERGENCIA 145

CAPITULO 7
EL PROBLEMA DEL AGENTE VIAJERO Y EL PROBLEMA DE LA MOCHILA. 154

7.1 EL PROBLEMA DEL AGENTE VIAJERO 155
7.2 ALGORITMO DE RAMIFICACION Y ACOTAMIENTO 159
METODO DE RAMIFICACION Y ACOTAMIENTO BASADO EN LA MATRIZ REDUCIDA 159
ALGORITMO 7.1: LITTLE-MURTY 168
7.3 EL PROBLEMA DE LA MOCHILA 169
REDUCCION DEL PROBLEMA LINEAL ENTERO AL PROBLEMA DE LA MOCHILA. 172
METODOS COMPUTACIONALES 174
7.4 METODO DE RAMIFICACION Y ACOTAMIENTO 174
ALGORITMO DE KOLESAR 175
ALGORITMO 7.2: KOLESAR 178
7.5 NOTAS HISTORICAS 182

APENDICE A 184

CONJUNTOS CONVEXOS 184
A.1 CONJUNTOS CONVEXOS 184
A.2 HIPERPLANOS Y POLITOPOS 185

APENDICE B

METODO DUAL SIMPLEX 189

BIBLIOGRAFIA 192

CAPITULO 1.

CONCEPTOS BASICOS

1.1 INTRODUCCION

Es muy difícil construir una única teoría capaz de describir todo el universo. En vez de ello, nos vemos forzados, de momento, a dividir el problema en varias partes, inventando un cierto número de teorías parciales.
Stephen W. Hawking. "Historia del Tiempo"

La toma de decisiones ha estado presente a lo largo de toda la historia de la humanidad. Sin embargo la racionalización y sistematización de esta tarea en problemas industriales y públicos de envergadura, solo fue posible hasta fines de la Segunda Guerra Mundial al desarrollarse lo que hoy conocemos como Investigación de Operaciones. Dos factores importantes confluyeron e impusieron un rápido crecimiento a la nueva disciplina, como base fundamental para la toma de decisiones en amplias esferas de la sociedad:

El primero fue el progreso sustancial en la modelación matemática y la consolidación del método simplex para resolver problemas de programación lineal, desarrollada por el matemático G. Dantzig, así como la programación dinámica, la teoría de inventarios y la teoría de colas que estaban relativamente bien desarrolladas antes de finalizar los años 50.

El segundo factor fue, la irrupción del computador digital que proporcionó al tomador de decisiones una tremenda capacidad en velocidad de cómputo, almacenamiento y retiro de información. Permitiendo la aplicación y popularización de las herramientas convencionales de la I. de O.

Estos factores han dejado profunda huella durante estos primeros 40 años de desarrollo de la I. de O. El primero de ellos, sobre todo la programación lineal, tuvo un papel predominante en las técnicas de solución de problemas, mientras que el segundo juega un papel de apoyo al primero, al permitir efectuar largos procedimientos de cálculo. Esto ocasionó una de las primeras desvirtuaciones que tenemos de la I. de O., la de reducirla a una simple herramienta de solución de problemas, más aún, de confundir la investigación de operaciones con la programación lineal, este hecho se manifiesta en planes de estudio, de maestrías en administración y diversas licenciaturas donde existe la materia de Investigación de Operaciones, con programas correspondientes relativos a la programación lineal.

La Investigación de Operaciones, tuvo sus orígenes a lo largo de la segunda guerra mundial. La ciencia ha contribuido, desde los tiempos de Arquímedes, a aportar ideas destructoras, y en las dos grandes guerras de este siglo dio su ayuda técnica para hacer posible el desarrollo de sus principales armas, desde la ametralladora hasta la bomba atómica. En la última guerra, los analistas de operaciones militares se encontraron trabajando en lugares extraños y en diversas circunstancias. En Burma hubo matemáticos que discutieron problemas artilleros con soldados británicos, en Princess Risborough, un cuartel seguro fuera de Londres; unos químicos, en combinación con colegas economistas valoraron la capacidad destructora de una bomba; hubo generales que consultaron la estrategia de los carros de combate en la campaña de Italia con bioquímicos y abogados; un famoso zoólogo británico fue el hombre clave en el trazado de un plan de bombardeo sobre Pantellaria; oficiales de marina pusieron bajo secreto a estadísticos y entomólogos, en relación con las pérdidas de submarinos en el Pacífico. Fue un intenso, irregular y paradójico intercambio de ideas entre aficionados y profesionales de la guerra, que produjo algunos éxitos brillantes, aunque hubo más fracasos que éxitos, pero el balance final es impresionante.

Lo que aportaron los científicos a los problemas operativos -aparte de sus conocimientos especiales- fue su aspecto científico. De hecho ésta fue su principal contribución, aportaron ideas nuevas, desecharon conceptos preconcebidos y obraron sólo ante la evidencia. El cálculo de probabilidades fue su herramienta indispensable, e hicieron uso de su teoría más sutil y de su técnica más eficaz, no estuvieron limitados por axiomas de laboratorio, sino que su guía inseparable fue el sistema experimental. Sin embargo a pesar de haber empezado en la guerra, la investigación de operaciones, se ha ido extendiendo a la ingeniería, las comunicaciones, la minería, los negocios, la manufactura y otras ramas de la industria.

En nuestro país la Investigación de Operaciones es realmente muy joven ya que hasta hace aproximadamente veinte años se comenzó a trabajar en forma sistemática. El primer paso en este sentido fue la familiarización con las técnicas básicas de la I. de O. Como era de esperarse, al principio - y actualmente todavía sucede aunque en menor grado- lo que se hizo fue copiar las aplicaciones desarrolladas mecánicamente, esto es, sin ninguna creatividad o esfuerzo por usar las herramientas de la I. de O., como lo que son, instrumentos de trabajo para la solución de problemas.

El creciente uso de la I. de O. ha hecho que sus técnicas se vayan desarrollando y diferenciando más, teniendo un desarrollo propio. Entre otras podemos identificar la Teoría de Juegos, Teoría de Colas, Simulación Digital, Procesos Estocásticos y Optimización; la tabla 1 nos muestra esquemáticamente ésta idea, y desarrollamos el modulo de optimización debido a la importancia para el presente documento de acuerdo a los diferentes problemas que resuelve.

Problemas como los de distribución de un producto a costo mínimo, redes eléctricas o hidráulicas, o expansión de capacidad, entre otros.

En relación a las técnicas de optimización podemos decir que la Programación Lineal se ha usado con éxito en la solución de problemas referentes a la asignación de personal, la mezcla de materiales, la distribución y el transporte y las carteras de inversión. La programación dinámica se ha aplicado con buenos resultados en áreas tales como la planeación de los gastos de comercialización, la estrategia de ventas y la planeación de la producción. La teoría de colas ha tenido aplicaciones en la solución de problemas referentes al congestionamiento del tráfico, al servicio de máquinas sujetas a descomposturas, a la determinación del nivel de la mano de obra, a la programación del tráfico aéreo, al diseño de presas, a la programación de la producción y a la administración de hospitales. Otras técnicas de Investigación de Operaciones, como la teoría de inventarios, la teoría de juegos y la simulación, han tenido exitosas aplicaciones en una gran variedad de contextos.

En cuanto al módulo de optimización es conveniente puntualizar que algunos problemas reales, exigen que sus soluciones sean enteras, esto es, las variables de decisión tienen sentido sólo si adquieren valores enteros. Por ejemplo, es necesario con frecuencia asignar hombres, máquinas y vehículos a las actividades, en cantidades que deben ser enteras. Esta restricción es difícil de manejarse en forma matemática, sin embargo, se han hecho algunos progresos en el desarrollo de procedimientos de solución para el caso de problemas de programación lineal, sujetos a la restricción adicional de que las variables de decisión (todas o algunas) deben tener valores enteros.

1.2 PROBLEMAS DE OPTIMIZACIÓN COMBINATORIA.

Un problema combinatorio es aquél que asigna valores numéricos discretos a algún conjunto finito de variables X, de tal forma que satisfaga un conjunto de restricciones y minimice alguna función objetivo.

Los problemas de optimización combinatoria abundan en la vida diaria. Una área importante y extensa de aplicaciones se refiere a la administración eficiente del uso de recursos escasos para incrementar la productividad. Estas aplicaciones incluyen problemas operacionales tales como distribución de bienes, planeación de la producción y secuenciación de máquinas . También incluyen problemas de planeación tales como inversión de capital, localización de medios y selección de cartera. También problema de diseño como diseño de redes de telecomunicación y transporte, diseño de circuitos y diseño de sistemas de producción automática. Los problemas de optimización discreta también se presentan en estadística (análisis de datos), física (determinación de estados

de energía mínimos), criptografía (diseñando códigos infranqueables), política (seleccionando distritos electorales favorables) y en matemáticas (como una técnica poderosa para probar teoremas combinatorios). Como ya se ha mencionado el gran avance de las computadoras ha repercutido en el desarrollo acelerado de la optimización discreta.

COMPLEJIDAD DE LOS PROBLEMAS.

Podemos clasificar los problemas en cuatro clases de acuerdo a su grado de dificultad como :

1. Problemas indecidibles. Son aquellos problemas para los cuales no se puede escribir un algoritmo. Por ejemplo, se ha probado que un programa que se detendrá en una Máquina de Turing (1937) es de esta clase. El problema de Turing al igual que el décimo problema de Hilbert y el de programación cuadrática en enteros son problemas indecidibles, es decir no sólo no existe un algoritmo polinomial para su solución sino que no existe algoritmo.

2. Problemas Intratables (problemas que se demuestran son difíciles): Son aquellos problemas para los cuales no se pueden desarrollar algoritmos polinomiales. En otras palabras, sólo se pueden resolver con algoritmos exponenciales.

3. Problemas NP (donde NP se entiende por polinomial no determinístico) Esta clase incluye problemas que se pueden resolver en tiempo polinomial si podemos *adivinar* correctamente que ruta computacional se puede seguir. El concepto de *adivinar* es extraño ya que todos los programas computacionales son determinísticos. En general, esta clase incluye todos los problemas que tienen algoritmos exponenciales pero que no se ha probado que no se puedan resolver con algoritmos de tiempo polinomial.

4. Problemas P (donde P se entiende por polinomial). Esta clase incluye todos los problemas que tienen algoritmos de tiempo polinomial. Mucha gente considera a esta clase como una subclase propia de la clase 3.

Un problema se dice polinomial si existe un algoritmo para el cual el tiempo requerido para su solución, está acotado por una función polinomial del tamaño del problema (donde entendemos el tamaño del problema como la longitud de un código, por ejemplo binario de los datos del problema). Se tiene así por ejemplo que en una gráfica G = [X,A] con N=X nodos y M = A arcos, una ruta más corta se encuentra a lo mas en un tiempo O(MN), un flujo máximo en un tiempo $O(N^3)$, un árbol de peso mínimo en un tiempo $O(N^2)$. Sin embargo no todos los problemas combinatorios son polinomiales.

Problemas como el del agente viajero o el de la mochila se conocen como NP-completos, donde NP se entiende como polinomial no determinístico. El problema del agente viajero se puede resolver con un algoritmo determinístico como el de recursividad, donde se especifica en cada paso que arco se debe considerar desde un nodo dado.

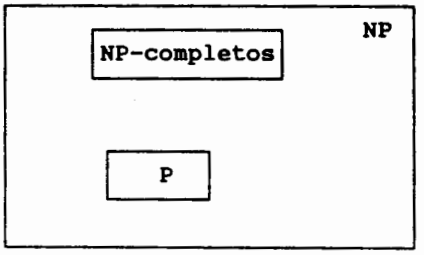
Para la pregunta de que si existe un recorrido con una distancia total que sea menor que B, el algoritmo recursivo implícitamente prueba todos los recorridos posibles y da una respuesta afirmativa si tal recorrido existe y negativa en caso contrario.

Ya que los recorridos se prueban uno por uno, y el último puede ser el que cumple con la propiedad deseada, el algoritmo recursivo para resolver el problema del agente viajero se considera $O(c^n)$.

Un algoritmo no determinístico puede *adivinar correctamente* que arco se debe incluir en el recorrido. Si existe un recorrido con una distancia total menor que B, el algoritmo no determinístico requiere un tiempo de O(n) para calcular la distancia total del recorrido y verificar si tal recorrido existe. Si el problema del agente viajero se puede resolver por un algoritmo no determinístico en tiempo polinomial se dice que el problema pertenece a la clase NP.

Existe un subconjunto de problemas NP que son los más difíciles, en el siguiente sentido (los problemas en el subconjunto son polinomialmente equivalentes):

Cualquier problema en NP se puede reducir a cualquier problema en éste subconjunto, podemos resolver todos los problemas en NP en tiempo polinomial. Los problemas en éste subconjunto se llaman NP-completos. Esquemáticamente se puede ver de la siguiente manera.



A continuación se listan algunos problemas representativos que aborda la optimización combinatoria:

El problema del agente viajero: Dada una gráfica (dirigida o no) con pesos específicos en los arcos, determine una trayectoria cerrada que incluya cada nodo de la gráfica una sola vez y que tenga un peso mínimo total.

El problema del cartero: Para una gráfica dada (dirigida o no) con pesos específicos en los arcos, determine una trayectoria que incluya cada arco en la gráfica al menos una vez y que su peso total sea mínimo.

El problema de la mochila: Determine un conjunto de valores enteros x_i i = 1, 2,...,n que minimice $f(x_1, x_2,...,x_n)$ sujeto a la restricción $g(x_1, x_2,...,x_n) \geq b$ donde b es un parámetro.

Planeación de máquinas en paralelo: Dado un conjunto T de tareas simples de operación , cada una con tiempo de procesamiento r_j , $1 \leq j \leq |T|$, asignar cada tarea a exactamente m máquinas tal que el tiempo en que se realizan todas las tareas sea mínimo.

Coloración de nodos: Dada una gráfica no dirigida, determine el número mínimo de colores necesarios para colorear cada nodo de la gráfica de tal forma que ningún par de nodos adyacentes (nodos conectados por un arco) tengan el mismo color

Arbol de búsqueda: Para una gráfica dada con pesos específicos en los arcos, determinar un subconjunto de peso mínimo total de arcos que forme una gráfica acíclica conectada que tiene al menos un arco incidente a cada vértice.

Ruta mas corta: Para una gráfica dada (dirigida o no) con peso o longitudes específicas en los arcos, encontrar una sucesión de arcos no repetidos de longitud total mínima que conecte dos vértices específicos y que sea factible a la dirección de los arcos.

Empacado de cajas: Para una lista de N pesos w_i, $1 \leq i \leq N$ y un conjunto de cajas, cada una de ellas con una capacidad fija, sea W, encontrar una asignación factible de pesos para las cajas que minimice el número total de cajas a usar.

Apareamiento: Dada una lista de artículos i=1,2,...,n y pesos w_{ij} asociados con aparear un artículo i con un artículo j, encontrar un esquema de peso máximo total para aparear los artículos en la lista tal que cada artículo este apareado con, a lo más, otro artículo.

Cubierta de conjuntos: Dado un conjunto finito S y una familia de subconjuntos $\{S_j \subset S | j \in J\}$ y costos asociados C_j a cada S_j, elegir una colección de subconjuntos de costo mínimo total que incluya a cada elemento de S al menos una vez.

Flujo máximo: Dada una gráfica (dirigida o no) y capacidades específicas en los arcos, encontrar un flujo máximo entre dos vértices específicos que sea factible con respecto a las capacidades.

Problema de Cargo Fijo: Dado un conjunto factible S de actividad o niveles de tráfico no negativos, $x = (x_1, x_2, \ldots, x_n)$, costos unitarios v_j para emplear x_j, y costos fijos f_j asignados siempre que x_j sea positiva, seleccionar un costo mínimo local $x \epsilon S$

Problemas como el del agente viajero tienen amplia aplicabilidad al asignar rutas, con un costo mínimo, por ejemplo un comerciante que desea recorrer n ciudades específicas con una distancia total mínima, entre otros. Otro caso es el problema de la mochila que entre otros, se aplica a problemas industriales, tales como: problemas de cargo fijo, selección de proyectos, corte en inventarios y control de presupuestos. Los problemas de asignación también tienen aplicación en inversiones de capital para proyectos independientes, donde además se combinan con problemas de tipo mochila 0-1.

El problema del cartero chino tiene también una amplia gama de aplicaciones tales como : el trazado de rutas de camiones de servicio público para la recolección de desechos sólidos, en el reparto de artículos para una comunidad, o en el recorrido de las plumas de un graficador para el trazado de mapas.

1.3 MÉTODOS DE SOLUCION PARA OPTIMIZACION ENTERA Y COMBINATORIA.

Un problema de optimización discreta se puede expresar como:

$$\min \text{ (o max)} \sum_{j=1}^{n} c_j x_j$$

sujeto a

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i \quad \text{para } i = 1, 2, \ldots, m$$

$$x_j \geq 0 \qquad \text{para } j = 1, 2, \ldots, n$$
$$x_j \text{ entero} \qquad \text{para } j \epsilon Z$$

dicho problema tiene un número finito de soluciones a considerar para identificar una óptima, se puede pensar en que dicho problema se puede resolver por enumeración total, esto es, tratando todas las posibilidades. Sin embargo, el esfuerzo computacional involucrado en la enumeración total se puede convertir en prohibitivo. Por ejemplo un problema con 200 variables de la forma 0 -1 puede tener alrededor de 2^{200} o 10^{60} casos a considerar, uno con 201 variables puede tener el doble, de aquí la necesidad de

desarrollar métodos de enumeración parcial, como los de ramificación y acotamiento o métodos de descripción de poliedros como los de planos de corte.; existen métodos que combinan ambos y algunos otros que se aplican en forma pura o en combinación con estos como programación dinámica con ramificación y acotamiento o teoría de grupos con planos de corte etc. A continuación enunciamos brevemente cada uno de ellos y a grandes rasgos en que consisten

a) Método de planos de corte.
b) Método de Ramificación y Acotamiento.
c) Método de la teoría de grupos.
d) Métodos de la programación dinámica.
e) Métodos heurísticos.

Estos métodos se engloban en lo que también se conoce como Optimización Combinatoria, término que ha emergido en años recientes para describir aquéllas áreas de la programación matemática que se refieren a la solución de problemas de optimización que tienen una estructura pronunciadamente combinatoria o discreta. Este título se usa para unificar términos que cubren Programación Entera, Teoría de Gráficas, partes de Programación Dinámica etc. A continuación se da una breve descripción de los métodos arriba mencionados.

a) **Método de Planos de Corte:** la idea básica es la siguiente: se comienza resolviendo el problema continuo lineal, si se encuentra la solución óptima en un punto extremo con coordenadas enteras, entonces el problema queda resuelto. En otro caso, se ve fácilmente que podemos cortar el conjunto solución (aumentando restricciones extras al problema) hasta eliminar este punto extremo sin excluir una solución entera . Tal restricción se llama un plano de corte.

Si los planos se escogen adecuadamente en cada etapa, entonces el poliedro inicial T se reducirá progresivamente hasta que coincida con la cubierta convexa de las soluciones enteras, al menos en una vecindad de la solución óptima.

Desafortunadamente, y esta es la dificultad esencial, no se conoce un método sistemático para generar todas las ecuaciones o desigualdades lineales que definen la cubierta convexa de los puntos enteros contenidos en un poliedro convexo dado.

b) **El Método de Ramificación y Acotamiento:** consiste esencialmente en dividir el conjunto de soluciones factibles de un problema en subconjuntos donde se buscará la solución óptima, desechando aquellos que bajo un criterio o una cota no son susceptibles de tomarse en consideración, ahorrando con esto una considerable cantidad de tiempo y esfuerzo. Actualmente las técnicas de ramificación y acotamiento se están empleando en varios campos de aplicación tales como: problemas de distribución, secuenciación de instalaciones, agente viajero,rutas de vehículos, el problema de la mochila, problemas de programación no lineal; el

progreso que se ha tenido se debe en gran parte al desarrollo que han tenido las computadoras, ya que por su gran velocidad se tiene una solución más exacta y rápida sin necesidad de recurrir a métodos aproximados.

Conviene mencionar que se dispone también de paquetes computacionales que usan ramificación y acotamiento(R-A) en sus procedimientos de solución, tales como el LINDO para resolver problemas de programación lineal y entera y el GINO para problemas no lineales. Además se están desarrollando algoritmos de R-A para computación paralela.

c) **Métodos de la Teoría de Grupos:** En un problema de programación entera los coeficientes de las desigualdades forman un conjunto finito que es cerrado bajo la suma cuando las operaciones aritméticas se toman modulo 1. Tal conjunto constituye un grupo, más aun, conforma un grupo abeliano, que además puede tener a lo más D elementos, donde D es el valor absoluto del determinante de la base actual del programa lineal y que a menudo puede generarse por un elemento en cuyo caso el grupo se llama grupo cíclico.

Un problema de grupo puede tratarse como un problema entero con una restricción y también como un problema de redes donde se desea encontrar la ruta más corta. Los algoritmos que resuelven el problema entero son usualmente del tipo de programación dinámica.

d) **Métodos de la Programación Dinámica:** Los problemas de optimización combinatoria a menudo pueden formularse como problemas de programación dinámica que no cuenta con una formulación matemática estándar para un problema, sino que es un enfoque de tipo general para la solución de problemas y las ecuaciones específicas que se usan se deben desarrollar para que representen cada situación individual. Entonces, se necesita un cierto grado de creatividad y un buen conocimiento de la estructura general de los problemas de programación dinámica para reconocer cuándo un problema se puede resolver por medio de éstos procedimientos y cómo esto se puede llevar a cabo. Las características principales de los problemas de programación dinámica son a grandes rasgos las siguientes:

1. El problema se puede dividir en etapas que requieren una política de decisión en cada una de ellas.

2. Cada etapa tiene un cierto número de estados asociados a ella.

3. El efecto de la política de decisión en cada etapa es transformar el estado actual en un estado asociado con la siguiente etapa (Tal vez de acuerdo a una distribución de probabilidad)

4. El procedimiento de solución está diseñado para encontrar una política óptima para el problema completo, es decir, una receta para las decisiones de la política óptima en cada etapa para cada uno de los estados posibles.

5. Dado el estado actual, una política óptima para las etapas restantes es independiente de la política adoptada en etapas anteriores.

6. El procedimiento de solución se inicia al encontrar la política óptima para la última etapa.

7. Se dispone de una relación recursiva que identifica la política óptima para la etapa n, dada la política para la etapa (n + 1).

8. Cuando se usa esta relación recursiva el procedimiento de solución se mueve hacia atrás etapa por etapa -encontrando cada vez la política óptima para esa etapa- hasta que se encuentra la política óptima desde la etapa inicial.

e) **Métodos heurísticos:** Dado el carácter no polinomial de una buena parte de los problemas de optimización combinatoria, se hace necesario contar con buenos métodos heurísticos, entre los cuales destacan:

Métodos constructivos. Los cuales construyen soluciones usando reglas heurísticas en forma determinística y secuencial.

Métodos de transformación local. Cuyo objetivo es perfeccionar una solución existente, mediante una búsqueda local en una vecindad bien definida del espacio de soluciones.

Métodos de descomposición. Consisten en subdividir el problema en pequeñas particiones.

Métodos inductivos. A partir de la solución de problemas pequeños se incrementan inductivamente dichas soluciones.

Métodos de extracción de semejanzas. Intentan reducir el tamaño del problema, mediante la acción combinada de planteamientos estadísticos y combinatorios. Por ejemplo, removiendo del problema rasgos comunes a varias soluciones obtenidas por métodos diferentes o diversas aplicaciones de un mismo método.

Métodos de aproximación. Transforman un problema intratable en uno tratable.

CAPITULO 2

COMPLEJIDAD COMPUTACIONAL

Las computadoras son mejores que nosotros para la aritmética; no es que sean muy buenas, es que nosotros somos bastante malos para ella.
Isaac Asimov

2.1 INTRODUCCION

Para que una computadora lleve a cabo una tarea es preciso decirle qué operaciones debe realizar, es decir, debemos describir como debe realizar la tarea. Dicha descripción se llama **algoritmo**.

Un algoritmo describe el método mediante el cual se realiza una tarea. Un algoritmo consiste en una secuencia de instrucciones, las cuales, realizadas adecuadamente, dan lugar al resultado deseado.

La noción de algoritmo no es exclusiva de la computación o de las matemáticas. Existen algoritmos que describen toda clase de procesos de la vida real, por ejemplo, las recetas de cocina, las partituras musicales, etc.

En todos los casos anteriores el ejecutor o procesador de las instrucciones que realiza la tarea correspondiente es el hombre. Sin embargo, el agente que interpreta y realiza las instrucciones de un algoritmo se llama **procesador**.

Un procesador puede ser una persona, una computadora, o cualquier otro sistema electrónico o mecánico. Un procesador realiza un proceso siguiendo, o ejecutando, el algoritmo correspondiente. La ejecución de un algoritmo requiere la ejecución de cada uno de los pasos o instrucciones que lo constituyen. De aquí se desprende que una computadora no es mas que un tipo particular de procesador.

Como se ha dicho ya, para que un procesador lleve a cabo un proceso debe estar previamente provisto de un algoritmo adecuado. Por ejemplo, el cocinero debe conocer la receta, el pianista, la partitura, etc.

Si el procesador de un algoritmo es una computadora, el algoritmo se debe expresar en forma de programa. Un programa se escribe en un lenguaje de programación, y la actividad que consiste en expresar un algoritmo en un lenguaje de programación se llama **programar**.

Cada paso del algoritmo se expresa mediante una instrucción o sentencia en el programa. Por tanto, un programa consiste en una

secuencia de instrucciones, cada una de las cuales especifica ciertas operaciones a realizar por la computadora.

Podría por tanto afirmarse que son mas importantes los algoritmos que los lenguajes de programación e incluso las mismas computadoras, considerando que un lenguaje de programación es simplemente un medio conveniente para expresar un algoritmo y una computadora es simplemente un procesador para ejecutarlo; es decir, tanto los lenguajes de programación como las computadoras son medios para lograr un fin: ejecutar un algoritmo.

Con esto no se pretende restar importancia a las computadoras y los lenguajes de programación. Los avances en la tecnología informática permiten día a día ejecutar algoritmos más rápidamente y con más precisión y a mejor precio.

Un aspecto importante es el **diseño de algoritmos**. ¿Cómo diseñar buenos algoritmos? El diseño de buenos algoritmos requiere creatividad e ingenio y no existen , en general, reglas para diseñar algoritmos. En otras palabras, no existe un algoritmo para diseñar algoritmos.

Si existen varios algoritmos para resolver un problema ¿cuál de ellos es el "mejor", en el sentido de que necesita menos recursos informáticos? ¿Cuáles son los mínimos recursos informáticos necesarios para llevar a cabo una tarea determinada? (es decir, ¿que recursos utilizará el mejor algoritmo posible para realizar dicha tarea?) ¿Se puede averiguar cuál es el mejor algoritmo? ¿Existen problemas para los cuales el mejor algoritmo posible requerirá tantos recursos que hará inviable su ejecución incluso con la computadora mas grande y rápida existente?

Todas estas cuestiones se engloban para su estudio bajo el título: **Complejidad de Algoritmos.**

2.2 COMPLEJIDAD DE ALGORITMOS: TIEMPO Y ESPACIO

Como sabemos, un programa es una representación de un algoritmo en un lenguaje de programación que puede interpretar y ejecutar una computadora.

La manera de representar un algoritmo en forma de programa no es en general única. Asimismo, para resolver un problema para el cual existe un algoritmo, no es único el algoritmo que lo resuelve. Cabe por tanto preguntarse cuál es el mejor algoritmo de entre dos algoritmos dados que resuelven el mismo problema.

Una posible alternativa para contestar dicha cuestión consiste en representar los dos algoritmos mediante un lenguaje de programación, a continuación ejecutarlos en una computadora y medir el tiempo requerido por cada uno de ellos para obtener la solución de un mismo problema particular.

El **tiempo** de ejecución requerido por un algoritmo para resolver un problema es uno de los parámetros importantes en la práctica para medir la bondad de un algoritmo pues, entre otros factores, el tiempo de ejecución equivale a tiempo de utilización de la computadora y, en consecuencia, coste económico.

Es más, si el tiempo de ejecución es demasiado grande, puede suceder que el algoritmo sea en la práctica inútil, pues el tiempo necesario para su ejecución puede sobrepasar el tiempo disponible de la computadora.

Resulta evidente que el tiempo real requerido por una computadora para ejecutar un algoritmo es directamente proporcional al número de operaciones básicas elementales que la misma debe realizar en su ejecución, medir por tanto el tiempo real de ejecución equivale a medir el número de operaciones elementales realizadas. (Nosotros supondremos desde ahora que todas las operaciones básicas se ejecutan en una unidad de tiempo. Para una mayor precisión habría que distinguir los tiempos de ejecución de cada una de las distintas operaciones elementales). Por esta razón se suele llamar **tiempo** de ejecución no al tiempo real físico, sino al número de operaciones elementales realizadas.

Otro de los factores importantes, en ocasiones decisivo, para comparar algoritmos es la cantidad de memoria de máquina requerida para almacenar los datos durante el proceso.

La cantidad de memoria utilizada por un algoritmo durante el proceso se suele llamar **espacio** requerido por el algoritmo.

Para entender la diferencia del espacio requerido por dos algoritmos que resuelven el mismo problema veamos un ejemplo:

Consideremos el siguiente problema:

Ejemplo 2.1

Dados n números naturales en forma secuencial, es decir, dados de uno en uno con un intervalo de tiempo entre uno y otro, encuentre cuál es el máximo de todos ellos.

Veamos entonces dos algoritmos que resuelven el mismo problema:

ALGORITMO 1

PASO 1. Almacenar los n números utilizando n variables $a_1,\ldots,a_n$

PASO 2. Asignar $m \leftarrow a_1$, $i \leftarrow 2$

PASO 3. Si $m > a_i$ entonces $i \leftarrow i + 1$ en caso contrario $m \leftarrow a_i$, $i \leftarrow i+1$

PASO 4. Si $i > n$ el máximo es m; FIN. En caso contrario volver al paso 3.

ALGORITMO 2

PASO 1. Asignar el primer elemento a la variable m.

PASO 2. Asignar el siguiente elemento a la variable a.

PASO 3. Si a > m entonces m ← a.

PASO 4. Mientras queden elementos volver al paso 2.

PASO 5. El máximo es m FIN.

Los dos algoritmos anteriores resuelven el problema del máximo. Sin embargo, el espacio requerido por el primero de ellos depende del valor de n. Cuanto mayor sea n mayor es el número de variables a las que hay que asignar valores en el paso 1 y, en consecuencia, mayor será la cantidad de espacio necesario.

El algoritmo 2 por el contrario utiliza sólo las variables m y a independientemente de la cantidad de números, pues este segundo algoritmo antes de recibir un nuevo número calcula el máximo de los números anteriores manteniéndolo en la variable m.

Está claro que para que el segundo algoritmo puede ejecutarse, el intervalo de tiempo que transcurre entre la entrada de dos números debe ser mayor o a lo sumo igual al tiempo requerido para efectuar las operaciones del paso 3.

En adelante nos ocuparemos solamente del **tiempo** requerido por un algoritmo para su ejecución.

Si observamos la fórmula propuesta para medir el tiempo requerido por un algoritmo, observamos que, al no ser única la manera de representarlo mediante un programa, y al no ser única tampoco la computadora en donde se ejecuta, resulta que dicha medida será variable dependiendo fundamentalmente de los siguientes factores:

1. El lenguaje de programación elegido

2. El programa que representa el algoritmo.

3. La computadora que lo ejecuta.

En definitiva, al existir gran cantidad de lenguajes, técnicas de programación y computadoras diferentes, resulta que la forma propuesta de medir el tiempo, y en consecuencia la bondad de un algoritmo, no resulta adecuada.

Por ésta razón surge la necesidad de medir el tiempo requerido por un algoritmo independientemente de su representación y del procesador que lo ejecute.

Por otra parte, si la cantidad de datos del problema a resolver es pequeña, prácticamente cualquier algoritmo que lo resuelva lo hará en un espacio corto de tiempo, siendo, en este caso, prácticamente indiferente el elegir uno u otro algoritmo para resolverlo.

Cuando aparece la dificultad en cuanto al tiempo requerido por un algoritmo es cuando el volumen de datos del problema a resolver aumenta.

Cuando el volumen de datos es suficientemente grande es cuando un algoritmo puede realmente aventajar a otro para resolver el mismo problema.

Veamos un ejemplo:

Ejemplo 2.2

Dada una lista $\{a_1, a_2, \ldots, a_n\}$ de números ordenados de menor a mayor, verifique si hay algún número repetido.

Los siguientes algoritmos resuelven el problema:

ALGORITMO 3

PASO 1. $i \leftarrow 1$, $j \leftarrow 2$

PASO 2. Si $a_i = a_j$ entonces la respuesta es SI.FIN

PASO 3. Si $j < n$ entonces $i \leftarrow i + 1$; $j \leftarrow j+1$ y volver al paso 2.

PASO 4. La respuesta es NO. FIN

ALGORITMO 4

PASO 1. $i \leftarrow 1$; $j \leftarrow 2$

PASO 2. Si $a_i = a_j$ entonces la respuesta es SI. FIN

PASO 3. Si $j < n$ entonces $j \leftarrow j + 1$ y volver al paso 2

PASO 4. Si $i < n - 1$ entonces $i \leftarrow i + 1$; $j \leftarrow i + 1$ y volver al paso 2.

PASO 5. La respuesta es NO. FIN.

Si contamos sólo las comparaciones que se efectúan en el paso 2 de ambos algoritmos, observamos que el primero compara cada número de la lista con el inmediatamente posterior, por lo que efectúa n-1 comparaciones como máximo (el peor caso será cuando no haya repeticiones o bien cuando estén repetidos los dos últimos números solamente); mientras que en el segundo algoritmo puede suceder el caso en que compare cada número con todos los que le siguen, por lo que el número de comparaciones puede llegar a ser $(n-1) + (n-2) + (n-3) + \ldots + 2 + 1$, es decir $n^2/2 - n/2$.

Teniendo en cuenta la tabla siguiente en la que se observan los valores que toman $n - 1$ y $n^2/2 - n/2$ para distintos valores de n

n	n - 1	$n^2/2 - n/2$
2	1	1
10	9	45
100	99	4950
1000	999	499500
100000	99999	49995000
1000000	999999	4999950000

se deduce inmediatamente que el algoritmo 3 es claramente mas ventajoso que el 4 (especialmente cuando el valor de n es grande)

2.3 PEOR CASO Y CASO PROBABILISTICO.

En el estudio anterior del tiempo de ejecución de los algoritmos hemos analizado el número de comparaciones que podrían llegar a realizarse en algún caso extremo. A estos casos en que el tiempo es el mayor posible de entre todos los casos que se pueden presentar se les llama **el peor caso.**

En ocasiones el peor caso se presenta a menudo al resolver un problema y en otras se presenta con poca frecuencia. Por esta razón tiene interés el estudio del tiempo de ejecución de un algoritmo en el caso medio o caso probabilistico. Este estudio entra dentro del campo del cálculo de probabilidades y de la estadística y, siendo

de gran interés en la evaluación de los algoritmos, resulta en ocasiones sumamente complejo.

En adelante el análisis que hagamos de los algoritmos será siempre estudiando el comportamiento del mismo en el peor caso.

A pesar de la ventaja que esto supone, en ocasiones un algoritmo cuyo comportamiento en el peor caso es desastroso puede ser útil en una gran cantidad de casos si se presenta el peor caso con una frecuencia muy pequeña.

El análisis de algoritmos se encarga del estudio del **tiempo y espacio** requerido por un algoritmo para su ejecución. Ambos parámetros, como hemos visto, pueden ser estudiados respecto del **peor caso** (o caso general) o respecto del caso probabilístico (o caso esperado). En la práctica, casi siempre es más difícil determinar el tiempo de ejecución promedio que el del peor caso, pues el análisis se hace intratable en matemáticas. Así pues, se utilizará el tiempo de ejecución del peor caso como medida principal de la complejidad del tiempo, aunque se mencionará la complejidad del caso promedio cuando pueda hacerse en forma significativa.

Tanto el tiempo como el espacio de un algoritmo son parámetros que, en general, dependen del tamaño n de los datos de entrada del algoritmo. En consecuencia son funciones T(n) y E(n) enteras de variable entera.

En general no resulta sencillo determinar el valor exacto de la función tiempo T(n) de un algoritmo. Para poder hacerlo es preciso conocer con exactitud cuáles y cuántas veces se realizan las operaciones básicas cuya ejecución requiere un tiempo constante conocido.

En definitiva, lo que aparece al analizar un algoritmo es un problema de combinatoria. No obstante, como hemos dicho, no siempre resulta fácil hacer el cálculo exacto del número de operaciones requeridas por un algoritmo. Por esta razón es por la que en muchas ocasiones el análisis de algoritmos se reduce a estudiar el **comportamiento** de la función tiempo T(n) y no cuál es su valor exacto.

2.4 ANALISIS ASINTOTICO DE FUNCIONES.

En ésta sección el objeto de estudio serán las funciones reales de variable natural $f: N \rightarrow R$. La idea fundamental consiste en comparar funciones de este tipo para poder decir cuál tiene mejor comportamiento asintótico, es decir, cuál es menor cuando la variable independiente es suficientemente grande.

Si sabemos hacer esto con dos funciones podremos utilizar las funciones de tiempo de dos algoritmos y así determinar cuál de

ellos tiene mejor comportamiento asintótico.

El problema que en ocasiones resulta complicado es el de hallar explícitamente la función tiempo de un algoritmo. El análisis asintótico que haremos permitirá, en ocasiones, conocer cómo se comporta una función aún sin conocer ésta.

DEFINICION 2.1 Si f y g son dos funciones de N en R, se dice que g domina asintóticamente a f (o simplemente que g domina a f) si existen enteros $k \geq 0$ y $m \geq 0$ tales que se verifica la desigualdad: $|f(n)| \leq k\,|g(n)|$ para todo entero $n \geq m$
Observe que si g domina a f y $g(n) \neq 0$, entonces $|f(n)/g(n)| \leq k$ para casi todos los enteros n (es decir, para todos salvo una cantidad finita). En concreto, para todos los valores $n \geq m$ se verifica dicha desigualdad.

En ésta situación, si f y g son funciones de tiempo de dos algoritmos F y G respectivamente, resulta que el algoritmo F nunca tardará mas de k veces el tiempo que tarda el algoritmo G en resolver un problema del mismo tamaño.

Por ejemplo si $f(n) = n$ y $g(n) = n^3$, se verifica que g domina a f: En efecto, si $m = 0$ y $k = 1$ se verifica que para todo $n \geq m$ se cumple la desigualdad

$$|n| \leq k|n^3|$$

en este caso, $|n| \leq |n^3|$ ya que $k = 1$ pues el cubo de un número natural es siempre mayor o igual que dicho número.

La definición de dominación establece una relación binaria en el conjunto de las funciones de N en R. El siguiente teorema da propiedades de dicha relación.

TEOREMA 2.1 La relación de dominación < definida por

$$f < g \leftrightarrow g \text{ domina a } f$$

es una relación reflexiva y transitiva. La demostración de este teorema queda de ejercicio, así mismo proponemos como ejercicio el comprobar que dicha relación no es simétrica, es decir, si g domina a f, no se verifica necesariamente que f domina a g. En consecuencia, la relación de dominación no es una relación de equivalencia.

Por otra parte, esta relación tampoco es relación de orden pues no verifica la propiedad antisimétrica (compruébelo también como ejercicio).

Si f es una función de N en R denotaremos por O(f) al conjunto de todas las funciones dominadas por f.

Con notación matemática:

$$O(f) = \{ g \mid g : N \to R \text{ es aplicación y } g < f\}$$

Si una función g pertenece al conjunto O(f) se suele decir que g es una o mayúscula de F o que **g es de orden f.**

Por ejemplo, decir que el tiempo de ejecución de un programa T(n) es $O(n^2)$, significa que existen constantes enteras positivas m y k tales que para $n \geq m$ se tiene $T(n) \leq kn^2$

Ejemplo 2.3

Suponga que T(0) = 1, T(1) = 4 y en general $T(n) = (n + 1)^2$. Entonces se observa que T(n) es $O(n^2)$ cuando **m=1** y k = 4, es decir para $n \geq 1$, se tiene que $(n + 1)^2 \leq 4n^2$ que es fácil de demostrar. Observe que no se puede hacer m = 0 pues T(0) = 1 no es menor que $k0^2 = 0$ para ninguna constante k.

Se dice que T(n) es O(f(n) si existen m y k tales que $T(n) \leq kf(n)$ cuando $n \geq m$. Cuando el tiempo de ejecución de un programa es O(f(n)) se dice que tiene velocidad de crecimiento f(n)

TEOREMA 2.2 Se verifican las siguientes propiedades de los conjuntos O(f):

1. $f \in O(f)$
2. Si $f \in O(g)$ entonces $cf \in O(g)$ para todo $c \in R^+$
3. Si $f \in O(g)$ y $h \in O(g)$ entonces $f + g \in O(g)$
4. $f \in O(g)$ si y sólo si $O(f) \subset O(g)$
5. Si $f \in O(g)$ y $g \in O(f)$ entonces $O(f) = O(g)$
6. Si $f \in O(g)$ y $g \in O(h)$ entonces $f \in O(h)$

Algunos conjuntos O(f) que aparecen con frecuencia al analizar algoritmos son:

$$O(1),\ O(\log n),\ O(n),\ O(n\log n),\ O(n^2),\ldots,O(n^p),\ldots,O(c^n),\ O(n!)$$

Si la función tiempo de un algoritmo es de orden 1 se dice que dicho algoritmo tiene complejidad **constante**, si es de orden log n se dice que es **logarítmica**, si es de orden n se dice que es **lineal**, si es de orden n^p, siendo p un número natural, se dice que es **polinómica**, si es de orden c^n con $c > 1$ se dice que es **exponencial** y si es de orden n! se dice que es **factorial.**

Observación: Mientras no se diga lo contrario, se entenderá que los logaritmos que aparecen son en base 2.

Ejemplo 2.4

La función $T(n) = 3n^3 + 2n^2$ es $O(n^3)$. Para comprobar esto, sean $m = 0$ y $k = 5$ entonces para $n \geq 0$, $3n^3 + 2n^2 \leq 5n^3$. También se podría decir que $T(n)$ es $O(n^4)$ pero sería una afirmación mas débil que decir que $T(n)$ es $O(n^3)$

Ejemplo 2.5

La función 3^n no es $O(2^n)$

demostración:

Suponga que existen m y k tales que para toda $n \geq m$, se tiene $3^n \leq k\ 2^n$, entonces $k \geq (3/2)^n$ para cualquier $n \geq m$, pero $(3/2)^n$ se hace arbitrariamente grande conforme n crece y por lo tanto, ninguna constante k puede ser mayor que $(3/2)^n$ para toda n.

Cuando se dice que $T(n)$ es $O(f(n))$ se sabe que $f(n)$ es una cota superior para la velocidad de crecimiento de $T(n)$. Para especificar una cota inferior para la velocidad de crecimiento de $T(n)$ se usa la notación $\Omega\ (g(n))$ que se lee "$T(n)$ es omega de $g(n)$" lo cual significa que existe una constante c tal que $T(n) \geq cg(n)$ para un número infinito de valores de n.

Ejemplo 2.6

Para verificar que la función $T(n) = n^3 + 2n^2$ es $\Omega(n^3)$, sea $c = 1$, entonces $T(n) \geq cn^3$ para $n = 0, 1, \ldots$

TEOREMA 2.3 Dadas las funciones de tiempo, se verifican las siguientes contenciones:

$$O(1) \subset O(\log n) \subset O(n) \subset O(n\log n) \subset O(n^2) \subset O(c^n) \subset O(n!)$$

siendo $c > 1$.

Además dichas contenciones son propias, en ninguna de ellas se da la igualdad.
demostración:

a) $O(1) \subset O(\log n)$, en efecto si $m = 2$ y $k = 1$ se verifica que para todo $n > 2$ es $1 \leq \log n$ por lo que la función constante 1 pertenece a $O(\log n)$, por lo tanto por el apartado 4 del teorema 2.2, $O(1) \subset O(\log n)$.

Además la contención es propia: si $O(\log n)$ estuviera contenido en $O(1)$ se verificaría que $\log n$ pertenecería a $O(1)$, y por lo tanto, $\log n$ estaría dominada por la función constante 1.

Esto significa que existen $m \geq 0$ y $k \geq 0$ tales que para todo $n > m$ sería

$$|\log n| \leq k\ |1|$$

es decir

$$\log n \leq k$$

lo cual es una contradicción, pues como

$$\lim_{n \to \infty} \log n = \infty$$

la función log n no puede estar acotada por una constante k.

b) $O(\log n) \subset O(n)$: Considerando $m = 0$ y $k = 1$ se verifica que para todo $n > m$ es

$$|\log n| \leq k|n|$$

ya que para todo $n > 0$ es $\log n < n$. Por lo tanto $\log n \in O(n)$ y por la misma razón de antes $O(\log n) \subset O(n)$

c) $O(n) \subset O(n \log n)$: si $n > 2$ se verifica que $\log n > 1$y, por tanto, $n\log n > 1.n = n$; por lo que tomando $m = 2$ y $k = 1$ se verifica que para todo $n > m$ es

$$|n| \leq k\ |n \log n|$$

luego $n \in O(n \log n)$ y por lo tanto $O(n) \subset O(n \log n)$.

d) $O(n \log n) \subset O(n^2)$: Como $\log n < n$ si $n > 0$ resulta que $n.\log n < n.n = n^2$. Por lo tanto, considerando $m = 0$ y $k = 1$ se verifica que para todo $n > m$ es

$$|n \log n| < k\ |n^2|$$

luego $n \log n \in O(n^2)$ y por lo tanto, $O(n \log n) \subset O(n^2)$

e) $O(n^2) \subset O(c^n)$ (si $c > 1$): Como en casos anteriores, es suficiente probar que para una n suficientemente grande se verifica la desigualdad

$$n^2 < c^n$$

lo que equivale, tomando logaritmos (tenga en cuenta que como $c > 1$ es $\log c > 0$), a la igualdad

$$\log n^2 < \log c^n$$

o bien

$$(\log n) / n < (\log c)/2$$

como

$$\lim_{n \to \infty} \frac{\log n}{n} = 0$$

y (log c)/2 es una constante positiva, resulta que existe un valor m tal que para todo valor n > m es

$$(\log n)/n < (\log c)/2$$

como queríamos probar.

f) $O(c^n)$ c $O(n!)$: Sean k = [c] y m = max {$[c^k]$, k} (donde [c] representa al menor entero mayor que c)

Entonces, si n > m se verifica

$$n! = n\ (n-1)\ (n-2)\ \ldots (k+1)k(k-1)\ldots 2\ 1$$

y como k ≥ [c] y [c] ≥ c, se verifica

$$(n-1)(n-2)\ldots(k+1)k \geq c^{n-k}$$

Por otra parte, como $n > c^k$, resulta

$$n! > c^k\ c^{n-k} = c^n$$

En consecuencia $n! > c^n$ si n > m y por lo tanto $O(c^n)$ c $O(n!)$ como antes.

TEOREMA 2.4 Si c es una constante, se verifica:

a) $\Sigma\ c \in O(n)$

b) $\Sigma\ i \in O(n^2)$

c) $\Sigma\ i^2 \in O(n^3)$

La demostración se deja como ejercicio.

2.5 VELOCIDAD DE CRECIMIENTO Y CALCULO DE TIEMPO DE EJECUCION DE UN PROGRAMA.

Partiremos del supuesto de que es posible evaluar programas comparando sus funciones de tiempo de ejecución sin considerar las constantes de proporcionalidad.. Es decir, un programa con tiempo de ejecución $O(n^2)$ es mejor que uno con tiempo de ejecución $O(n^3)$, sin embargo, además de los factores constantes debidos al compilador y la máquina, existe un factor constante debido a la naturaleza del programa mismo. Es posible, que con una combinación determinada de compilador y máquina, el primer programa tarde $100n^2$ milisegundos, mientras el segundo tarda $5n^3$ milisegundos. En éste caso ¿no es preferible el segundo programa al primero?.

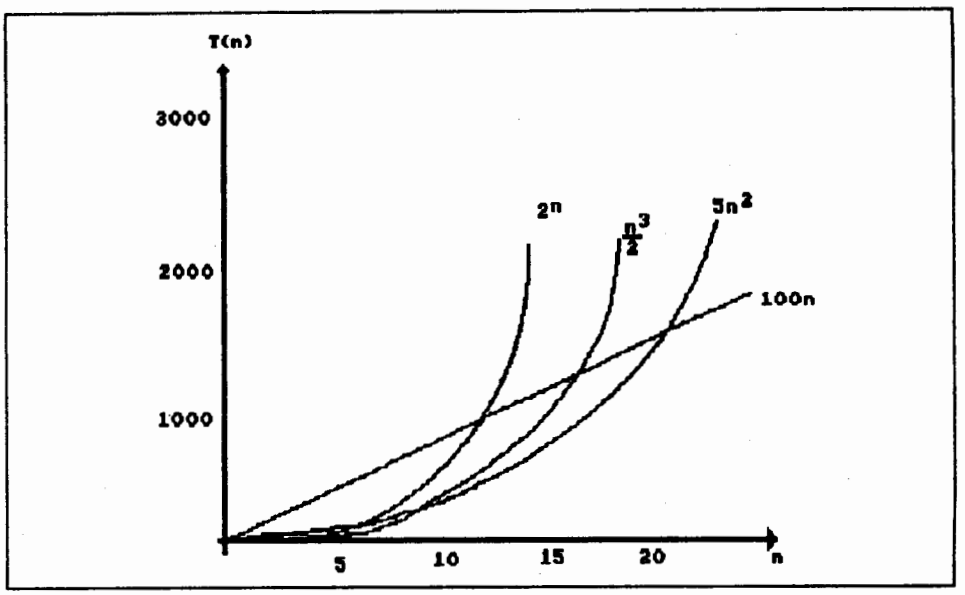
La respuesta a esto depende del tamaño de las entradas que se espera que procesen los programas. Para entradas de tamaño $n < 20$, el programa con tiempo de ejecución $5n^3$ será mas rápido que el de tiempo de ejecución $100n^2$. Así pues, si el programa se va a ejecutar con entradas pequeñas, será preferible el programa cuyo tiempo de ejecución es $O(n^3)$. No obstante, conforme n crece, la razón de los tiempos de ejecución que es $5n^3/100n^2 = n/20$, se hace arbitrariamente grande. Así, a medida que crece el tamaño de la entrada, el programa $O(n^3)$ requiere un tiempo significativamente mayor que $O(n^2)$. Pero si hay algunas entradas grandes en los problemas para cuya solución se están diseñando estos dos programas, será mejor optar por el programa cuyo tiempo de ejecución tiene la menor velocidad de crecimiento.

Ejemplo 2.7

En la figura 2.1 se muestran gráficamente los tiempos de ejecución de cuatro programas de distintas complejidades de tiempo, medidas en segundos, para una combinación determinada de compilador y máquina

Suponga que se dispone de 1000 segundos o alrededor de 17 minutos para resolver un problema determinado ¿Que tamaño de problema se puede resolver? Considerando estos tiempos, en la segunda columna de la siguiente tabla se muestra que los cuatro algoritmos pueden resolver problemas de un tamaño similar en 10^3 segundos. Si se adquiere una máquina que funciona diez veces mas rápido sin costo adicional, entonces es posible dedicar 10^4 segundos a la solución de problemas que antes requerían 10^3 segundos. El tamaño máximo de problema que es posible resolver ahora con los cuatro algoritmos se muestra en la tercer columna de la tabla, y la razón entre los valores de la segunda y tercer columnas se muestran en la cuarta.

Se observa que un aumento de 1000% en la velocidad de la computadora origina apenas un incremento del 30% en el tamaño del



problema que se puede resolver con el programa $O(2^n)$. Los aumentos adicionales de un factor de diez en la rapidez de la computadora a partir de este punto originan aumentos porcentuales aún menores en el tamaño de los problemas. De hecho, el programa $O(2^n)$ sólo puede resolver problemas pequeños, independientemente de la rapidez de la computadora.

Tiempo de ejecución T(n)	Tamaño máximo de problema para 10^3 segundos	Tamaño máximo de problema para 10^3 segundos	Incremento en el tamaño max. de problema
100n	10	100	10.0
$5n^2$	14	45	3.2
$n^3/2$	12	27	2.3
2^n	10	13	1.3

En la tercer columna de la tabla se puede apreciar una superioridad evidente del programa O(n), éste permite un aumento de 1000% en la rapidez de la computadora. Y se observa que los programas $O(n^3)$ y $O(n^2)$ permiten aumentos de 230% y 320% respectivamente en el tamaño de problema, para un incremento del 1000% en la rapidez de la computadora. Estas razones se mantendrán vigentes para incrementos adicionales en la rapidez de la computadora.

A medida que la computadoras aumenten su rapidez y disminuyan su precio, también el deseo de resolver problemas mas grandes y complejos seguirá creciendo. Así la importancia del descubrimiento y el empleo de algoritmos eficientes (cuyas velocidades de crecimiento sean pequeñas) irá en aumento.

OBSERVACIONES.

1. La velocidad de crecimiento del tiempo de ejecución del peor caso no es el único criterio, ni necesariamente el más importante para evaluar un algoritmo.

2. De acuerdo con el punto anterior, si un programa sólo se va a usar algunas veces, el costo de su escritura y depuración es el dominante, de manera que el tiempo de ejecución raramente influirá en el costo total. En tal caso debe elegirse el algoritmo que sea más fácil de aplicar correctamente.

3. Un algoritmo eficiente pero complicado puede no ser apropiado porque posteriormente puede suceder que tenga que darle mantenimiento otra persona distinta del escritor. Se espera que al difundir el conocimiento de las principales técnicas de diseño de algoritmos eficientes, se podrán utilizar libremente algoritmos más complejos, pero debe considerarse la posibilidad de que un programa resulte inútil debido a que nadie entiende sus sutiles y eficientes algoritmos.

4. Existen ejemplos de algoritmos eficientes que ocupan mucho espacio para poder aplicarlos sin un almacenamiento secundario lento, lo cual puede anular la eficiencia.
5. En los algoritmos numéricos, la precisión y la estabilidad son tan importantes como la eficiencia.

TIEMPO DE EJECUCION DE UN PROGRAMA

Comenzaremos con algunas reglas básicas de suma y producto en notación asintótica:

Sean $T_1(n)$ y $T_2(n)$ los tiempos de ejecución de dos fragmentos P_1 y P_2 de un programa y que $T_1(n)$ es $O(f(n))$ y $T_2(n)$ es $O(g(n))$. Entonces

$$T_1(n) + T_2(n)$$

el tiempo de ejecución de P_1 seguido de P_2 es $O(\text{máx } [f(n), g(n)])$. Esto se puede verificar observando que para algunas constantes, c_1, c_2, n_1, n_2 si $n \geq n_1$, entonces

$$T_1(n) \leq c_1 f(n)$$

y si $n \geq n_2$ entonces

$$T_2(n) \leq c_2\, g(n)$$

Sea $m = \max(n_1, n_2)$, si $n \geq m$ entonces

$$T_1(n) + T_2(n) \leq c_1\, f(n) + c_2\, g(n)$$

De aquí se concluye si $n \geq m$ entonces

$$T_1(n) + T_2(n) \leq (c_1 + c_2)\,\text{máx}\,[f(n), g(n)]$$

por lo tanto $T_1(n) + T_2(n)$ es $O(\text{máx}\,[f(n), g(n)])$.

Ejemplo 2.8

La regla de la suma anterior se puede usar para calcular el tiempo de ejecución de una secuencia de pasos de programa, donde cada paso puede ser fragmento de un programa arbitrario con ciclos y ramificaciones. Suponga que se tienen tres pasos cuyos tiempos de ejecución son respectivamente, $O(n^2)$, $O(n^3)$ y $O(n \log n)$. Entonces el tiempo de ejecución de los dos primeros pasos ejecutados en secuencia es $O(\max(n^2, n^3))$ que es $O(n^3)$, el tiempo de ejecución de los tres juntos es $O(\max(n^3, n\log n))$ que es $O(n^3)$.

En general el tiempo de ejecución de una secuencia fija de pasos, dentro de un factor constante, es igual al tiempo de ejecución del paso con mayor tiempo de ejecución. En raras ocasiones dos pasos pueden tener tiempos de ejecución inconmensurables (ninguno es mayor que el otro, ni son iguales). Por ejemplo, puede haber pasos con tiempo de ejecución $O(f(n))$ y $O(g(n))$, donde

$$f(n) = \begin{cases} n^4 \text{ si } n \text{ es par} \\ n^2 \text{ si } n \text{ es impar} \end{cases} \qquad g(n) = \begin{cases} n^2 \text{ si } n \text{ es par} \\ n^3 \text{ si } n \text{ es impar} \end{cases}$$

En tales casos, la regla de la suma debe aplicarse directamente, en el ejemplo, el tiempo de ejecución es $O(\text{máx}\,[f(n), g(n)])$, esto es n^4 si n es par y n^3 si n es impar.

Otra observación útil sobre la regla de la suma es que si $g(n) \leq f(n)$ para toda $n \geq m$ entonces $O(f(n) + g(n))$ es lo mismo que $O(f(n))$. Por ejemplo $O(n^2 + n)$ es lo mismo que $O(n^2)$

La regla del producto es la siguiente: Si $T_1(n)$ y $T_2(n)$ son $O(f(n))$ y $O(g(n)$, respectivamente, entonces $T_1(n)T_2(n)$ es $O(f(n)g(n))$. Su demostración se deja como ejercicio. Según la regla del producto, $O(cf(n))$ significa lo mismo que $O(f(n))$ si c es una constante positiva cualquiera. Por ejemplo $O(n^2/2)$ es lo mismo que $O(n^2)$

Una vez que se han estudiado las herramientas necesarias para analizar el comportamiento de la función tiempo de un algoritmo, se hará el análisis de algunos algoritmos

Ejemplo 2.9

Encuentre el máximo de un conjunto de n números naturales [a_1, a_2,...,a_n]

Solución

ALGORITMO MAXIMO SECUENCIAL

Entrada: La lista L = { a_1,...,a_n}

PASO 1 $m \leftarrow a_1$ $i \leftarrow 2$

PASO 2 Si $m < a_i$ entonces $m \leftarrow a_i$

PASO 3 $i \leftarrow i + 1$

PASO 4 Si $i > n$ FIN; en caso contrario volver al paso 2

Salida: el máximo es m

Análisis del Algoritmo:

En el paso 1 se realizan dos operaciones, en el paso 2, en el peor caso, otras dos, en el paso 3 una y en el paso 4, en el peor caso 2. Por lo tanto, cada vez que del paso 4 volvemos al paso 2 serán necesarias 5 operaciones; como, tras aplicar el paso 1, el ciclo 2-4 se ejecuta n-1 veces (hasta que i toma el valor n+1), el número total de operaciones que requiere este algoritmo en el peor caso será:

$$T(n) = 2 + 5(n-1)$$

y el comportamiento asintótico queda dado por la expresión:

$$T(n) \in O(n)$$

Comparamos éste algoritmo con el siguiente algoritmo recursivo

ALGORITMO MAXIMO BINARIO

Entrada: La lista L = { $a_1,\ldots,a_i, a_{i+1},\ldots,a_j$} con n elementos

PASO 0 Si i = j entonces m ← a_1 e ir al paso 5

PASO 1 k ← [i+j/2] (donde [ ] indica parte entera de i+j/2)

PASO 2 L_1 ← { $a_i,\ldots,a_k$}, L_2 ← {$a_{k+1},\ldots,a_j$}

PASO 3 m_1 ← MAXIMO BINARIO (L_1), m_2 ← MAXIMO BINARIO (L_2)

PASO 4 Si $m_1 > m_2$ entonces m ← m_1 . En caso contrario m ← m_2

PASO 5 FIN
Salida: el máximo es m

Análisis del Algoritmo:

Si llamamos T(n) al tiempo que requiere el algoritmo para hallar el máximo de una lista de n números, se verifica que el tiempo requerido para ejecutar el paso 3 será 2 T(n/2) (cuando n sea potencia de 2). Como el número de operaciones que requieren los pasos 0, 1, 2, 4 y 5 es constante se verificará que

$$T(n) \leq 2\ T(n/2) + c$$

donde c es una constante. Como t(1) es constante, podemos suponer que $T(1) \leq c$ por lo que se verifica que $T(n) \in O(n)$.

De esto último se concluye que el comportamiento asintótico de ambos algoritmos es el mismo. Observe que en el análisis asintótico de éste segundo algoritmo no fue necesario obtener la función de tiempo del mismo.

2.5 NOTAS HISTORICAS

En algún lugar, entre 400 y 300 a.n.e. el gran matemático griego Euclides inventó un algoritmo para encontrar el máximo común divisor (m.c.d.) entre dos números naturales. El m.c.d. de X y Y es el mayor entero que divide a ambos. Por ejemplo, el m.c.d. de 80 y 32 es 16. Los detalles del algoritmo no nos interesan por el momento, pero el algoritmo de Euclides se considera el primer algoritmo no trivial desarrollado. La palabra **algoritmo** se deriva del nombre del matemático persa Mohammed Al-Khowârizmî quien vivió durante el siglo noveno y quién tiene el mérito de haber elaborado paso a paso reglas para sumar, restar, multiplicar y dividir números decimales ordinarios. El nombre de éste matemático en latín se convirtió en Algorismus, de donde declinó en algoritmo. Claramente Euclides y al- Khowârizmî fueron algorítmicos por excelencia.

CAPITULO 3

METODOS DE PLANOS DE CORTE

"A partir del momento en que escribí esa página vi claramente que mi búsqueda de la exactitud se bifurcaba en dos direcciones. Por una parte la reducción de los acontecimientos contingentes a esquemas abstractos con los que se pueden efectuar operaciones y demostrar teoremas; y por otra, el esfuerzo de las palabras por expresar con la mayor precisión posible el aspecto sensible de las cosas"
Italo Calvino. Seis Propuestas para el Próximo Milenio

Como vimos en el capítulo 1 existen varios métodos para resolver problemas de programación entera, algunos de ellos se enmarcan dentro de las técnicas de planos de corte. Estos tiene una razón histórica de estudio ya que fueron los primeros en usarse para abrir la brecha de la programación entera y sirvieron para cimentar el camino que más adelante rompería con muchas barreras existentes cuando fueron publicados. Este capítulo se desarrolla como sigue: Primero se plantea en términos generales el método, en la siguiente sección se ve el algoritmo de Gomory para problemas mixtos y puros.

El objetivo general de los algoritmos de planos de corte consiste en deducir desigualdades suplementarias a partir de las restricciones de variable entera que eventualmente producen un programa lineal cuya solución óptima es entera.

3.1 METODOS DE PLANOS DE CORTE

Los métodos de planos de corte se aplican a problemas de la forma general

$$\begin{aligned} &\text{minimizar } c^t x \\ &\text{sujeto a} \\ &\quad x \in S \end{aligned} \qquad (3.1)$$

donde $S \subset E^n$ es un conjunto convexo. Los problema que incluyen minimización de una función convexa sobre un conjunto convexo, como

$$\begin{aligned} &\text{minimizar } f(y) \\ &\text{sujeto a} \\ &\quad y \in R \end{aligned} \qquad (3.2)$$

donde $R \subset E^{n-1}$ es un conjunto convexo y f, una función convexa, se puede convertir fácilmente a la forma (3.1) escribiendo (3.2) de manera equivalente como

$$\begin{aligned} &\text{minimizar } r \\ &\text{sujeto a} \\ &f(y) - r \leq 0 \end{aligned}$$

lo cual con x = (r, y) es un caso especial de (3.1)

FORMA GENERAL DEL ALGORITMO

La forma general de un algoritmo de plano cortante para el problema (3.1) es la siguiente:

Dado un politope P_k S

PASO 1 Minimice $c'x$ sobre P_k, y obtenga un punto x_k en P_k. Si $x_k \epsilon S$, se para; x_k es el óptimo, en otro caso vaya al paso 2

PASO 2 Encuentre un hiperplano H_k separando el punto x_k de S, esto es, encuentre $a_k \epsilon E^n$, $b_k \epsilon E^1$ tales que

$$S \subset \{ x \mid a_k' x \leq b_k \}$$

$$x_k \epsilon \{ x \mid a_k' x > b_k \}$$

Actualice P_k para obtener P_{k+1} incluyendo la restricción $a_k' x \leq b_k$

Este proceso se ilustra en la figura 3.1

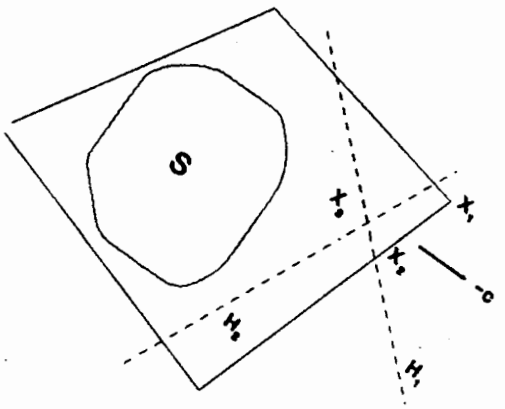


Figura 3.1

Los algoritmos específicos difieren principalmente en la manera de seleccionar el hiperplano que separa el punto actual x_t del conjunto de restricciones S. Por supuesto, esta selección es el aspecto más importante del algoritmo, pues es la profundidad del corte asociado al hiperplano de separación, la distancia del hiperplano al punto actual, que determina cuánta mejora hay en la aproximación al conjunto de restricción y, por tanto, la rapidez de convergencia del método.

Los algoritmos específicos también difieren algo respecto a la manera de actualizar el politope una vez determinado el hiperplano nuevo. El procedimiento más directo consiste simplemente en añadir la desigualdad lineal asociada a ese hiperplano a aquellas determinadas antes. Esto proporciona la mejor aproximación actualizada posible al conjunto restricción, pero tiende a producir, después de un gran número de iteraciones, una cantidad enorme de desigualdades que expresan la aproximación. Así, en algunos algoritmos, las desigualdades primitivas, que no son obligatorias en el punto actual, dejan de tenerse en consideración.

La desventaja de los métodos de planos de corte, como se verá mas adelante es que resultan muy ineficientes para resolver problemas enteros de tamaño medio. Estos métodos generan en cada iteración una restricción y una variable extra. Sin embargo, su ventaja es que ilustran lo que se pretende hacer con la región de factibilidad del problema entero para lograr su solución.

DUALIDAD

El algoritmo general de plano de corte se puede considerar como una aplicación ampliada de la dualidad en programación lineal, y aunque este punto de no es especialmente útil en el análisis del método, revela la conexión básica entre los métodos de planos de corte y dual. La base de éste punto de vista es el hecho de que S puede escribirse como la intersección de todos los semiespacios que lo contienen; así

$$S = \{ x \mid a_i^t x \leq b_i \epsilon I \}$$

donde I es un conjunto de índices (infinito) correspondiente a los semiespacios que contienen a S. Con S considerado de este modo el problema (3.1) se puede tomar como un problema de programación lineal (infinito).

Para éste programa lineal existe (al menos formalmente) el problema dual

$$\text{maximizar} \sum_{i \in I} \alpha_i b_i$$

sujeto a

$$\sum_{i \in I} \lambda_i a_i = c$$

$$\lambda_i \geq 0, \qquad i \in I$$

Seleccionando un subconjunto finito de I, por ejemplo I, y formando

$$P = \{x : a_i^T x \leq b_i,\ i \in I\}$$

resulta un polítope que contiene S. Al minimizar $c^T x$ sobre este polítope, se generan un punto y un subconjunto correspondiente de restricciones activas I_A, Entonces el problema dual con la restricción adicional $\lambda_i = 0$ para $i \in I$, tendrá una solución factible, pero esta solución general, no será óptima. Así, una solución a un problema de polítope corresponde a una solución factible, pero no óptima del dual. Por esta razón, se puede considerar que el método de plano cortante busca la optimalidad del dual (de dimensión infinita).

A continuación se presentan algunos conceptos que se usarán continuamente en los algoritmos de planos de corte.

Se entiende por [X] al entero más grande menor que el número X, es decir

$$[X] = \text{Máx } Y \leq X,\ Y \text{ entero} \qquad (3.1)$$

Por ejemplo [6.51] = 6, [-6.51] = -7, [0.3] = 0, [-0.3] = -1.

Se considera el siguiente problema entero

$$\text{Max } cx$$

sujeto a

$$Ax = b$$

$$x \geq 0, \text{ entero.} \qquad (3.2)$$

El problema lineal correspondiente a (3.2) es

sujeto a

$$Ax = b$$

$$x \geq 0 \qquad (3.3)$$

Una restricción típica de (3.3) es

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n + x_{N+i} = x_{Bi},$$

donde $x_1,\ldots,x_n$ son variables no básicas; $x_{n+1},\ldots,x_{n+m}$ son los vectores básicos en y x_{Bi} es la variable básica correspondiente.

Esta restricción se puede escribir

$$\sum_{j=1}^{n} [a_{ij}]\, x_j + \sum_{j=1}^{n} (a_{ij}-[a_{ij}])\, x_j + x_{n+i} = [x_{Bi}] + (x_{Bi}-x_{Bi}])$$

reagrupando se tiene:

$$\Sigma\ [a_{ij}]x_j + x_{n+i} - [x_{Bi}] = (x_{Bi} - [x_{Bi}]) - \sum_{j=1}^{n} (a_{ij}-[a_{ij}])x_j$$

además se demuestra que:

$$\sum_{j=1}^{n} (a_{ij} - [a_{ij}])x_j \geq (x_{Bi} - [x_{Bi}]) \qquad i=1,\ldots,m$$

que es un corte

Ejemplo 3.1

Se deriva un corte de la siguiente restricción

$$2.8x_1 - 3.6x_2 - 0.7x_3 + 4x_4 + x_5 = 8.93$$

donde x_5 es la variable básica. Se tiene, separando la parte entera de la fraccionaria:

$$2x_1 + 0.8x_1 - 4x_2 + 0.4x_2 - x_3 + 0.3x_3 + 4x_4 + x = 8 + 0.93$$

por lo que el corte resulta ser

por lo que el corte resulta ser

$$0.8x_1 + 0.4x_2 + 0.3x_3 \geq 0.93$$

A continuación se analizan varios de estos métodos. Al final del análisis se presentará una comparación de los mismos, relativa al tiempo que tardan en resolver un mismo problema *tipo*, en una computadora y veremos un algoritmo que utiliza el método dual simplex y que permite números fraccionarios en los cálculos.

3.2 ALGORITMO DUAL FRACCIONARIO.

Propósito: Resolver el problema de programación entera.

Descripción

Paso 1. Comenzamos con una tabla entera. (lo que requiere datos racionales. Si aparecen números irracionales, la convergencia del algoritmo no se puede garantizar). Se resuelve el programa entero como uno lineal. Si no es factible entonces termine. Si la solución óptima es entera, el programa entero está resuelto, terminamos. Si no vaya al paso 2.

Paso 2. Derive una nueva restricción de desigualdad (o corte) de los requerimientos y restricción (actual) enteros que aísle el punto óptimo (actual) (es decir, que hace la solución del programa de programación lineal no factible). Pero que no elimine a alguna solución entera. Aumente la nueva desigualdad hasta arriba del tablero simplex que entonces exhibe la infactibilidad primal (el valor de la variable de holgura asociada en el nuevo renglón será negativo). Vaya al paso 3.

Paso 3. Reoptimice usando el método dual simplex (lexicográfico) si el nuevo programa lineal no es factible* el problema entero no tiene solución, termine. Si la nueva solución óptima es entera, el programa entero está resuelto. Termine. De otra manera vaya al paso 2.

De acuerdo con la figura 3.2 ABCDE es la región factible asociada con las restricciones originales de un programa de programación lineal.

---

* Del teorema de dualidad de la programación lineal, infactibilidad primal indicará que el dual es no acotado. En cómputo esto indica que un tableau se alcanzará donde una columna pivote no se puede encontrar (ver ejemplo 3.2).

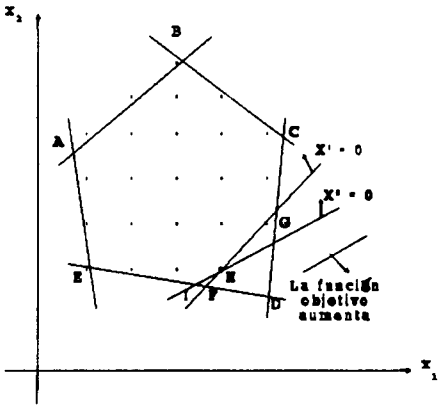


Figura 3.2

Las soluciones enteras corresponden a los puntos y el punto D es la (primer) solución óptima del problema de programación lineal (p.p.l.) Se aumenta una nueva restricción $x' \geq 0$ que aísle a D y reoptimizando se tiene que la nueva solución óptima del p.p.l. es F (la región factible es ABCGFE). Se agrega una nueva restricción $x'' \geq 0$ que elimina a F (la región factible es ABCGHIE). Resolviendo el programa lineal la solución óptima entera es el vértice H.

A menudo es relativamente fácil obtener desigualdades que son satisfechas por todas las soluciones enteras y que también aislen a la solución óptima continua actual. La dificultad es probar que un número finito de tales restricciones producirá un programa lineal cuya solución óptima sea entera. Es posible obtener "el fenómeno de la curva" como el de la figura 3.3

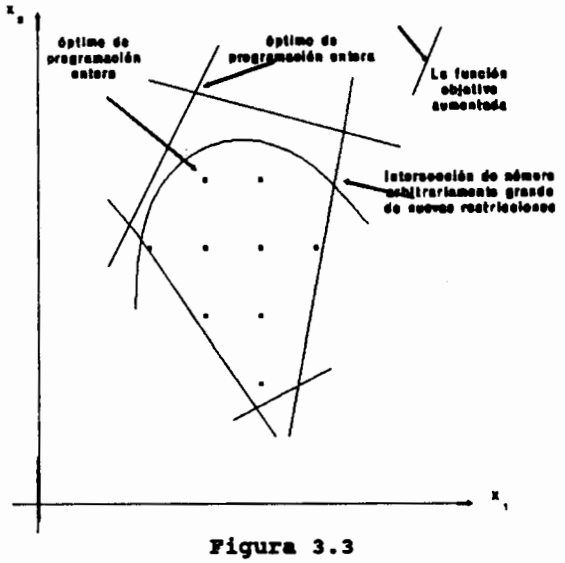


Figura 3.3

Donde la curva es la intersección de un número arbitrariamente grande de cortes.

Este algoritmo lo podemos reescribir en la siguiente forma:

Paso 1. Se resuelve el problema entero como un problema de programación lineal olvidándose por el momento de las condiciones de integralidad.

Paso 2. Si el resultado óptimo del paso 1 o del paso 3 es entero, pare. Se ha obtenido el resultado óptimo del problema original. De otra manera continúe en el paso 3.

Paso 3. Seleccione el máximo $(x_{Bi} - [x_{Bi}])$ fraccionario y genere un corte.

$$\sum_{j=1}^{n} (a_{ij} - [a_{ij}])x_j \geq (x_{Bi} - [x_{Bi}])$$

Donde $[x_{Bi}]$ es el entero más grande menor que x_{Bi} p ej. [6.51] = 6, [0.3] = 0. Añada este corte como una restricción adicional, y resuelva el problema por el método dual simplex y regrese al paso 2.

Ejemplo 3.2

Resuelva usando el algoritmo fraccionario de Gomory el siguiente problema entero

$$\max z = 5x_1 + 2x_2$$

sujeto a

$$\begin{aligned} 2x_1 + 2x_2 + x_3 &= 9 \\ 3x_1 + x_2 + x_4 &= 11 \end{aligned}$$

$$x_i \geq 0 \qquad x_i \text{ enteros}$$

Paso 1. La solución del programa lineal genera el siguiente tableau óptimo

z	x_1	x_2	x_3	x_4	x_B
1	0	0	0.25	1.5	18.75
0	0	1	0.75	-0.5	1.25
0	1	0	-0.25	0.5	3.25

Paso 3. Como $x_1 = 3.25$ y $x_2 = 1.25$ no son enteros, se debe generar un corte. El máximo $(x_{Bi} - [x_{Bi}])$ corresponde al primer renglón del tableau, cuya restricción lee

$$z + 0.25x_3 + 1.5x_4 = 18.75$$

por lo tanto el corte es:

$$(0.25 - 0)x_3 + (1.5-1)x_4 \geq 18.75 - 18$$

$$0.25x_3 + 0.5x_4 \geq 0.75$$

agregando el corte y una variable superflua S_1, tableau queda:

z	x_1	x_2	x_3	x_4	S_1	x_B
1	0	0	0.25	1.5	0	18.75
0	0	1	0.75	-0.5	0	1.25
0	1	0	-0.25	0.5	0	3.25
0	0	0	-0.25	-0.5	1	-0.75

aplicando el dual simplex (2 iteraciones más) se obtiene el siguiente tableau óptimo (para el problema lineal) pero que aún no es entero.

z	x_1	x_2	x_3	x_4	S_1	x_B
1	0	0.5	0	0	2.5	17.5
0	0	-0.5	0	1	-1.5	0.5
0	1	0.5	0	0	0.5	3.5
0	0	1	1	0	-1	2

$$x_1 = 3.5 \qquad x_3 = 2 \qquad x_4\ 0.5$$

por lo tanto, el nuevo corte se toma de

$$z + 0.5x_2 + 2.5\ S_1 = 17.5$$

que tiene el máximo $(x_{bi} - x_{Bi}])$ de todas las posibles candidatos x_{Bi} fraccionales. El nuevo corte que se añade es

$$(0.5-0)x_2 + (2.5-2)S_1 \geq 17.5 - 17$$

es decir

$$0.5\ x_2 + 0.5\ S_1 \geq 0.5$$

el nuevo tableau a resolver por medio del dual simplex es

Tableau 3

z	x_1	x_2	x_3	x_4	S_1	S_2	x_B
1	0	0.5	0	0	2.5	0	17.5
0	0	-0.5	0	1	-1.5	0	0.5
0	1	1	0	0	0.5	0	3.5
0	0	1	1	0	-1	0	2
0	0	-0.5	0	0	-0.5	1	-0.5

cuyo resultado óptimo da

Tableau 4

z	x_1	x_2	x_3	x_4	S_1	S_2	x_B
1	0	0	0	0	2	1	17
0	0	0	0	1	-1	-1	1
0	1	0.5	0	0	0	1	3
0	0	0	1	0	-2	2	1
0	0	1	0	0	1	-2	1

$z = 17 \quad x_1 = 3,\ x_2 = 1,\ x_3 = 1,\ x_4 = 1$

La ilustración gráfica del efecto de los cortes se ven en la figura 3.4

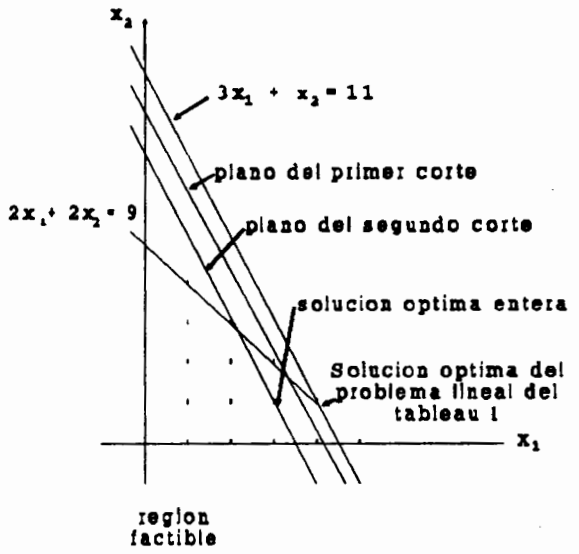


Figura 3.4

Se ve del ejemplo anterior que aunque a cada iteración le corresponde un aumento de una restricción y una variable superflua el número de variables no básicas se mantiene constante e igual a n. Por lo tanto, nunca habrá más de n variables de holgura fuera de la base, y se puede eliminar cualquier corte cuya variable de holgura se haya convertido en básica. Esto evita el crecimiento del tableau. Se puede por lo tanto trabajar con un tableau compactado de la siguiente manera: con cada corte, se añade la restricción de no-negatividad en la forma $-x_j \leq 0$, se omite la matriz identidad y se elimina la restricción de corte después de haber efectuado la iteración. La restricción de no negatividad se convierte en el corte actualizado únicamente cambia el valor de las variables no básicas. También, cuando una variable se convierte en no básica, se restará en la fila correspondiente una restricción de no-negatividad.

Ejemplo 3.3

Si tomamos las columnas asociadas a las variables no básicas, x_3, x_4 del tableau 1 del ejemplo anterior, la columna correspondiente a la variable de holgura del nuevo corte S_1, la columna del lado derecho y añadiendo las filas correspondientes a las condiciones de no-negatividad de las variables no básicas x_3 y x_4 se tiene:

		x_3	x_4	S_1	LD
vectores	z	0.25	1.5	0	18.75
en la	x_2	-0.75	-0.5	0	1.25
base	x_1	-0.25	0.5	0	3.25
Cond. de no negatividad de las variables no básicas	x_3	-1	0	0	0
	x_4	0	-1	0	0
corte		-0.25	-0.5	1	-0.75

Aplicando el dual simplex se pivotea en el elemento -0.25 produciendo el siguiente tableau

	x_3	x_4	S_1	LD
z	0	1	1	18
x_2	0	-2	3	-1
x_1	0	1	-1	4
x_3	0	2	-4	3
x_4	0	-1	0	0
	1	2	-4	3

se elimina esta columna

se elimina esta fila

A este tableau se le elimina la última fila (el corte) si corresponde a una variable de holgura (si no, no se elimina), y la columna básica (en este caso) la x_3, al empezar la siguiente iteración.

Ejemplo 3.4

$$\text{Max } z = 2x_1 + x_2$$

sujeto a

$$\begin{aligned} x_1 + x_2 + x_3 &= 5 \\ -x_1 + x_2 + x_4 &= 0 \\ 6x_1 + 2x_2 + x_5 &= 21 \end{aligned}$$

usando el simplex se tiene:

z	x_1	x_2	x_3	x_4	x_5	LD
1	-2	-1	0	0	0	0
0	1	1	1	0	0	5
0	-1	1	0	1	0	0
0	6	2	0	0	1	21
0	-7	-3	-1	0	-1	-26

z	x_1	x_2	x_3	x_4	x_5	LD
1	0	-3	0	-2	0	0
0	0	2	1	1	0	5
0	1	-1	0	-1	0	0
0	0	8	0	6	1	21
0	0	-10	-1	-7	-1	-26

z	x_1	x_2	x_3	x_4	x_5	LD
1	0	0	1.5	-.5	0	7.5
0	0	1	.5	.5	0	2.5
0	1	0	.5	-.5	0	2.5
0	0	0	-4	2	1	1
0	0	0	4	-2	-1	-1

z	x_1	x_2	x_3	x_4	x_5	LD
1	0	0	.5	0	.25	7.75
0	0	1	1.5	0	-.25	2.25
0	1	0	-.5	0	.25	2.75
0	0	0	-2	1	.5	.5
0	0	0	.5	0	0	0

el valor de la función objetivo $z = 7.75$ con $x_1 = 2.75$, $x_2 = 2.25$ $x_4 = 0.5$, $x_3 = x_5 = 0$

Se generan los cortes de la siguiente manera

$$z + .5x_3 + 0.25\ x_5 = 7.75$$

$\therefore$ el corte es

$$(0.5-0)x_3 + (0.25-0)x_5 \geq 7.75-7$$

o bien

$$0.5x_3 + 0.25x_5 \geq 0.75$$

también se tiene que

$$-0.5x_3 + 0.25x_5 = 2.75$$

$\therefore$ el corte es: $(-1+0.5)x_3 + (0.25-0)x_5 \geq 2.75-2\ 0.5x_3 + 0.25x_5 \geq 0.75$

que es igual al primero entonces se escoge cualquiera de ellos y se agrega al tableau.

Otro aspecto importante es si es posible identificar geométricamente los cortes, por ejemplo el corte

$$0.5x_3 + 0.25x_5 \geq 0.75$$

es equivalente a $2x_1 + x_2 \leq 7$ ya que

$$z + 0.5x_3 + 0.25x_5 = 7.75$$

lo que implica $2x_1 + x_2 + 0.5x_3 + 0.25x_5 = 7.75$

de donde, si $0.5x_3 + 0.25x_5 \geq .75$, entonces $2x_1 - x_2 \leq 7$

o por ejemplo, el corte que resulta de

$$-2x_3 + 0.5x_5 = 0.5$$

es

$$0.5x_5 \geq 0.5$$

es equivalente a $6x_1 + 2x_2 \leq 20$, ya que

$$6x_1 + 2x_2 + x_5 = 21$$

si $\quad 0.5x_5 \geq 0.5 \rightarrow \ x_5 \geq 1$

entonces $\quad 6x_1 \_ 2x_2 \leq 20$

Si analizamos esto geométricamente tenemos en la figura 3.5

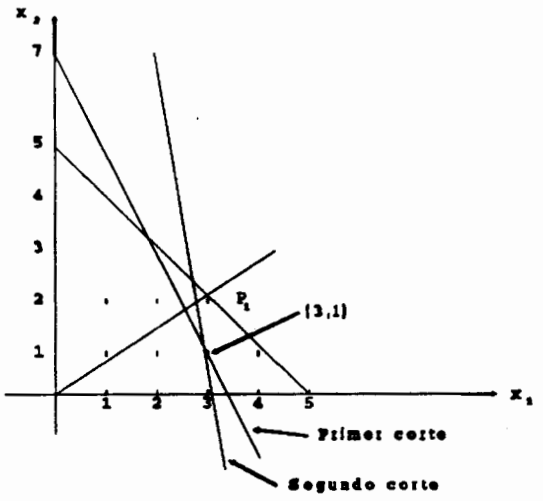


Figura 3.5

Agregando los cortes uno a uno y usando el dual simplex se obtiene:

$$\begin{aligned} z &= 7 \\ x_1 &= 3 \\ x_2 &= 1 \\ x_4 &= 2 \\ x_5 &= 1 \\ x_3 &= 1 \end{aligned}$$

El paso de una tabla a otra consiste en ir de P_1 a P_2 ya que la restricción

$$\frac{x_3}{2} + \frac{S_1}{2} \geq \frac{1}{2}$$

es equivalente a

$$2x_1 + x_2 \leq 7$$

Cuando se tienen empates se suelen usar las llamadas **Reglas Lexicográficas** que consisten en:

Un vector es lexicográficamente positivo si la primera componente diferente de cero es positiva. Un vector lexicográficamente positivo se escribe $x \succ 0$ o $x \succ 0$, por ejemplo

$$x = (0.3, -2,9) \succ 0$$

$$y = (108, -3, 12) \succ 0$$

$$z = (0, 0, -3, 12) \text{ no es } \succ 0$$

Definir un vector X es lexicográficamente mayor que otro vector Y si el vector X-Y $\succ$ 0, ejemplo

$$X = (0, 3, -2)$$

$$Y = (1, 2, 2)$$

$$Y - X = (1, -1, 4) \succ$$

$$X - Y = (-1, 1, -4) \text{ no es } \succ 0$$

3.3 LA NOTACION DE TABLEAU

Para facilitar la exposición y conformar una notación estándar escribimos un programa entero de la siguiente forma.

$$\max x_0 = a_{00} - a_{01}x_1 - a_{02}x_2 - \ldots - a_{0n}x_n$$

sujeto a

$$\left.\begin{aligned} x_1 &= -1\ (-x_1) \\ x_2 &= -1(-x_2) \\ &\vdots \\ x_n &= -1(-x_n) \end{aligned}\right\} \text{variables originales}$$

$$x_{n+1} = a_{n+1,0} + a_{n+1,1}(-x_1) + a_{n+1,2}(-x_2) + \ldots + a_{n+1,n}(-x_n)$$

$$\vdots$$

$$x_{n+m} = a_{n+m,0} + a_{n+m,1}(-x_1) + a_{n+m,2}(-x_2) + \ldots + a_{n+m,n}(-x_n)$$

$$x_j \geq 0 \quad \text{entero} \qquad j=1,\ldots,n+m$$

Observaciones:

1. En relación al programa entero

$$\text{maximizar } z = cx$$

$$\text{sujeto a}$$

$$Ax \leq b$$

$$x \geq 0$$

$$y \qquad x \text{ entero}$$

donde $A = (a_{ij})$ es de mxn, tenemos $x_0 = z$, $a_{00} = 0$, $c_j = -a_{0j}$ donde $j=1,\ldots,n$ $b_i = a_{n+i,0}$ $(i=1,\ldots,m)$ $a_{ij} = a_{n+i,j}$ $(i=1,\ldots,m,\ j=1,\ldots,n)$ y x_{n+i} $(i=1,\ldots,m)$ son las variables de holgura.

2. Los requerimientos x_j enteras $(j=1,\ldots,n)$ y la suposición de que los datos son enteros aseguran que el valor de la función objetivo (x_0) y las variables de holgura $(x_{n+1},\ldots,x_{n+m})$ son enteros para cualquier solución entera.

   Se puede exhibir el problema en el siguiente tableau simplex donde cada variable x_i $(i=0,1,\ldots,n+m)$ aparece como combinación lineal de variables no básicas $x_1,\ldots,x_n$

	1	$-x_1$	$\ldots$	$-x_j$	$\ldots$	$-x_n$	
$x_0 =$	a_{00}	a_{01}	$\ldots$	a_{0j}	$\ldots$	a_{0n}	
$x_1 =$	0	-1					≥ 0
$\vdots$	$\vdots$		$\ddots$				$\vdots$
$x_j =$	0			-1			
$\vdots$	$\vdots$				$\ddots$		
x_n	0					-1	≥ 0
$x_{n+1} =$	$a_{n+1,0}$	$a_{n+1,1}$	$\ldots$	$a_{n+1,j}$	$\ldots$	$a_{n+1,n}$	≥ 0
$\vdots$							$\vdots$
$x_{n+m} =$	$a_{n+m,0}$	$a_{n+m,1}$	$\ldots$	$a_{n+m,j}$	$\ldots$	$a_{n+m,n}$	≥ 0

además usamos la siguiente notación:

J Contiene los n índices de las actuales variables no básicas como aparecen de izquierda a derecha en la línea superior del tableau. Inicialmente J = {1,2,...,n}

J(j)(j=1,...,n) es el j-ésimo elemento en J. Por ejemplo, si n=3 y J = {2,5,1}, entonces J(1) = 2, J(2) = 5 y J(3) = 1.

$\succ$ ($\prec$) denota una columna (lexicográficamente positiva (o negativa). Equivalentemente, aquella cuyo primer elemento $\neq 0$ es positivo o negativo.

[y] Significa el mayor entero menor o igual que, y

α_j(j=0,1,...,n) Son las primeras m+n+1 componentes en la j-ésima columna del tableau; esto es

$$\alpha_j = \begin{bmatrix} a_{0j} \\ \cdot \\ \cdot \\ \cdot \\ a_{n+m,j} \end{bmatrix}$$

3.4 EL CORTE DE GOMORY.

En un tableau simplex óptimo $a_{0j} \geq 0$ (j=1,...,n) $a_{i0} \geq 0$ (i=1,...,n+m) y $\alpha_j \succ 0$ (j=1,...,n), (como a_{00} puede ser negativo, no podemos decir que $\alpha_0 \succ 0$) más aún si a_{i0} (i=0,1,...,n+m) son todos enteros, el programa entero se resuelve con $x_i = a_{i0}$ (i=0,...,m+n). De otra forma hay al menos un renglón fuente o generador v con

$$x_v = a_{v0} + \sum_{j=1}^{n} a_{vj}(-x_{J(j)}) \text{ y } a_{v0} \text{ fraccionario}$$

Entonces el k-ésimo corte de Gomory tiene la forma:

$$x_{n+m+k} = - f_{v0} + \Sigma\, (-f_{vj})(-x_{J(j)}) \geq 0 \qquad (3.1)$$

donde x_{n+m+k} es la variable de holgura de Gomory asociada en la k-ésima desigualdad aumentada y $f_{vj} = a_{vj} - [a_{vj}]$ (j=0,1,...,n).

Observe que $0 < f_{v0} < 1$, $0 \leq f_{vj} < 1$ j=1,...,n

y cuando la desigualdad se agrega al tableau, se introduce más infactibilidad primal ya que $x_{n+m+1} = -f_{v0} < 0$. Demostraremos posteriormente que la ecuación (3.1) se satisface por toda solución entera y que tiene suficientes propiedades para soportar un algoritmo finito. Pero antes de continuar veremos un ejemplo

Ejemplo 3.5

$$\max x_0 = -4x_1 - 5x_2$$

sujeto a

$$-x_1 - 4x_2 \leq -3$$
$$-3x_1 - 2x_2 \leq -7$$

y $x_1, x_2 \geq 0$ enteros

El dual asociado es

$$\min w_0 = -5w_1 - 7w_2$$

sujeto a

$$-w_1 - 3w_2 \geq -4 \quad (w_{s1})$$
$$-4w_1 - 2w_2 \geq -3 \quad (w_{s2})$$

y $w_1, w_2 \geq 0$

Las entradas encerradas con un círculo en el tableau sin los pivotes y las flechas (→) indican el renglón fuente. Este último es el que tiene la mayor componente fraccionaria a_{i0}. En este caso de empates se selecciona el mayor número.

Antes de introducir variables de holgura no negativas x_3 y x_4 obtenemos el tableau 1.

Tableau 1

	1	$-x_2$	$-x_2$	$(x_1 = x_2 = 0)$
x_0	0	4	5	Selección del renglón pivote
x_1	0	-1	0	Min (-5, -7) = -7 o
x_2	0	0	-1	x_4 es no básica
x_3	-5	-1	-4	Selección de la col. pivote
x_4	-7	⊖3	-2	$\min\left\{\frac{4}{\lvert-3\rvert}, \frac{5}{\lvert-2\rvert}\right\} = \frac{4}{3}$

o x_1 se convierte en básica.

En este tableau se ve la infactibilidad primal y la factibilidad dual. Por lo tanto se usa el método dual simplex lexicográfico.

En el dual, se multiplica cada restricción por -1 y se aumentan las variables de holgura w_{s1} y w_{s2} para obtener:

$$w_1 + 3w_2 + w_{s1} = 4$$

$$4w_1 + 2w_2 + w_{s2} = 5$$

Entonces haciendo las variables no básicas $w_1 = w_2 = 0$ se obtiene una solución básica factible con variables $w_{s1} = 4$ y $w_{s2} = 5$. Los valores aparecen en el renglón x_0 del tableu.

Tableau 2

	1	$-x_4$	$-x_2$
x_0	-28/3	4/3	7/3
x_1	7/3	-1/3	2/3
x_2	0	0	-1
x_3	-8/3	-1/3	-10/3
x_4	0	-1	0

$(x_1 = 7/3 \quad x_2 = 0)$

Tableau 3

	1	$-x_4$	$-x_3$
x_0	-112/10	11/10	7/10
x_1	18/10	-4/10	2/10
x_2	0	0	-1
x_3	0	-1	0
x_4	0	-1	0
x_5	-8/10	-6/10	-2/10

$(x_1 = 18/10 \quad x_2 = 8/10)$

$a_{10} = 18/18$

$f_{10} = 18/10 - [18/10] = 8/10$

$a_{11} = -4/10$

$f_{11} = -4/10 - [-4/10] = 6/10$

$a_{12} = 2/10$

$f_{12} = 2/10 - [2/10] = 2/10$

El tableau 3 exhibe la solución óptima del problema de programación lineal $x_0 = -112/10$ con $x_1 = 18/10 \quad x_2 = 8/10$ y $x_3 = x_4 = 0$. Se selecciona el renglón 1 como fuente y se pivotea obteniendo.

Tableau 4

	1	$-x_5$	$-x_3$	($x_1 = 14/6$ $x_2 = 4/6$)
x_0	-76/6	11/6	2/6	
x_1	14/6	-4/6	2/6	
x_2	4/6	1/6	-2/6	
x_3	0	0	-1	
x_4	8/6	-10/6	2/6	
x_5	0	-1	-1/6	
x_6	-4/6	-1/6	-4/6	

Se aumenta un 2_0 corte después de otro pivoteo se tiene $x_1 = 2$ y $x_2 = x_3 = x_4 = 1$ con $x_0 = -13$.

Tableau 5

	1	$-x_5$	$-x_6$	($x_1 = 2$ $x_2 = 1$)
x_0	-13	7/4	2/4	
x_1	2	-3/4	2/4	
x_2	1	1/4	-2/4	
x_3	1	1/4	-6/4	
x_4	1	-7/4	2/4	
x_5	0	-1	0	
x_6	0	0	-1	
x_7	0	-3/4	-2/4	

3.5 ALGORITMO ENTERO PURO DE GOMORY

Este método es una variante del anterior y pertenece a un proceso de optimización con punto de partida que es dual factible. Todos los coeficientes de la matriz A deben ser enteros (requisito que no se tiene en el algoritmo anterior). El método asegura que esta condición de integralidad de los coeficientes permanece constante durante todas las iteraciones. Los pasos a seguir son los siguientes:

Paso 1. Se escoge la x_{Bi} más negativa. Se designa ésta fila con r. Si el método dual simplex genera un pivote -1, itérese y repita el paso 1. Si no, se continúa en el paso 2.

Paso 2. Se escoge aquella columna no básica con $a_{rj} < 0$ que lexicográficamente sea la menor. Se designa a la columna por k. Al primer elemento distinto de cero de dicha columna se le designa por a_{pk} (> 0) siendo su fila correspondiente la p.

Paso 3. Para las columnas con $a_{rj} < 0$, se calcula el índice $u_j = \{a_{pj}/a_{pk}\}$ si es que a_{pj} es el primer elemento distinto de cero en la columna p. De otra manera $u_j = \infty$.

Paso 4. Se calcula $\lambda = \max\left\{\dfrac{|a_{rj}|}{u_j}\right\}$ para $a_{rj} < 0$ y $u_j \neq \infty$

Paso 5. Se derivan un corte

$$\sum_{j=1}^{n} \left[\frac{a_{rj}}{\lambda}\right] x_j \leq \left[\frac{x_{Bi}}{\lambda}\right]$$

Paso 6. Se anexa este corte al tableau, junto con su variable de holgura correspondiente, y se aplica el método dual simplex pivoteando en algún elemento del corte[2]. Si el resultado es $x_B \geq 0$ (entero) entonces es la solución óptima. Si no, se regresa al paso 1. Para efectos de notación, el elemento del tableau en la fila i columna j se le designa a_{ij}.

---

[2] de acuerdo al dual simplex.

Ejemplo 3.6

Resuelva el siguiente problema usando el algoritmo entero puro

$$\max z = -10x_1 - 14x_2 - 21x_3$$

sujeto a

$$\begin{aligned} 8x_1 + 11x + 9x_3 &\geq 12 \\ 2x_1 + 2x_2 + 7x_3 &\geq 14 \\ 9x_1 + 6x_2 + 3x_3 &\geq 10 \end{aligned}$$

$$x_i \geq 0 \quad \text{entero} \quad i=1,2,3$$

Tableau original:

		x_1	x_2	x_3	LD
	z	10	14	21	0
Elementos en la base	x_4	-8	-11	-9	-12
	x_5	-2	-2	-7	-14
	x_6	-9	-6	-3	-10

Iteración 1.

Paso 1. Se elige $x_B = -14$. Por lo tanto $r = 3$ (fila 3).

Paso 2. Los candidatos a_{rj} (<0) son: $a_{31} = -2$, $a_{32} = -2$, $a_{33} = -7$. De las 3 columnas la primera es la lexicográficamente más pequeña, por lo tanto $k = 1$. El elemento $a_{pk} = a_{11} = 10$ por lo que $p = 1$.

Paso 3

$$u_1 = \left\lfloor \frac{a_{11}}{a_{11}} \right\rfloor = \left\lfloor \frac{10}{10} \right\rfloor = 1$$

$$u_2 = \left\lfloor \frac{a_{12}}{a_{11}} \right\rfloor = \left\lfloor \frac{14}{10} \right\rfloor = 1$$

$$u_3 = \left\lfloor \frac{a_{13}}{a_{11}} \right\rfloor = \left\lfloor \frac{21}{10} \right\rfloor = 2$$

Paso 4. $\lambda = \max\left\{\frac{|a_{0j}|}{u_j}\right\} = \max\left\{\frac{2}{1}, \frac{2}{1}, \frac{7}{2}\right\} = 3.5$

Paso 5. El corte se deriva de la restricción

$$\left[\frac{-2}{3.5}\right] x_1 + \left[\frac{-2}{3.5}\right] x_2 + \left[\frac{-7}{3.5}\right] x_3 \leq \left[\frac{-14}{3.5}\right]$$

y es igual a

$$-x_1 - x_2 - 2x_3 \leq -4$$

que añadida al tableau junto con su variable superflua S_1, da:

Paso 6.

	S_1	x_1	x_2	x_3	LD	
z	0	10	14	21	0	
x_4	0	-8	-11	-9	-12	
x_5	0	-2	-2	-7	-14	
x_6	0	-9	-6	-3	-10	
	1	-1	-1	-2	-4	corte

se aplica el dual simplex pivoteando en el elemento $a_{s1} = -1$ como la columna x_1 se va a convertir en e_5 después de la iteración, se agrega una columna e_5 (denominada S_1 que corresponde a la primera variable superflua). El nuevo tableau es:

S_1	x_1	x_2	x_3	LD
10	0	4	1	-40
-8	0	-3	7	20
-2	0	0	-3	-6
-9	9	3	15	26
-1	1	1	2	4

Iteración 2

Paso 1

	S_1	x_2	x_3	LD
	10	4	1	-40
x_4	-8	-3	7	20
x_5	-2	0	-3	-6
x_6	-9	3	15	26
x_1	-1	1	2	4

Se elige $x_B = -6$ y por lo tanto r = 3

Paso 2

Los candidatos a_{rj} (< 0) son: $a_{31} = -2$ y $a_{33} = -3$. De las dos columnas con $a_{rj}<0$, la lexicográficamente más pequeña es la que corresponde a j = 3. Por lo tanto k = 3. El elemento $a_{pk} = a_{13} = 1$ y p = 1

Paso 3

$$u_1 = \left\lceil \frac{10}{1} \right\rceil = 10 \qquad u_2 = \infty \qquad u_3 = \left\lceil \frac{1}{1} \right\rceil = 1$$

Paso 4. $\lambda = \text{Máx} \left\{ \frac{2}{10}, \frac{3}{1} \right\} = 3$

Paso 5. El corte se deriva de la restricción

$$\left\lceil \frac{-2}{3} \right\rceil S_1 + \left\lceil \frac{-3}{3} \right\rceil X_3 \leq \left\lceil \frac{-6}{3} \right\rceil$$

y es igual a

$$-S_1 - X_3 \leq -2$$

Este corte se añade al tableau, quedando éste como

Paso 6.

S_1	X_2	X_3	X_B		
10	4	1	-40	Z	
-8	-3	7	20	x_4	vectores
-2	0	-3	- 6	x_5	en la
-9	3	15	26	x_6	base
-1	1	2	4	x_1	
-1	0	(-1)	- 2	x_3	← corte

Aplicando el *método dual simplex* pivoteando en el elemento $a_{53} = -1$ y agregando una columna unitaria e_6 a la que se denomina S_2, (S_2 es también la variable superflua del nuevo corte), se tiene antes de la iteración.

S_2	S_1	X_2	X_3	X_B		
0	10	4	1	-40	Z	
0	-8	-3	7	20	X_4	vectores
0	-9	3	15	26	X_5	en la
0	-1	1	2	4	X_6	base
0	-1	1	2	4	X_1	
1	-1	0	(-1)	- 2	X_3	

y después de la iteración

S_2	S_1	X_2	X_3	X_B		
1	9	4	0	-42	Z	
7	-15	-3	0	6	X_4	vectores
-3	1	0	0	0	X_5	en la
15	-24	3	0	-4	X_6	base
2	-3	1	0	0	X_1	
-1	1	0	1	2	X_3	

Se elimina la columna X_3

Iteración 3.

Paso 1. Se elige $X_B = -4$ y por lo tanto $r = 4$.

Paso 2. Los candidatos $a_{rj}(< 0)$ son: $a_{42} = -24$. Por lo tanto, $k = 2$ y $a_{pk} = 9$, siendo $p = 1$.

Paso 3. $u_1 = \infty$, $u_2 = \left[\frac{9}{9}\right] = 1$, $u_3 = \infty$.

Paso 4. $\lambda = \text{Máx}\left\{\frac{24}{1}\right\} = 24$

Paso 5. El corte se deriva de la restricción

$$\left[\frac{15}{24}\right]S_2 + \left[\frac{-24}{24}\right]S_1 + \left[\frac{3}{24}\right]X_2 \leq \left[\frac{-4}{24}\right]$$

y es igual a

$$0S_2 - S_1 + 0X_2 \leq -1.$$

Este corte se añade al tableau, quedando éste como

Paso 6.

S_3	S_2	S_1	X_2	X_B		
0	1	9	4	-42	Z	
0	7	-15	-3	6	X_4	
0	-3	1	0	0	X_5	
0	15	-24	3	-4	X_6	vectores
0	2	-3	1	0	X_1	en la
0	-1	1	0	2	X_3	base
1	0	(-1)	0	-1	S_1	← corte

Aplicando el *método dual simplex*, pivoteando en el elemento encerrado en un círculo y añadiendo una columna unitaria e_7 denominada S_3, se tiene:

Iteración 4.

Paso 1.

S_3	S_2	S_1	X_2	X_B		
9	1	0	4	-51	Z	
-15	7	0	-3	21	X_4	
1	-3	0	0	-1	X_5	
-24	15	0	3	20	X_6	vectores
- 3	2	0	1	3	X_1	en la
1	-1	0	0	1	X_3	base
- 1	0	1	0	1	S_1	

Eliminando la columna S_1 se elige $X_B = -1$, por lo que r=3.

Paso 2. Los candidatos $a_{rj}(< 0)$ son $a_{32} = -3$. Por lo tanto $k = 2$ y $a_{pk} = 1$, siendo $p = 1$.

Paso 3. $u_1 = \infty$, $u_2 = \left| \frac{1}{1} \right| = 1$, $u_3 = \infty$.

Paso 4. $\lambda = \text{Máx} \left\{ \frac{3}{1} \right\} = 3$

Paso 5. El corte se deriva de la restricción

$$\left[\frac{1}{3} \right] S_3 + \left[\frac{-3}{3} \right] S_2 + \left[\frac{0}{3} \right] X_2 \leq \left[\frac{-1}{3} \right]$$

y es igual a

$$0S_3 - S_2 + 0X_2 \leq -1,$$

o sea

$$-S_2 \leq -1.$$

Este corte se añade al tableau. Como no interesa conocer el valor óptimo de la variable superflua S_1, se puede eliminar la última fila del último tableau.

Paso 6.

S_4	S_3	S_2	X_2	X_B		
0	9	1	4	-51	Z	
0	-15	7	-3	21	X_4	
0	1	-3	0	- 1	X_5	vectores
0	-24	15	3	20	X_6	en la
0	- 3	2	1	3	X_1	base
0	1	-1	0	1	X_3	
1	0	(-1)	0	-1	S_2	← corte

Aplicando el método *dual simplex*, pivoteando en el elemento encerrado en un círculo y añadiendo una columna unitaria e, denominada S_4 se tiene:

S_4	S_3	S_2	X_2	X_B		
-1	9	0	4	-52	Z	
7	-15	0	-3	14	X_4	
1	1	0	0	2	X_5	vectores
-3	-24	0	3	5	X_6	en la
15	-3	0	1	1	X_1	base
2	1	0	0	2	X_3	
-1	0	1	0	1	S_2	

Como todas las componentes de la columna X_B (a excepción del valor de la función objetivo), son no-negativas y enteras, la solución óptima es:

$$X_1 = 1,\ X_2 = 0,\ X_3 = 2,\ X_4 = 14,\ X_5 = 2,\ X_6 = 5,\ S_2 = 1$$

y el máximo de la función objetivo es

$$Z^* = -52$$

o sea, que el mínimo de la función objetivo es

$$Z^* = 52$$

Ejemplo 3.7

$$\max z = -4x_1 - 5x_2$$

sujeto a

$$-x_1 - 4x_2 \leq -5$$

$$-3x_1 - 2x_2 \leq -7$$

$$x_1,\ x_2 \geq 0 \quad x_1,\ x_2 \epsilon 2$$

Z	x_1	x_2	x_3	x_4	LD
1	+4	+5	0	0	0
0	-1	-4	1	0	-5
0	-3	-2	0	1	-7

Iteración 1.

Paso 1. r = 3

Paso 2. Los candidatos a_{rj} (<0) son: $a_{31} = -3$ $a_{32} = -2$ de las 2 columnas la primera es lexicográficamente mas pequeña por lo tanto k = 1. El elemento $a_{pk} = a_{11} = 4$ por lo que p = 1.

Paso 3.

$$u_1 = \left[\frac{a_{11}}{a_{11}}\right] = \left[\frac{4}{4}\right] = 1$$

$$u_2 = \left[\frac{a_{12}}{a_{11}}\right] = \left[\frac{5}{4}\right] = 1$$

Paso 4. $\lambda = \max\left\{\frac{|a_{rj}|}{u_j}\right\} = \max\left\{\frac{3}{1}, \frac{2}{1}\right\} = 3$

Paso 5. El corte se deriva de la restricción

$$\left[\frac{-3}{3}\right] x_1 + \left[\frac{-2}{3}\right] x_2 \leq \left[\frac{-7}{3}\right]$$

y es igual a

$$- x_1 - x_2 \leq -3$$

que añadido al tableau junto con su variable superflua S_1 da:

	S_1	x_1	x_2	LD
z	0	4	5	0
x_4	0	-1	-4	-5
x_5	0	-3	-2	-7
	1	(-1)	-1	-3

Se aplica el dual simplex pivoteando en $a_{41} = -1$ como x_1 se va a convertir en básica (denominada S_1) queda

		S_1	x_5	x_2	LD	
	z	4	0	1	-12	
→	x_3	-1	0	(-3)	-2	←
	x_4	-3	0	1	2	
		-1	1	1	3	
		1	-1	-1	-3	

Iteración 2.

como $1 < 4$ la columna 2 es el pivote k

$u_2 = 1$ y u_1 es el mayor entero para el cual

$u_1 = 4$

el corte es

$$\left[\frac{-1}{3}\right] S_1 + \left[\frac{-3}{3}\right] x_2 \leq \left[\frac{-2}{3}\right]$$

$$-S_1 - x_2 \leq -1$$

Se agrega al tableau

	S_1	x_1	x_2	LD
z	0	1	4	-12
x_4	0	-3	-1	-2
x_6	0	1	-3	2
S_1	0	+1	-1	-3
S_2	-1	-1	0	-1

se pivotea en -1

	S_1	x_1	x_2	LD
z	+1	0	-4	-13
x_4	3	0	-1	1
x_2	-1	0	-3	1
S_1	-1	0	-1	+2
S_2	1	1	0	1

$$x_5 \geq 0 \quad y \quad x_6 \geq 0$$

$$x_5 = -3 + x_1 + x_2 \geq 0$$

$$x_6 = -4 + x_1 + 2x_2 \geq 0$$

Por supuesto en cada tableau, cada coeficiente es entero, dicho tableau se refiere al plano (x_1, x_2) como se muestra en la figura 3.6

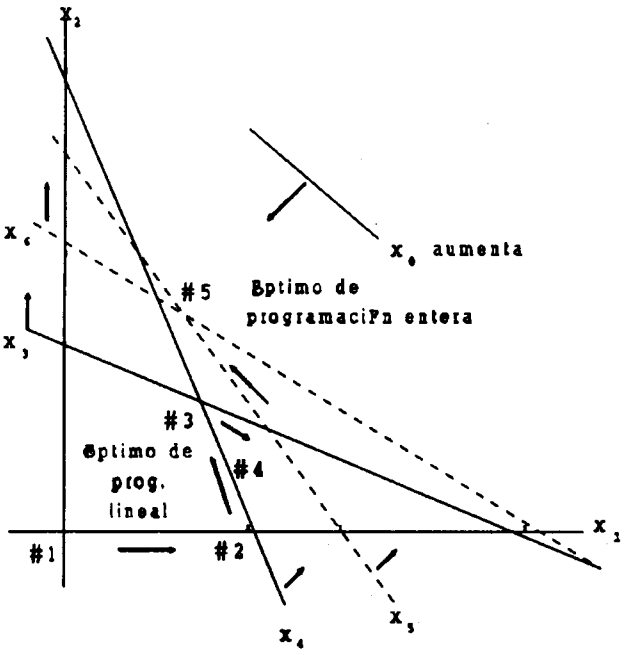


Figura 3.6

CAPITULO 4

PROPIEDADES DE LOS METODOS DE PLANOS DE CORTE

Todo método es imperfecto
Charles-Jean Henri Nicolle

4.1 COMPARACIÓN ENTRE LOS CORTES GENERADOS POR EL ALGORITMO FRACCIONAL Y EL ALGORITMO ENTERO PURO DE GOMORY.

Ejemplo 4.1

$$\max z = -4x_1 - 5x_2$$

sujeto a

$$-x_1 - 4x_2 \leq -5 \qquad -x_1 - 4x_2 + x_3 = -5$$

$$-3x_1 - 2x_2 \leq -7 \qquad -3x_1 - 2x_2 + x_4 = -7$$

$x_1, x_2 \geq 0$ enteros.

el tableau queda:

TABLEAU 1

	x_1	x_2	x_3	x_4	LD
z	+ 4	+ 5	0	0	0
x_3	- 1	- 4	1	0	-5
x_4	- 3	- 2	0	1	-7

Se selecciona el renglón min {-5, -7} = - 7, x_4 se convierte en no básica, y la columna

$$\min\left\{\frac{-4}{-3}, \frac{-5}{-2}\right\} = \frac{4}{3},$$ x_1 se convierte en básica.

TABLEAU 2

	x_1	x_2	x_3	x_4	LD
z	0	-7/3	0	-4/3	28/3
x_3	0	-10/3	1	-1/3	-8/3
x_1	1	2/3	0	-1/3	-7/3

TABLEAU 3

z	0	0	7/10	11/10	-112/10
x_2	0	1	-3/10	1/10	4/5
x_1	1	0	1/5	-4/19	18/10

se tiene el tableau óptimo con $x_1 = \frac{18}{10}$, $x_2 = \frac{8}{10}$

se genera entonces un corte con el x_{Bi} que tenga la fracción mayor:

$$z + 0.7x_3 + 0.11x_4 = -11.2$$

$$-0.3x_3 + 0.1x_4 = 0.8$$

$$0.2x_3 - 0.4x_4 = 1.8 \leftarrow$$

como hay empate escogemos cualquiera de los dos

$$(0-.2-0)x_3 + (-0.4+1)x_4 = 1 + 0.8$$

$$0.2x_3 + 0.6x_4 \geq 0.8$$

TABLEAU 4

		x_1	x_2	x_3	x_4	x_5	LD
	z	0	0	7/10	11/10	0	-112/10
	x_2	0	1	-2/10	1/10	0	4/5
	x_1	1	0	1/5	-4/10	0	18/10
*	x_5	0	0	-1/5	-3/5	1	-4/5

se aplica el dual simplex y se pivotea:

TABLEAU 5

	x_1	x_2	x_3	x_4	x_5	LD
z	0	0	1/3	0	11/6	-38/3
x_2	0	1	-1/3	0	1/6	2/3 ←
x_1	1	0	1/3	0	-2/3	7/3
x_4	0	0	1/3	1	5/3	4/3

se aumenta el corte

$$-0.3x_3 + 0.16x_5 = 0.66 \qquad 0.7x_3 + 0.16x_5 \geq 0.66$$

$$(-1 + 0.7)x_3 + 0.16x_5 = 0.66$$

$$-x_3 + 0.7x_3 + 0.16x_5 = 0.66$$

quedando

$$0.66x_3 + 0.16x_5 \geq 0.66$$

TABLEAU 6

	x_3	x_5	x_6	LD
z	1/3	11/6	0	-38/3
x_2	-1/3	1/6	0	2/3
x_1	1/3	2/6	0	7/3
x_4	1/3	5/3	0	4/3
x_6	-2/3	-1/6	1	-2/3 ←

Aplicando el dual simplex y pivoteando se obtiene la solución óptima entera $z = -13$, $x_1 = 2$, $x_2 = x_3 = x_4 = 1$.

Expresamos las nuevas restricciones términos de las variables originales (x_1, x_2).

$$* \quad x_5 = -\frac{4}{5} + \frac{3}{5}x_4 + \frac{1}{5}x_3 \geq 0 \text{ (del tableau 4)}$$

pero $x_3 = -5 + x_1 + 4x_2$ (del tableau 1)

y $x_4 \leq -7 + 3x_1 + 2x_2$

sustituyendo en (*) se tiene

$$x_5 = -6 + 2x_1 + 2x_2 \geq 0$$

ó $$x_1 + x_3 \geq 3$$

También $x_6 = -\frac{2}{3} + \frac{1}{6} x_5 + \frac{2}{3} \geq 0$

pero $x_5 = -6 + 2x_1 + 2x_2$

$x_3 = -5 + x_1 + 4x_2$

sustituyendo se tiene

$$x_6 = -5 + x_1 + 3x_2 \geq 0$$

ó $$x_1 + 3x_2 \geq 5$$

Nota: Si se continúa agregando cortes se obtiene un tableau entero puro

Usando el mismo ejemplo pero con el Algoritmo Entero Puro se tiene:

TABLEAU 1

	x_1	x_2	x_3	x_4	LD	
z	4	5	0	0	0	
x_3	-1	-4	1	0	-5	
x_4	(-3)	-2	0	1	-7	←

Se usa x_4 como renglón fuente y como $4 < 5$ la columna 1 es el pivote

Paso 2. $a_{pk} = a_{11} = 4 \quad p = 1$

Paso 3. $u_1 = \left[\frac{a_{11}}{a_{11}}\right] = \left[\frac{4}{4}\right] = 1$

$$u_j = \left[\frac{a_{pj}}{a_{pk}}\right] \quad u_2 = \left[\frac{a_{12}}{a_{11}}\right] = \left[\frac{5}{4}\right] = 1$$

Paso 4. Se cacula $\lambda = \max\left\{\frac{|a_{rj}|}{u_j}\right\}$

$$\lambda = \max\left\{\frac{-3}{1}, \quad \frac{|-2|}{1}\right\} = 3$$

Paso 5. Se deriva un corte

$$\sum_{j=1}^{n} \left[\frac{a_{rj}}{\lambda}\right] x_j \leq \left[\frac{x_{Bi}}{\lambda}\right]$$

$$\left[\frac{-3}{3}\right]x_1 + \left[\frac{-2}{3}\right]x_2 \leq \left[\frac{-7}{3}\right]$$

$$-1x_1 - 1x_2 \leq -3$$

Paso 6.

Se anexa al tableau y se aplica el dual simplex pivoteando en algún elemento del corte.

	x_1	x_2	x_3	x_4	S_1	LD
z	4	5	0	0	0	0
x_3	-1	-4	1	0	0	-5
x_4	-3	-2	0	1	0	-7
S_1	-1	-1	0	0	1	-3

Se pivotea usando el dual simplex y se obtiene:

TABLEAU 2

	x_1	x_2	x_3	x_4	S_1	LD	
z	0	+1	0	0	4	-12	
x_3	0	(-3)	1	0	-1	-2	←
x_4	0	+1	0	1	-3	+2	
S_1	1	1	0	0	-1	3	

como x_3 es el único renglón no factible es el renglón fuente y como $1 < 4$ la columna 2 es la columna pivote $u_2 = 1$ y $u_1 = 3$ y se obtiene el corte:

$$-x_5 - x_2 \leq -1$$

aumentando este a la tabla se tiene

TABLEAU 3

	x_2	S_1	S_2	LD
z	1	4	0	-12
x_3	-3	-1	0	-2
x_4	1	-3	0	2
S_1	1	-1	0	3
S_2	(-1)	-1	1	-1

pivoteando se obtiene:

TABLEAU 4

	S_1	S_2	LD
z	3	1	-13
x_1	-2	1	2
x_2	1	-1	1
x_3	2	-3	1
x_4	-4	1	-1

por lo tanto $z = -13$
$x_1 = 2$
$x_2 = 1$
$x_3 = x_4 = 1$
$x_5 = -3 + z_1 + x_2 \geq 0$
$x_6 = -4 + x_1 + 2x_2 \geq 0$

4.2 ALGUNAS PROPIEDADES AL AUMENTAR CORTES

4.2.1 CASO DUAL FRACCIONARIO

Considere la derivación del primer corte. En el tableau óptimo simplex existe un renglón i con x_{Bi} que no es entera. (De otra forma el programa entero ya está resuelto). La ecuación correspondiente es

$$x_i = x_{Bi} + \sum_{j=1}^{n} a_{ij} (-x_{J(j)})$$

donde $x_{Bi} > 0$. Ya que x_i se requiere que sea entero.

Teorema 4.1. Dado

$$\text{Max } z = cx$$

sujeto a

$$Ax = b$$
$$x \geq 0$$

se tiene que

$$\sum_{j=1}^{n} (a_{ij} - [a_{ij}])x_j \geq (x_{Bi} - [x_{Bi}])$$

Demostración. Dada una restricción

$$\sum_{j=1}^{n} a_{ij}\, x_j + x_{n+i} = x_{Bi}$$

se le puede multiplicar por $h \neq 0$ y por $[h] \neq 0$ arbitrario quedando respectivamente

$$\sum_{j=1}^{n} h\, a_{ij}\, x_j + h\, x_{n+i} = h\, x_{Bi} \qquad (4.1)$$

$$\sum_{j=1}^{n} [h]a_{ij}\, x_j + [h]\, x_{n+i} = [h]\, x_{Bi} \qquad (4.2)$$

De la expresión (4.1) y de la definición de máximo entero se tiene:

$$\sum [ha_{ij}]x_j + [h]x_{n+i} \leq [h]\, x_{Bi} \qquad (4.3)$$

$\therefore$ como la parte izquierda de la desigualdad es entera se tiene

$$\sum_{j=1}^{n} [ha_{ij}]x_j + [h]\, x_{n+i} \leq [hx_{Bi}] \qquad (4.4)$$

(4.4) menos (4.2) da

$$\sum_{j=1}^{n} [ha_{ij}]x_j - \sum_{j=1}^{n} [h]a_{ij}]x_j + [h]\, x_{n+i} - [h]x_{n+i} \leq [hx_{Bi}] - [h]x_{Bi}$$

$$\sum_{j=1}^{n} ([ha_{ij}] - [h]\, a_{ij})x_j \leq [h\, x_{Bi}] - [h]x_{Bi}$$

o $$\sum [h]a_{ij}] - [ha_{ij}]x_j \geq [h]x_{Bi} - [hx_{Bi}] \qquad (4.5)$$

como h es arbitrario, se hace h = 1 quedando (4.5) como:

$$\sum_{j=1}^{n} (a_{ij} - [a_{ij}])x_j \geq (x_{Bi} - [x_{Bi}])$$

La primer propiedad que podemos observar es que la desigualdad aumentada es equivalente a una desigualdad entera pura en las variables no básica originales. La segunda prueba que, si existe una solución entera siempre podemos obtener un tableau óptimo entero puro. El tableau inicial se supone que es entero puro.

Teorema 4.2 Cada desigualdad aumentada (el corte de Gomory) se convierte en una desigualdad entera pura cuando se expresa en términos de las variables no básicas originales.

Demostración.

Para ver esto, sea el renglón generado para la primer desigualdad

haciendo el cambio de variable x_{Bi} por a_{i0}

$$x_i = a_{i0} + \sum a_{ij}(-x_{J(j)})$$

entonces la restricción generada es

$$x_{m+n+1} = - f_{Bi} + \sum_{j=1}^{n} (-f_{ij})(-x_{J(j)})$$

$$= ([a_{i0}] - a_{i0}) + \sum_{j=1}^{n} ([a_{ij}] - a_{ij})(-x_{J(j)})$$

$$= \left\{ [a_{i0}] + \sum_{j=1}^{n} [a_{ij}](-x_{J(j)}) \right\} - \left\{ a_{i0} + \sum a_{ij}(-x_{J(j)}) \right\}$$

Ahora, cada variable no básica $x_{J(j)}$ es una variable no básica original o una variable básica original, en el primer caso tenemos

$$x_{J(j)} = x_g \quad \text{p.a.} \quad g = 1,\ldots,n$$

Y en el último caso, tenemos: $x_{J(j)} = a_{n+i,0} + \sum_{j=1}^{n} a_{n+i,j}(-x_j)$ p.a. $i = 1,\ldots,m$.

Pero los coeficientes en la última igualdad son enteros puros, ya que los datos originales se suponen enteros. Entonces, cada $x_{J(j)}$ se puede escribir como una combinación lineal entera pura de las variable no básica originales. Así, la primer expresión entre llaves del lado derecho da una expresión entera para cuando se escribe en términos de $x_1,\ldots,x_n$.

El segundo término entre llaves del lado derecho, es igual a x_i. Otra vez, esta variable es una variable original no básica o aparece como una combinación lineal entera de las variables no básicas originales. En cualquier caso, el segundo término da una expresión entera cuando se escribe en términos de $x_1,\ldots,x_n$. Por lo tanto $x_{m+n+1} \geq 0$ se convierte en una desigualdad entera pura cuando se expresa en términos de las variables originales no básicas.

Se puede emplear este razonamiento, para los cortes subsecuentes.

Teorema 4.3 Después de que se ha encontrado una solución óptima entera, podemos continuar aumentando cortes (derivados de los renglones que contienen fracciones) para obtener un tableau óptimo entero puro.

Demostración.

Primero observe que después que se encuentra un óptimo entero, los cortes subsecuentes, derivados de cualquier renglón i en el tableau que todavia contiene fracciones tienen un término $f_{i0} = 0$ ya que x_{Bi} es entero (esto es, aquellas $x_{Bi} \neq 0$ tienen su parte fraccionaria igual a cero). Consecuentemente sucesivos pivoteos dejaran a la columna 0 sin cambios, es decir que el tableau mantiene el punto óptimo entero.

El determinante $|B|$ (n+1 x n+1) de la base dual B es igual a el producto de menos valor de todos los pivotes precedentes. Pero en la fase post-óptima el negativo del elemento pivote es una fracción propia. Entonces $|B|$ es positiva y decrece después de cada pivoteo. También cada base B es una matriz entera ya que cada uno de sus renglones es el negativo de algún renglón de A^1 la matriz entera inicial, por lo tanto $|B|$ es entero. Así el pivoteo postóptimo terminará si el tableau se hace entero puro o si $|B|$ es igual a 1. Finalmente los dos casos coinciden, esto es si el tableau es entero puro $|B|$ es necesariamente igual a uno. Para ver esto, denote los elementos del k-ésimo tableau como la matriz A^k y escriba la base dual actual como B con inversa B^{-1}. Entonces, de la programación lineal tenemos que

$$A^k = A^1B^{-1}$$

ó $$A_1B = A^1$$

Ya que A^1 contiene la matriz negativa de identidad -I entonces deben existir renglones de A^k que contiene a la matriz cuadrada C tal que $CB = -I$ o $-C = B^{-1}$.

Si A^k es entero puro se sigue que $-C$ o B^{-1} es entero puro y entonces $|-C| = |B^{-1}| = d$ donde d es algún entero. Pero $|B|$ es un entero positivo d_1 y como $|B| = 1/|B_{-1}|$ entonces se tiene $d_1 = 1/d$. Pero d y d_1 son enteros, entonces $d = d_1 = 1$.

Por otro lado, si $|B| = 1$ tenemos (usando $B^{-1} = \frac{\text{adj } B}{|B|}$) que

$$A^k = A^1 B^{-1} = A^1 \frac{\text{adj}(B)}{|B|} = A^1 \text{ adj}(B)$$

es una matriz entera, ya que B es entera lo que implica que adj(B) es entera.

Ejemplo 4.2

Del ejemplo que vimos

$$\max \quad z = -4x_1 - 5x_2$$

sujeto a

$$\begin{aligned} -x_1 - 4x_2 &\leq -5 \\ -3x_1 - 2x_2 &\leq -7 \end{aligned}$$

$$x_1, x_2 \geq 0, \quad x_1, x_2 \in z$$

se tiene la solución entera

	$1x_5$	$1x_6$	LD
z	7/4	1/2	-13
x_1	-3/4	1/2	2
x_2	1/4	-1/2	1
x_3	1/4	-3/2	1
x_4	-7/4	1/2	1
x_5	1	0	0
x_6	0	1	0
x_7	-3/4	-2/4	0

$z = -13 \qquad x_1 = 2 \qquad x_2 = x_3 = x_4 = 1$

La matriz inicial A^1 del tableau 1 se exhibe a continuación donde se han incluido las variables de holgura de Gomory x_5, x_6, x_7 y x_8 en términos de las variables no básicas originales y la ecuación trivial $-1 = -1(1)$ para tomar en cuenta la función original dual.

El dual del programa lineal asociado se puede escribir como

minimizar $w_0 = -5w_1 + 2w_2 - 6w_2 - 5w_4 - 7w_5 - 3w_6$

sujeto a

$$w_1 + 3w_2 + 2w_3 + w_4 + 2w_5 + w_6 + w_7 = 4$$

$$4w_1 + 2w_2 + 2w_3 + 3w_4 + 3w_5 + w_6 = 5$$

$$w_i \geq 0$$

en el tableau 1, w_0, w_3 y w_4 son básicas, entonces escribimos sus coeficientes como renglones y obtenemos

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad |B| = 1$$

TABLEAU 1

	1	$(-x_1)$	$(-x_2)$
-1	-1	0	0
z	0	4	S
x_1	0	-1	0
x_2	0	0	-1
x_3	-5	-1	-4
x_4	-7	-3	-2
x_5	-6	-2	-2
x_6	-5	-1	-3
x_7	-7	-2	-3
x_8	-3	-1	-1

A^1

Usando la notación de Beale

Columnas	α_0	α_1	...	α_n	
variables	1	$(-x_1)$	...	$(-x_n)$	
$x_0 =$	a_{00}	a_{01}		a_{0n}	
$x_1 =$	0	-1		0	≥ 0
$x_2 =$	0	...	-1	0	≥ 0
$x_n =$	0	. . . . .		-1	≥ 0
$x_{n+1} =$	$a_{n+1,0}$	$a_{n+1,1}$		$a_{n+1,n}$	≥ 0
.					
.					
.					
$x_{n+m} =$	$a_{n+1,0}$	$a_{n+m,1}$		$a_{n+m,n}$	≥ 0

J(j) es el j-ésimo índice en el conjunto actual de índices J correspondientes a las variables no básicas α_j (j=0, 1,...,n) son las primeras m+n+1 componentes en la columna j del tableau.

$$x_5 = -\frac{8}{10} + \frac{6}{10}x_4 + \frac{2}{10}x_3 \Rightarrow x_5 = -6 + 2x_1 + 2x_2$$

$$x_5 = -6 + 2x_1 + 2x_2 \qquad o \qquad x_1 + x_2 \geq 3$$

$$x_6 = -5 + x_1 + 3x_2$$

$$x_6 \Rightarrow \quad x_1 + 3x_2 \geq 5$$

$$x_7 = 0 - \frac{3}{4}(-x_5) - \frac{2}{4}(-x_6)$$

$$x_7 = -\frac{9}{2} + \frac{3}{2}x_1 + \frac{3}{2}x_2 - \frac{5}{2} + \frac{1}{2}x_1 + \frac{3}{2}x_2$$

$$x_7 = -7 + 2x_1 + 3x_2$$

$$x_8 = \frac{1}{2}x_5 \Rightarrow x_8 = -3 + x_1 + x_2$$

En el tableau 1 w_0, w_7 y w_8 son variables básicas, entonces los coeficientes de sus renglones se pueden arreglar en

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad y \quad |B| = 1$$

En el tableau óptimo x_5 y x_6 son no básicas (esto es w_0, w_3 y w_4 son variables básicas para el dual), otra vez, escribiendo sus coeficientes en un arreglo de renglón *

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 6 & 2 & 2 \\ 5 & 1 & 3 \end{bmatrix}$$

y $|B| = 4$ que es el producto de los negativos de los pivotes precedentes, esto es

$$4 = (4/6)\ (6/10)\ (10/3)(3)$$

*) Para ver explícitamente que la base dual tiene estos valores note que en el tableau 5 w_3 y w_4 son variables básicas duales. Esto se sigue porque el método dual simplex mantiene la holgura complementaria. Esto es, $x_j\ w_{5j} = 0\ V\ j\ (j \neq 0)$ y $w_i\ x\ si = 0\ V\ i\ i \neq 0)$ donde $w_i(w_{5j})$ es la i-ésima variable dual (j-ésima de holgura) y x_j(x si) es la j-ésima variable primal (i-ésima de holgura). Para mantener estas igualdades el algoritmo dual simplex aporta las variables (del ejemplo) como sigue:

	original		holgura					
Primal	x_1	x_2	x_3	x_1	x_5	x_6	x_7	x_8
Dual	w_7	w_2	w_1	w_2	w_3	w_4	x_5	w_6
	holgura		original					

Esto es, cualquier tableau simplex contiene un par de soluciones básicas primales y duales, donde si una variable primal es no básica (básica) la variable dual correspondiente es básica (no básica). Esto forza a la holgura complementaria. En nuestro ejemplo, como x_5 y x_6 son no básicas el dual simples asigna que w_3 y w_4 son básicas. Incluyendo la función original dual w_0 como poner variable básica, los coeficientes de (w_0, w_3, w_4) escritos como renglones producen

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 6 & 2 & 2 \\ 5 & 1 & 3 \end{bmatrix}$$

se lee en los renglones de

z	0	0	1
x_5	2	2	6
x_6	1	3	5

$x_5 = -6 + 2x_1 + 2x_2$

$x_6 = -5 + x_1 + 3x_2$

En el tableau (#7) entero puro x_8 y x_7 son variables no básicas o

# 7	$-x_8$	x_7	LD
z	2	1	-13
x_1	-3	1	2
x_2	2	-1	1
x_3	5	-3	1
x_4	-5	1	1
x_5	-2	0	0
x_6	3	-2	0
x_7	0	-1	0
x_8	-1	0	0

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 1 \\ 7 & 2 & 3 \end{bmatrix}$$

$|B| = 1 = 2\ (1/2)$

$x_7 = 0\ -3/4)(-x_5) - (2/4(-x_6)$

$$x_8 = -\frac{1}{2}(-x_5)$$

$$x_8 = -\frac{1}{2}(+6 - 2x_1 - 2x_2)$$

$x_8 = -3 + x_1 + x_2$

$$x_7 = 0 - \frac{3}{4}(+ 6 - 2x_1 - 2x_2) - \frac{1}{2}(5 - x_1 - 3x_2)$$

$$x_7 = 0 - \frac{18}{4} + \frac{3}{2}x_1 + \frac{3}{2}x_2 - \frac{5}{2} + \frac{1}{2}x_1 + \frac{3}{2}$$

$x_7 = -7 + 2x_1 + 3x_2$

Note que:

$$\underset{A^7}{\begin{bmatrix} -1 & 0 & 0 \\ -13 & 2 & 1 \\ 2 & -3 & 1 \\ 1 & 2 & -1 \\ 1 & 5 & -3 \\ 1 & -5 & 1 \\ 0 & -2 & 0 \\ 0 & 3 & -2 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix}} \quad \underset{B}{\begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 1 \\ 7 & 2 & 3 \end{bmatrix}} = \underset{A^1}{\begin{bmatrix} -1 & 0 & 0 \\ 0 & 4 & 5 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ -5 & -1 & -4 \\ -7 & -3 & -2 \\ -6 & -2 & -2 \\ -5 & -1 & -3 \\ -7 & -2 & -3 \\ -3 & -1 & -1 \end{bmatrix}}$$

Previamente usamos el hecho de que el arreglo A^k correspondiente al k-ésimo tableau se obtiene postmultiplicando la matriz entera pura A^1 de los datos originales por la inversa de la base dual B^{-1} asociada con el tableau K. Podemos usar también esto para demostrar que todo elemento en A^k se puede expresar como $L/|B|$ donde L es matriz entera. Para ver esto, sea

$$A^k = A^1B^{-1} = A^1 \frac{\text{adj (B)}}{|B|}$$

pero B es entera ya que está hecha de renglones negativos de A^1. Por lo tanto adj(B) es entera o A^1 adj (B) es una matriz entera. Entonces, cada elemento en el tableau actual se puede escribir como una fracción cuyo numerador es un entero y cuyo denominador es el determinante de la base actual. Este hecho se puede extender a la matriz $F(A^k)$ cuyos renglones son las partes fraccionales de los renglones correspondientes de A^k. Esto es, un elemento de $F(A^k)$, f_{ij} es igual a $a_{ij} - [a_{ij}]$ entonces

$$f_{ij} = a_{ij} - [a_{ij}] = \frac{L}{|B|} - \frac{[a_{ij}]|B|}{|B|} = \frac{L - [a_{ij}]|B|}{|B|}$$

se tiene que cada f_{ij} se puede escribir como un entero dividido por $|B|$ más aún, como $|B|$ es un entero positivo y f_{ij} es una fracción propia se sigue que $f_{ij} = I_j/|B|$ donde I_j es el menor entero no negativo que $|B|$. Entonces el corte de Gomory

$$\sum_{j=1}^{n} f_{ij}\, x_{J(j)} \geq f_{i0}$$

se puede expresar como:

$$\sum_{j=1}^{n} \frac{I_j}{|B|}\, x_{J(j)} \geq \frac{J_0}{|B|}$$

donde I_j $(j=1,\ldots,n)$ es un entero no negativo menor que $|B|$ e I_o es un entero positivo que $0 < f_{i0} < |B|$. Usamos esta versión del corte para escribir el siguiente teorema.

Teorema 4.4. Considere el hiperplano

$$\frac{I_0}{|B|} = \sum_{j=1}^{n} \frac{I_j}{|B|}\, x_{J(j)}$$

correspondiente a una desigualdad aumentada. Entonces el hiperplano pasa por al menos un punto entero, (no necesariamente dentro de la región factible) si y sólo si el m.c.d. de $I_1,\ldots,I_n$ divide I_0.

4.2.2 CASO DEL ALGORITMO ENTERO PURO.

La forma del corte:

Sea el renglón generador cualquier renglón r del tableau con $a_{r0} < 0$

$$x_r = a_{r0} + \sum a_{rj}\,(-x_{J(j)}) \qquad (4.6)$$

entonces el corte entero puro es:

$$x' = \left[\frac{a_{r0}}{\lambda}\right] + \sum_{j=1}^{n} \left[\frac{a_{ij}}{\lambda}\right] (-x_{J(j)}) \geq 0 \qquad (4.7)$$

donde x′ es una variable de holgura de Gomory no negativa y λ es un número positivo que se calcula como ya se describió.

La desigualdad (4.7) cumple

$$\left[\frac{a_{r0}}{\lambda}\right] < 0$$

o el renglón generador es un legítimo renglón pivote.

También observe que

$$-1 \leq \frac{a_{rp}}{\lambda} < 0 \qquad a_{rp} < 0$$

y $\lambda \geq \lambda_p = -a_{rp}$.

por lo tanto el elemento pivote es

$$\left[\frac{a_{rp}}{\lambda}\right] = -1$$

Note que el pivote siempre es -1 o equivalentemente, el tableau simplex permanece entero puro y lexicográficamente dual factible.

Designe cualquier renglón (o combinación lineal entera de renglones) como:

$$x = a_0 + \sum_{j=1}^{n} a_j (-x_{J(j)}) \qquad (4.8)$$

con $a_0 < 0$ como el renglón fuente. Si ninguno existe el tableau es entero óptimo para tener la forma del corte, representamos los términos en (4.8) incluyendo el coeficiente 1 de x como un producto mas un residuo.

Sea a cualquier número y λ un número positivo. Sea

$$f = \frac{a}{\lambda} - \left[\frac{a}{\lambda}\right]$$

entonces $0 \leq f < 1$. Sea $r = \lambda f$ entonces

$$\frac{r}{\lambda} = \frac{a}{\lambda} - \left[\frac{a}{\lambda}\right]$$

o

$$a = \left[\frac{a}{\lambda}\right]\lambda + r$$

como $0 \leq f < 1$ y $\lambda > 0$ entonces $0 \leq r < \lambda$, y relacionando los coeficientes con (4.8) se tiene:

$$a_j = \left[\frac{a_j}{\lambda}\right]\lambda + r_j \qquad j = 0, 1, \ldots, n \qquad (4.9)$$

y

$$1 = \left[\frac{1}{\lambda}\right]\lambda + r \qquad (4.10)$$

donde $\lambda > 0$, $0 \leq r_j < \lambda$ y $r < \lambda$ sustituyendo (4.9) y (4.10) en (4.8) se tiene:

$$\left\{\left[\frac{1}{\lambda}\right]\lambda + r\right\} x = a_0 + \sum_{J=1}^{n}\left\{\left[\frac{a_j}{\lambda}\right]\lambda + r_j\right\}(-x_{J(j)}) \geq 0$$

$$\left[\frac{1}{\lambda}\right]\lambda x + rx = \left[\frac{a_0}{\lambda}\right]\lambda + r_0 + \left\{\sum_{j=1}^{n}\left[\frac{a_j}{\lambda}\right]\lambda + r_j\right\}(-x_{J(j)} \geq 0$$

$$\left|\frac{1}{\lambda}\right|\lambda x + rx = \left[\frac{a_0}{\lambda}\right]\lambda + r_0 + \left\{\sum_{j=1}^{n}\left[\frac{a_j}{\lambda}\right]\lambda\ (-x_{J(j)})\right\} + \sum_{j=1}^{n} rj$$

$$(-x_{J(j)}) \geq 0$$

Se ponen los residuos del lado izquierdo excepto r_0.

(4.11)

$$rx + \sum_{j=1}^{n} r_j x_{J@} = r_0 + \lambda\left\{\left[\frac{a_0}{\lambda}\right] + \sum_{j=1}^{n}\left[\frac{a_j}{\lambda}\right](-x_{J@}) + \left[\frac{1}{\lambda}\right](-x)\right\}$$

los términos en (4.11) con llaves los denotamos por x′ esto es

$$x' = \left[\frac{a_0}{\lambda}\right] + \sum_{j=1}^{n}\left[\frac{a_j}{x}\right](-x_{J@}) + \left[\frac{1}{\lambda}\right](-x) \qquad (4.12)$$

entonces para cualquier solución entera no negativa que satisfaga (4.8) y ∴ (4.11) por demostrar $x' \geq 0$ $x' \epsilon Z$

Claramente, para valores enteros de las variables tendremos x′ entera ya que todos los coeficientes en (4.12) son enteros. Para ver que x′ debe ser no negativa es necesario examinar (4.12) para cualquier solución factible el lado izquierdo de la igualdad es no negativo. Suponga al mismo tiempo que x′ es un entero negativo, es decir, x′ es -1, -2, -3 etc. como $r_0 < \lambda$ se tendría que el lado izquierdo de (4.11) $r_0 + \lambda x'$ es negativo lo que es imposible, por lo tanto x′ definido como en (4.12) es una variable entera no negativa. Mas aún cuando $\lambda > 0$

$$\left[\frac{1}{\lambda}\right] = 0 \quad \text{y (7) se reduce a}$$

$$x' = \left[\frac{a_0}{\lambda}\right] + \sum_{j=1}^{n}\left[\frac{a_j}{\lambda}\right](-x_{J@}) \geq 0 \qquad (4.13)$$

que $a_0 < 0$ significa que:

$$\left[\frac{a_0}{\lambda}\right] < 0$$

la desigualdad (4.13) se puede agregar al tableau y usar como renglón pivote y para λ suficientemente grande el elemento pivote se garantiza que sea -1 por ejemplo

$$\lambda \geq \max_{a_j<0} |a_j| \Rightarrow \left[\frac{a_j}{\lambda}\right] = -1 \quad \text{para toda } a_j < 0$$

Selección de la columna pivote:

Si los coeficientes en el renglón pivote se denotan por b_j ($j = 0,1,\ldots,n$) la columna pivote α_p con el dual simplex lexicográfico satisface

$$\frac{\alpha_p}{-b_p} \overset{\mathcal{L}}{<} \frac{\alpha_j}{-b_j} \tag{4.14}$$

para toda $_j$ ($j \neq 0$) con $b_j < 0$ con referencia al algoritmo entero puro, tenemos $b_p = -1$ y $b_j = [a_j/\lambda]$ entonces por (4.14) la columna pivote α_p debe satisfacer

$$\alpha_p \overset{\mathcal{L}}{<} \frac{\alpha_j}{-\left[\frac{a_j}{\lambda}\right]}$$

para toda j ($j \neq 0$) con $\frac{a_j}{\lambda} < 0$ como $-\left[\frac{a_j}{\lambda}\right]$ es un entero positivo

tenemos:

$$\alpha_p \overset{\mathcal{L}}{<} \frac{\alpha_j}{-\left[\frac{a_j}{\lambda}\right]} \overset{\mathcal{L}}{\leq} \alpha_j \tag{4.15}$$

Entonces hemos probado que la columna pivote es lexicográficamente la menor columna que contiene un elemento negativo en el renglón pivote y por lo tanto el renglón fuente. Ahora podemos seleccionar λ mas ventajosamente.

Reglas de selección de λ

El objetivo cuando se selecciona λ es obtener un pivote -1 y también producir un renglón pivote que resultará en un mayor decrecimiento lexicográfico en la columna 0. El último objetivo es importante porque aumentará la tasa de convergencia. Suponga que varios valores de λ dan un pivote -1. Entonces el menor de esos valores para λ se debe seleccionar, ya que resulta en un mayor decrecimiento lexicográfico en α_0. Esto se sigue de que la columna

0 que se abandona es

$$\alpha'_0 = \alpha_0 + \left[\frac{a_0}{\lambda}\right]\alpha'_p$$

y $[a_0/\lambda]$ es un entero negativo cuyo valor absoluto se incrementa tanto como λ decrece. Usando este resultado y los 2 objetivos podemos explicar la necesidad de los cálculos para encontrar λ.

Si λ_p es la columna pivote se tiene por (4.15)

$$\alpha_p < \min \frac{\alpha_j}{-\left[\frac{a_j}{\lambda}\right]} \qquad (4.16)$$

para toda $_j$ $(j \neq 0)$ con $a_j < 0$. Sea $u_p = 1$ y sea u_j $(j \neq p)$ el mayor entero para el dual

$$\alpha_p < \min \frac{\alpha_j}{u_j} \qquad (4.17)$$

Como $-\left[\frac{a_j}{\lambda}\right]$ es un entero en (4.16) y u_j es el mayor entero en (4.17) tenemos:

$$-\left[\frac{a_j}{\lambda}\right] \leq u_j \qquad (4.18)$$

El menor positivo λ que satisface (4.18) es

$$\lambda_j = -\frac{a_j}{u_j} \qquad (4.19)$$

entonces para garantizar (4.16) o equivalentemente para mantener las columnas α_j (j=1,...,n) lexicográficamente positivas tenemos

$$\lambda \geq \max \lambda_j \qquad (4.20)$$

El mínimo λ que satisface (4.20) es $\lambda = \max \lambda_j$. Entonces λ que satisfaga los 2 objetivos, requiere que:

1. Sea α_p la columna lexicográficamente menor con $a_p < 0$ ($p \neq 0$).

2. Para cada $a_j < 0$ ($j \neq 0$). Sea $u_p = 1$ y sea u_j ($j \neq p$) el mayor entero que satisface (4.17).

3. Para cada $a_j < 0$ ($j \neq 0$) encuentre λ_p tal que

$$\lambda_p = - \frac{a_p}{u_p}$$

4. Sea $\lambda = \max \lambda_j$.

Observaciones

1. Como las ecuaciones $x_j = -1(-x_j)$ (j=1,...,n) se usaron, las columnas 1 a n en el tableau simplex son linealmente independientes. Por lo tanto la columna pivote α_p está unicamente determinada.

2. $\lambda_p \geq 1$ como $\lambda_p = - \frac{a_p}{u_p} = - a_p \geq 1$

3. Si $\lambda = 1$ el renglón fuente es el renglón pivote, y se agrega una nueva desigualdad (por supuesto, el elemento pivote $a_p = -1$).

4. Si el primer elemento en α_p es cero y el primer elemento en α_j no es cero, entonces u_p ($j \neq p$) es arbitrariamente grande. En este caso, λ_j ($j \neq p$) es arbitrariamente pequeño y se puede tomar como 0, ya que λ_p siempre será el mayor.

5. Si $a_p \leq a_j < 0$ entonces $\lambda_p \geq \lambda_j$ ya que

$$\lambda_p = - a_p \geq - a_j - \frac{a_j}{u_j} = \lambda_j$$

Así, λ_p no necesita calcularse siempre que $a_p = a_j < 0$

6. Por esta selección λ no necesita ser entero

4.3 LA FUERZA RELATIVA AL AUMENTAR DESIGUALDADES

La selección de λ como se describió previamente produce un pivote -1 y también resulta en cambios mayores en la columna 0. Sin embargo puede suceder que no sea mas fuerte. De hecho, puede ser mas débil que el renglón (o desigualdad) de la que se deriva. Para ilustrarlo, suponga que el renglón fuente es:

$$x = - 4 - 3 (-x_1) - 5(-x_2) \geq 0 \qquad (4.21)$$

la forma del corte es:

$$x' = \left[\frac{a_{v0}}{\lambda} \right] + \sum_{j=1}^{n} \left[\frac{a_{vj}}{\lambda} \right] (-x_{j(j)}) \geq 0$$

para $\lambda = 2$ se tiene:

$$x' = -\left[\frac{4}{2} \right] + \left[\frac{-3}{2} \right] (-x_1) + \left[\frac{-5}{2} \right] (-x_2) \geq 0$$

$$x' = [-2] + [-1.5] (-x_1) + [-2.5](-x_2) \geq 0$$

$$x' = -2 + 2x_1 + 3x_2 \geq 0 \qquad (4.22)$$

para $\lambda = 3$ se tiene

$$x' = \left[- \frac{4}{3} \right] + \left[\frac{3}{3} \right] (-x_1) + \left[- \frac{5}{3} \right](-x) \geq 0$$

$$x' = -2 + x_1 + 2x_2 \geq 0 \qquad (4.23)$$

para $\lambda = 4$ se tiene

$$x' = -\left[\frac{4}{4}\right] + -\left[\frac{3}{4}\right](-x_1) + -\left[\frac{5}{4}\right](-x_2) \geq 0$$

$$x' = -1 + x_1 + 2x_2 \geq 0 \qquad (4.24)$$

Si las vemos geométricamente, en la figura 4.1

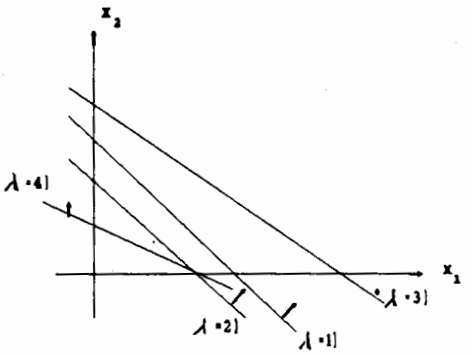


Figura 4.1

podemos observar que, la desigualdad (4.21) es más fuerte que la (4.22) y a su vez la (4.22) que la (4.24), sin embargo es más débil que la (4.23).

De aquí podemos concluir que algunas veces es posible incrementar el valor de λ, encontrado por los cálculos usuales para así obtener un corte entero puro mas fuerte. Considere la desigualdad de la cual podemos extraer un corte (donde omitimos los subíndices de los renglones)

$$x' = \left[\frac{a_0}{\lambda}\right] + \sum_{j=1}^{n} \left[\frac{a_j}{\lambda}\right] (-x_{J@}) \geq 0 \qquad (4.25)$$

Entonces si λ se puede incrementar tal que $[a_0/\lambda]$ y $[a_j/\lambda]$ para $a_j > 0$ no cambian, tendremos una desigualdad mas fuerte si alguna o todas las $[a_j/\lambda]$ restantes ($a_j < 0$) cambian. Esto es cierto porque el mayor λ es el menor en valor absoluto en los coeficientes en (5). Así, un cambio en $[a_j/\lambda]$ ($a_j < 0$) pone el hiperplano $x' = 0$ mas lejos del origen en el espacio $x_{J@}$, lo que da como resultado, una desigualdad mejorada. Para ver esto note que la intersección del hiperplano con el eje $x_{J@}$ ocurre cuando

$$x_{J@} = \frac{\left\lceil \frac{a_0}{\lambda} \right\rceil - \frac{2}{3}}{\left\lceil \frac{a_j}{\lambda} \right\rceil \quad \frac{-5}{3}} = \frac{-2}{-5} \quad \frac{-1}{-5} \quad \frac{-1}{-2}$$

como $[a_0/\lambda] < 0$ tenemos, cuando $a_j < 0$, $[a_0/\lambda]/[a_j/\lambda]$ se incrementa cuando λ se incrementa. Como $[a_0/\lambda]$ permanece igual, la desigualdad mas fuerte produce el mismo cambio en la columna 0 que la desigualdad estándar.

Para ilustrar esto, considere el tableau parcial siguiente, donde x_0 está en la función objetivo y x es el renglón fuente.

	1	$(-x_1)$	$(-x_2)$	$(-x_3)$	$(-x_4)$
x_0	20	1	3	4	4
x	-20	-7	-8	-15	18

Entonces $\alpha_p = \alpha_1$, $u_1 = 1$, $u_2 = 2$, $u_3 = 3$ y $\lambda_1 = 7$, $\lambda_2 = 4$, $\lambda_3 = 5$ o $\lambda = 7$. Entonces el corte de Gomory es:

$$x' = -3 - 1(-x_1) - 2(-x_2) - 3(-x_3) + 2(x_4) \geq 0 \quad (\lambda = 7)$$

Para $7 \leq \lambda < 10$ el valor de $[-20/\lambda]$ será -3 y para $7 \leq \lambda \leq 9$ tenemos $[18/\lambda] = 2$. Así, si tomamos

$$\lambda = \min (9, 10 - \epsilon) = 9 \quad (\epsilon > 0 \text{ suficientemente pequeño})$$

tenemos el corte más fuerte

$$x' = -3 - 1(-x_1) - 1(-x_2) - 2(-x_3) + 2(-x_4) \geq 0 \quad (\lambda = 9)$$

Una segunda derivación del corte fraccional y su relación con el corte entero puro.

Durante la derivación del corte entero puro, detuvimos la ecuación ($\lambda > 0$)

$$x^* = \left[\frac{a_0}{\lambda}\right] + \sum_{j=1}^{n} \left[\frac{a_j}{\lambda}\right] (-x_{J@}) + \left[\frac{1}{\lambda}\right](-x) \qquad (4.26)$$

donde el renglón fuente es

$$x = a_0 + \sum_{j=1}^{n} a_j \ (-x_{J@}); \qquad (4.27)$$

x^* es una variable entera no negativa. El corte entero puro se obtuvo de seleccionar $\lambda > 1$. Como x^* es un entero no negativo para cualquier número λ positivo, tomando $\lambda = 1 \Rightarrow 1/\lambda = 1$ y usando (7) la desigualdad (6) se convierte en:

$$x^* = [a_0] + \sum_{j=1}^{n} [a_j] \ (-x_{J@}) - a_0 + \Sigma \ a_j(-x_{J@})$$

reordenando términos se tiene

$$x^{\bullet} = [a_0] - a_0 + \sum_{j=1}^{n} ([a_j] - a_j) \quad (-x_{J@})$$

o bien

$$x^{\bullet} = - f_0 + \sum_{j=1}^{n} (-f_j) \ (-x_{J@})$$

de donde podemos observar, que ésta última desigualdad es el corte fraccionario de Gomory que se obtuvo en forma distinta originalmente. Entonces, el fraccionario ($\lambda = 1$) y el entero puro ($\lambda > 1$) se pueden obtener usando la misma derivación.

4.4 ALGORITMO DUAL FRACCIONARIO MIXTO.

El siguiente algoritmo para un programa entero mixto, es una extensión directa del algoritmo de planos de corte fraccionario. Como el anterior utiliza el método dual simplex y permite números fraccionarios en sus cálculos y no requiere que el primer tableau sea entero puro.

Paso 1. Resuelva el programa entero mixto como uno lineal. Si no es factible entonces el programa entero mixto tampoco lo es, termine. Si la solución óptima es entera en las variables restringidas a ser enteras el programa está resuelto, termine. De otra forma vaya al paso 2.

Paso 2. Desde un renglón correspondiente a una variable restringida entera que no tiene un valor entero, derive una nueva desigualdad que corte el punto óptimo actual pero que no elimine cualquier solución mixta entera. Aumente esta nueva desigualdad al final del tableau simplex que entonces exhibe una infactibilidad primal (la variable de holgura asociada con el nuevo renglón será negativa).

Paso 3. Reoptimice usando el método dual simplex lexicográfico si el nuevo programa lineal no satisface (el dual no será acotado), el problema entero mixto no tiene solución, termine. Si la nueva solución óptima es entera en las variables restringidas a enteras, el programa entero mixto está resuelto - termine. De otra forma vaya al paso 2.

La forma del corte.

En el tableau simplex óptimo $a_{0j} \geq 0$ $(j=1,\ldots,n)$ $a_{i0} \geq 0$ $(i=1,\ldots,n)$ $a_{i0} \geq 0$ $(i=1,\ldots,n+m)$ y $\alpha_j \geq 0 (j=1,\ldots,n)$. Más aún, si a_{i0} $(i=0,1,\ldots,I)$ es entero, el programa entero mixto está resuelto. De otra forma, hay un renglón fuente v $(0 \leq v \leq I)$ con

$$x_v = a_{v0} + \sum_{j=1}^{n} a_{vj} \, (-x_{J(j)})$$

y a_{v0} fraccionario. Entonces el k-ésimo corte de Gomory tiene la forma

$$x_{n+m+k} = - f_{v0} + \sum_{j=1}^{n} (-g_{vj}) \, (-x_{J(j)}) \geq 0$$

donde

$$g_{vj} = \begin{cases} a_{vj} & \text{Si } a_{vj} \geq 0 \text{ y } x_{J(j)} \text{ es una variable continua} \\ \dfrac{f_{v0}}{f_{v0}-1} \, a_{vj} & \text{Si } a_{vj} < 0 \text{ y } x_{J(j)} \text{ es una variable continua} \\ f_{vj} & \text{Si } f_{vj} \leq f_{v0} \text{ y } x_{J(j)} \text{ es una variable entera} \\ \dfrac{f_{v0}}{1-f_{v0}} \, (1-f_{vj}) & \text{Si } f_{vj} > f_{v0} \text{ y } x_{J(j)} \text{ es una variable entera.} \end{cases}$$

y $f_{vj} = a_{vj} - [a_{vj}]$ $(j=0,1,\ldots,n)$. Observe que $0 < f_{v0} < 1$ y así cuando la desigualdad se agrega al final del tableau, se introduce una infactibilidad primal. La variable de Gomory $x_{m+n+k} - f_{v0} < 0$.

Ejemplo 4.3

$$\max \quad x_0 = -4x_1 - 5x_5$$

sujeto a

$$\begin{aligned} -x_1 - 4x_2 &\leq -5 \quad (x_3) \\ -3x_1 - 2x_2 &\leq -7 \quad (x_4) \end{aligned}$$

$$x_0 \, , \, x_1 \in Z \qquad x_1, \; x_2 \geq 0$$

El programa lineal se puede escribir en la forma estándar aumentando las variables de holgura x_3 y x_4. Este programa ya fue resuelto y el primer tableau óptimo está dado por

TABLEAU 3

	1	$(-x_4)$	$(-x_3)$
x_0	-112/10	11/10	7/10
x_1	18/10	-4/10	2/10
x_2	8/10	1/10	-3/10
x_3	0	0	-1
x_4	0	-1	0
x_5	-8/10	-16/10	-2/10

derivación del corte.

$$f_{10} = \frac{18}{10} - \left[\frac{18}{10} \right] = \frac{8}{10}$$

$$g_{11} = \frac{8/10}{(8/10-1)} \quad (-4/10)$$

$$g_{12} = 2/10$$

$$x_5 = -\frac{8}{10} - \frac{16}{10}(-x_4) - \frac{2}{10}(-x_3)$$

TABLEAU 4

	1	$(-x_5)$	$(-x_3)$
x_0	-188/16	11/16	9/16
x_1	2	-4/16	4/16
x_2	12/16	1/16	-5/16
x_3	0	0	-1
x_4	8/16	-10/16	2/16
x_5	0	-1	0
x_6	-4/16	-11/16	-9/16

derivación del corte

$$f_{00} = -188/16 - [188/10] = 4/16$$

$$g_{01} = 11/16$$

$$g_{02} = 9/16$$

$$x_6 = -\frac{4}{16} - \frac{11}{16}(-x_5) - \frac{9}{16}(-x_3)$$

TABLEAU 5

	1	$(-x_6)$	$(-x_3)$
x_0	-12	1	0
x_1	23/11	-4/11	5/15
x_2	8/11	1/11	-4/11
x_3	0	0	-1
x_4	8/11	-10/11	7/11
x_5	4/11	-16/11	9/11
x_6	0	-1	0
x_7	-1/11	-4/110	-5/11

derivación del corte

$$f_{10} = \frac{23}{11} - \left[\frac{23}{11}\right] = \frac{1}{11}$$

$$f_{11} = \frac{1/11}{(1/11-1)}(-4/11) = \frac{4}{110}$$

$$g_{12} = \frac{5}{11}$$

$$x_7 = -\frac{1}{11} - \frac{4}{110}(-x_6) - \frac{5}{11}$$

TABLEAU 6

	1	$(-x_6)$	$(-x_7)$	tableau óptimo
x_0	-12	1	0	
x_1	2	-2/5	1	
x_2	4/5	6/50	-4/5	
x_3	1/5	4/50	-11/5	
x_4	3/5	-48/50	7/5	
x_5	1/5	-76/50	9/5	
x_6	0	-1	0	
x_7	0	0	-1	

El tableau 6 despliega la solución entera mixta óptima $x_0 = -12$ $x_1 = 2$ $x_2 = 4/5$. Expresando las desigualdades aumentadas en términos de las variables originales no básicas x_1 y x_2 se obtiene:

$$x_5 = -13 + 5x_1 + 4x_2 \geq 0$$
$$x_6 = -12 + 4x_1 + 5x_2 \geq 0$$
$$x_7 = -14/5 + (3/5)x_1 + 2x_2 \geq 0$$

Observe que las variables de holgura de Gomory no son necesariamente una combinación lineal entera de las variables originales no básicas. Entonces no tienen que ser enteras en toda solución entera mixta. Por lo tanto, como esas restricciones (renglones) no son elegibles para generar desigualdades es razonable adaptar la estrategia de goteo a la variable de holgura de Gomory y definir el renglón inmediatamente después de que las variables de holgura correspondientes a la columna pivote.

Algoritmos de planos de corte compuestos usan ambos, los cortes fraccionarios y los enteros puros. Por ejemplo, se podría usar primero el método simplex para obtener -al menos cercanamente- una solución óptima continua. Entonces desde el punto en que estemos, dependerá de cual es aplicable el fraccional o el entero puro. (A menos que el esté disponible tableau sea entero puro y óptimo en al menos uno).

4.5 NOTAS HISTORICAS

La idea de generar restricciones fue propuesta por Dantzig, Fulkerson y Johnson en 1954 en su trabajo del problema del agente viajero, posteriormente en 1957 Markowitz y Manne lo propusieron también. Sin embargo, no fue hasta 1958 que Gomory desarrolló el primer algoritmo de planos de corte aplicable a cualquier programa entero. Poco tiempo después junto en Beale generalizó sus resultados al caso mixto entero. En 1960 Gomory propuso un segundo algoritmo de planos de corte para el programa entero que requiere sólo sumas y restas en sus cálculos (una técnica "completamente entera"). Todos estos métodos mantienen programas líneas que son duales factibles y a menudo se clasifican como algoritmos de planos de corte duales.

Glover y Yong desarrollaron algoritmos de planos de corte para el programa entero que mantienen problemas lineales que son primales factibles. Como las soluciones enteras que se producen sucesivamente sin primales factibles, la técnica se conoce como algoritmo de planos de corte primales.

CAPITULO 5

LA TECNICA DE RAMIFICACION Y ACOTAMIENTO: ENFOQUE CLASICO

El segundo (principio), era la división de cada una de las dificultades con que tropieza la inteligencia al investigar la verdad, en tantas partes como fuera necesario para resolverlas. Y el último (principio), consistía en hacer enumeraciones tan completas y generales, que me dieran la seguridad de no haber incurrido en ninguna omisión"
Descartes, Discurso del Método, Vol II.

El procedimiento de ramificación y acotamiento (R-A) proporciona una metodología de búsqueda de la solución óptima en un problema de optimización discreta. En el método de R-A, el conjunto de soluciones factibles se particiona en subconjuntos más simples (Esto es lo que se debería hacer en la práctica, si uno está buscando por ejemplo una aguja en un pajar. El pajar es grande y es imposible buscar en todo simultáneamente, así que se puede dividir visualmente en lado derecho e izquierdo y seleccionar uno de ellos para buscar la aguja, manteniendo el otro lado en espera para buscar después si es necesario). A continuación un subconjunto prometedor se selecciona y se hace un esfuerzo para encontrar la mejor solución factible y se almacena esta información, en algunos casos podría darse por terminado el método. Se particiona nuevamente el subconjunto en dos o más subconjuntos más simples (bajo la operación denominada ramificación) y se repite el mismo proceso

5.1 DESCRIPCION CONCEPTUAL DE RAMIFICACION Y ACOTAMIENTO

El método de Ramificación y Acotamiento (R-A) , surgido hace dos décadas, ha resultado ser una de las mejores herramientas prácticas para la solución de problemas de optimización discreta. Su atractivo radica en la habilidad de eliminar implícitamente grupos grandes de soluciones potenciales sin evaluarlos explícitamente. A semejanza de la programación dinámica, la técnica de ramificación y acotamiento es una estrategia, y como tal se debe combinar con la estructura del problema específico que se desea resolver, para así formar un algoritmo de solución adecuado.
Hay muchos problemas de optimización discreta para los cuales los métodos "directos" o no existen o son ineficaces. Los problemas pueden ser tales que las restricciones o función(es) objetivo(s) son no convexas, o que todos o algunos de los valores están restringidos a valores discretos. La técnica de ramificación y acotamiento nos posibilita resolver problemas "difíciles" usando los métodos existentes para la solución de problemas "fáciles".
Suponga que se desea resolver un problema "difícil" de optimización discreta, como el siguiente:

$$\text{Minimizar } C^0(x)$$
$$\text{Sujeto a}$$

$$g_1^0(x) \geq 0,$$
$$g_2^0(x) \geq 0,$$
$$\cdots$$
$$g_m^0(x) \geq 0$$

$$x \in X^0$$

donde el conjunto X^0 denota el dominio permisible de optimización por ejemplo el octante positivo del espacio n-euclideano, x denota un vector $(x_1, x_2, \ldots, x_n)$ y $g_i^0(x)$ son las funciones que restringen el problema. Un vector x que satisfaga las restricciones y se encuentre en el dominio de optimización se dice que es una solución factible, y uno que además minimiza la función objetivo es una solución óptima.

Existen cuatro razones importantes para aceptar ampliamente los algoritmos de ramificación y acotamiento en la optimización discreta:

a) El método es conceptualmente simple y fácil de entender
b) Es fácilmente adaptable a un amplio rango de situaciones problemáticas
c) es ajustable para su implantación computacional y
d) Los métodos alternativos usualmente no están disponibles.

La estrategia de aplicación de método de R-A se basa en la premisa de que el problema que se va a resolver tenga los siguientes atributos:

a) Naturaleza Combinatoria. Un problema combinatorio tiene como mínimo las siguientes propiedades:

a.1) Se da un conjunto finito de objetos
a.2) Cada objeto puede tomar un cierto rango de atributos.
a.3) Se desarrolla una solución al problema fijando los valores de atributos para todos los objetos
a.4) Solo se permiten ciertas combinaciones de los valores de atributos.

b) Ramificabilidad. Es una propiedad del problema que implica:

b.1) Construir un conjunto finito y contable que contenga todas las soluciones del problema (esto se sigue de 1);
b.2) Particionar recursivamente un conjunto no vacío de soluciones en subconjuntos disjuntos.

c) Racionalidad. Un problema racional es tal que:

c.1) Cada solución tiene un único valor calculado de los valores de sus atributos.
c.2) La "mejor" solución es aquella con mayor (o menor) valor.

d) Acotabilidad. Una estimación del valor de la mejor solución contenida en cualquier conjunto de soluciones se puede obtener tal que:

d.1) El valor actual de la mejor solución en el conjunto es inferior o igual a la estimación (así la estimación es una cota superior);
d.2) Se hace un mínimo esfuerzo para obtener dicha estimación;
d.3) La estimación es razonablemente cercana al valor actual.

El concepto de R-A explota éstas propiedades para poder implícita y explícitamente construir un árbol que describa todas las operaciones realizadas en el problema, y efectuar una búsqueda para encontrar la mejor solución.

5.2 EL ARBOL DE BUSQUEDA.

El proceso de ramificación del problema de optimización discreta puede representarse por medio de un árbol de búsqueda. Cada nodo del árbol está asociado a un subconjunto de T, conjunto de todas las soluciones originales del problema. Sea T_i un subconjunto de T, esto es, una colección de soluciones del problema. La Ramificación es el proceso de particionar un subconjunto T_i en m subconjuntos disjuntos $v_1, v_2, \ldots, v_m$ donde

$$v_1 \cup v2 \cup \ldots \cup v_m = T_i,$$
$$v_i \cap v_j = \phi, \quad i \neq j.$$

O bien, la unión de ellos es T_i y la intersección entre ellos es vacía.

El proceso de ramificación se puede visualizar como la creación o desarrollo ordenado de un árbol donde el nodo inicial representa a T, y los nodos restantes representan subconjuntos T_i de T. A cada nodo N se le asocia un subconjunto T_i.

En la figura 5.1 se ilustra un árbol que exhibe estos conceptos para el caso m=2. El nodo N=1 corresponde a T, el conjunto de todas las soluciones. Los nodos 2 y 3 están asociados con los conjuntos ajenos T_1 y T_2 de T. Note que no hay ramificaciones desde los nodos 3, 4 y 5.

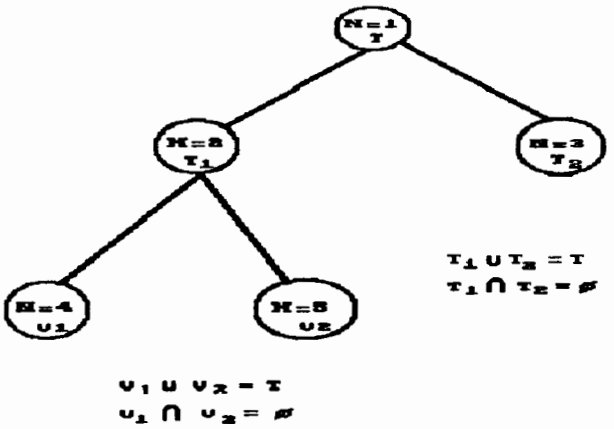


figura 5.1

La relación, entre subconjuntos de T y la ramificación para la creación del árbol, se formaliza como sigue:

$T_i(N)$ es el subconjunto T_i de T representado por el nodo N.

Un Nodo Intermedio es un nodo N en el cual no ha habido ramificación.

Un Nodo Final es un nodo intermedio N para el cual Ti(N) consiste de una solución única .

L(N) es una cota inferior de la función objetivo Z(x) en el conjunto de las soluciones asociadas con el nodo N, esto es, $L(N) \leq Z(x)$ para toda $x \in Ti(N)$.

Una forma conceptual y al mismo tiempo general de describir el método de Ramificación y Acotamiento es como sigue.

EL ALGORITMO BASICO DE RAMIFICACION Y ACOTAMIENTO

PROPOSITO: Encontrar la solución óptima de un problema de optimización discreta

DESCRIPCION

Paso 0. Se inicia con el conjunto de todas las soluciones factibles del problema en cuestión. Se forma el primer nodo del árbol

Paso 1. Se procede a ramificar los nodos hacia nodos nuevos, utilizando alguna regla de ramificación

Paso 2. Se determinan cotas inferiores para los nuevos nodos. Para cada nuevo nodo N se obtiene una cota inferior L(N) sobre el valor de la función objetivo para las soluciones factibles del nodo.

Paso 3. Se selecciona un nodo intermedio desde el cual se ramifica. Cada nuevo nodo se excluye o se sondea si: Se encuentra que el nodo no contiene soluciones factibles, o se ha identificado la mejor solución factible en el nodo (de tal forma que L(N) corresponde al valor de su función objetivo);

Paso 4. Reconocer cuando un nodo final contiene una solución óptima. El proceso termina cuando no existen nodos restantes (no sondeados) y la solución de apoyo actual es la solución óptima,(si no existe tal solución de apoyo entonces el problema no tiene soluciones factibles). De otra manera se regresa al paso 1.

Si el objetivo es maximizar solo se invierten los papeles de las cotas y se invierten las direcciones de las desigualdades.

CONVERGENCIA DEL ALGORITMO.

La operación de ramificación del algoritmo garantiza una solución óptima en un número finito de iteraciones. Como T es finito el proceso de ramificación conducirá eventualmente a una solución T_i, como nodo final, a menos que se detenga previamente. El proceso de ramificación produce una partición de T para la cual cada subconjunto Ti, obedece, a la definición de partición. Así todos los nodos finales posibles se pueden generar y obtenerse así la solución óptima. Las características de la ramificación prometen una enumeración completa para el algoritmo de R-A a menos que se pueda encontrar una solución óptima antes de hacerlo. Las características de acotamiento del algoritmo de R-A proporcionan la posibilidad de reconocer una solución óptima antes de completar la enumeración.

5.3 ESTRATEGIAS OPERATIVAS EN EL ALGORITMO BASICO.

Empezaremos primero con las distintas opciones para efectuar la operación de ramificación. Identificamos tres formas distintas de ramificar: ordenada, inmediata del sucesor y ramificación del sucesor. En aras de hacer más sencilla la exposición planteamos un problema de permutación y a través de él se exponen las distintas técnicas de ramificación.

Se desea ordenar un conjunto de n artículos $A=\{a_1,\ldots,a_n\}$, esto es, buscamos un ordenamiento que sea óptimo de los n artículos bajo condiciones dadas. El conjunto de soluciones factibles consiste de vectores n-dimensionales.

$$F = \{x \mid x_i \in A \text{ y } x_i \neq x_j \text{ si } i \neq j\}$$

donde el vector x, representa un arreglo de los n artículos. La exigencia de que x_i sea diferente a x_j si i es distinto de j, se debe a que un mismo artículo a_i, no puede ocupar dos lugares en un mismo arreglo.

A continuación se discuten las posibles reglas de ramificación para el caso en que n=4 y A={a,b,c,d}

RAMIFICACION ORDENADA

Vamos a llamar X al conjunto de soluciones factibles, que contiene los artículos ya seleccionados, llamaremos R(X) al rango de X, es el artículo que vamos a seleccionar para introducir a la solución factible que se construye, llamaremos nivel de X, al número de artículos seleccionados, así sea $R=\{r_1,\ldots,r_p\}$ un subconjunto de los índices de los artículos {1,2,...,n} que contiene p elementos distintos, que se han introducido al arreglo.

Un nodo X de un árbol de búsqueda se determina al imponer las siguientes restricciones.

$$x_{r1}=a_{j1}, \quad x_{r2}=a_{j2}, \ldots, x_{rp}=a_{jp}$$

para algún conjunto $\{a_{j1},\ldots,a_{jp}\}$ de p artículos distintos (los seleccionados), de donde, la siguiente selección de X la haremos sobre los restantes n-p artículos a ordenar, lo que nos genera una partición de X en n-p subconjuntos ajenos, al especificar el artículo que tenga orden r(X) para alguna r(X) que no está en R, seleccionamos también el nuevo conjunto de soluciones factibles a particionar.

La regla de ramificación ordenada, nos especifica las reglas explícitas de selección, para asignar un rango a X: r(X), de esta forma r(X) se considera sólo en función del nivel de X (donde p = nivel de X). Por ejemplo: r(X)=p+1 consiste en seleccionar primero,

el artículo que irá en primer lugar del arreglo, después el del segundo lugar y así sucesivamente, de ésta forma, si se tiene un conjunto $X=\{x_1, x_2, x_3, x_4\}$ donde la regla de selección es "mayor que" se tiene $x_1\ x_2\ x_3\ x_4$. Si la regla fuera $r(X) = n-p$ entonces empezaríamos por seleccionar el último del arreglo, y terminar en el primero, en el ejemplo anterior la solución es la misma pero la selección se hace considerando primero a x_4, después a x_3 y así hasta x_1.

La figura 5.2 muestra un árbol de enumeración completa, para n=4 y A={a, b, c, d} generado por ramificación ordenada.

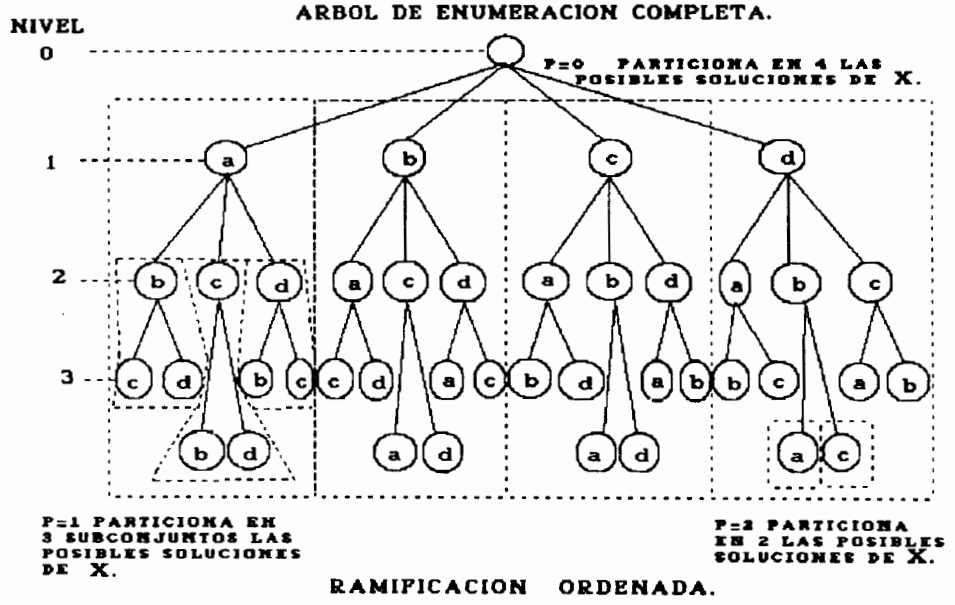


figura 5.2

Otra regla de selección consiste en ir insertando en forma ordenada cada uno de los artículos, de la manera siguiente, al primer artículo le asignamos su lugar en el arreglo solución. Por ejemplo considere los nodos a, b, c y d en los niveles 0, 1 2 y 3 respectivamente. Entonces Z corresponde a la solución

(b,d,a,c) si r(a)=3, r(b)=1 r(c)=4, r(d)=2
(b,a,d,c) si r(a)=2, r(b)=1 r(c)=4, r(d)=3
(a,b,c,d) si r(X)= (Nivel de X)+1 para toda X
(d,c,b,a) si r(X)= n-(Nivel de X) para toda X

RAMIFICACION INMEDIATA DEL SUCESOR

Sea X un nodo del árbol especificado por p(= nivel de X) relaciones de la forma $r_j = r_i + 1$ que puede interpretarse como el renglón a_j ordenado inmediatamente después del renglón a_i). Entonces X se puede particionar en dos conjuntos:

$$X(kl) = \{ x \mid x \in X \text{ y } r_l = r_k + 1 \}$$

$$X(kl^*) = X - X(kl) = \{x \mid x \in X \text{ y } r_l \neq r_k + 1\}$$

Se requiere una regla de selección para decidir el par de índices k y l. La figura 5.3 da un ejemplo de un árbol de enumeración completa para n=4 y A={a, b ,c, d}

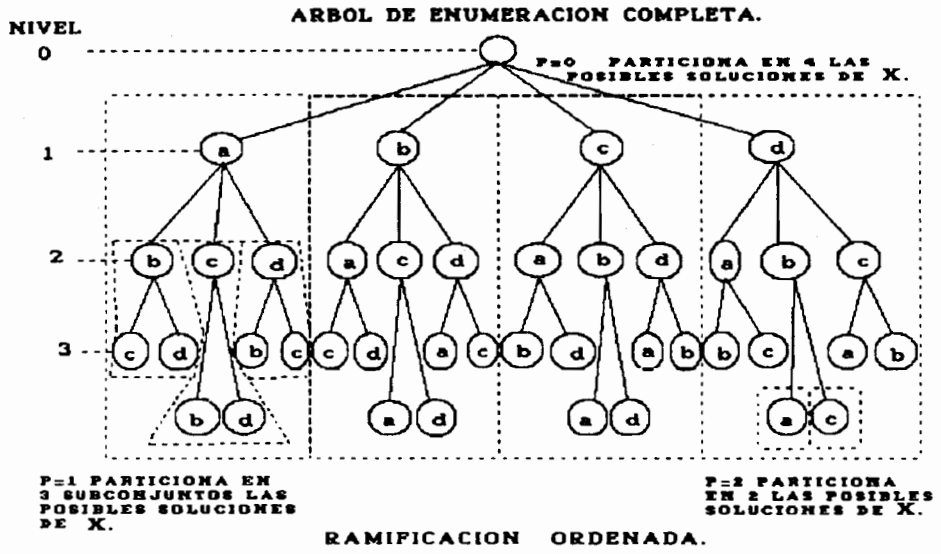


figura 5.3

RAMIFICACION DEL SUCESOR

Sea X un nodo del árbol especificado por p relaciones $r_j > r_i$ (que se interpretan como un renglón a_j ordenado posteriormente a un renglón a_i). Donde X puede particionarse en dos conjuntos

$$X(kl) = \{x \mid x \epsilon X \ y \ \ x_l > x_k\}$$

$$X(lk) = X - X(kl) = \{x \mid x \epsilon X \ \ y \ \ x_k > x_l\}$$

Nuevamente se requiere de una regla de selección para escoger la pareja (k,l), $k \neq l$. La figura 5.4 da un ejemplo de un árbol de enumeración completa.

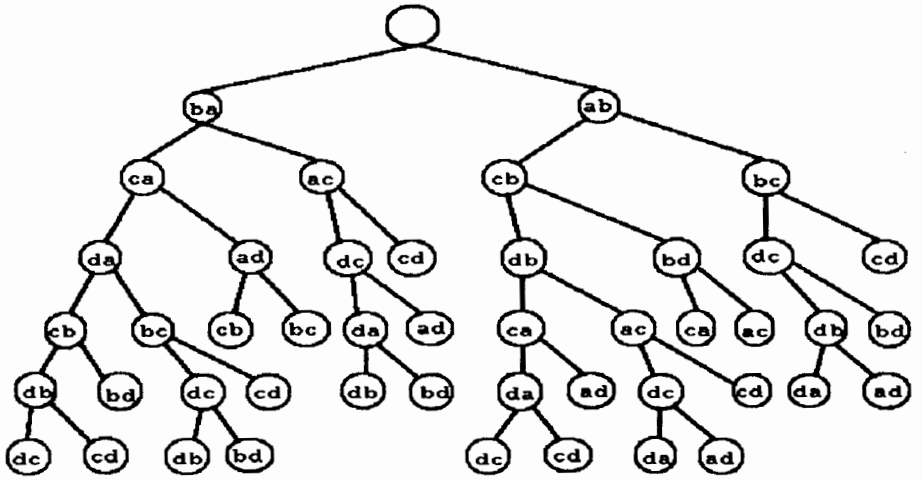


figura 5.4

ACOTAMIENTO

Un árbol de búsqueda puede formarse con una pequeña parte de un árbol de enumeración completa (en el cual no se descarta ningún nodo, se consideran todos los posibles), debido al tamaño tan grande de éste. Para poder hacer esto escogemos una regla de acotamiento adecuada. Una función de acotamiento es una función $g:2^T \rightarrow R$ que asigna un número real g(x) a cada subconjunto T tal que si X está contenido en T entonces la cota g(X) satisface

$$g(X) \leq f(x) \text{ para toda } x \epsilon X \cap F$$

(Note que las cotas son todas cotas inferiores ya que estamos suponiendo un problema de minimización). Podemos suponer también que

$$g(\{x\})=f(x) \text{ para toda } x \in F \qquad (5.1)$$

Cotas para un problema dado P pueden obtenerse relajando alguna de las restricciones. Esto da el problema relajado Q cuyo conjunto factible F_Q contiene al conjunto factible F_P del problema original P. Sea X un subconjunto de T entonces

$$g(X) = \min_{y \in X \cap F_Q} f(y) \leq \min_{y \in X \cap F_P} f(y) \leq f(x) \quad \text{para toda } x \in X \cap F$$

claramente g es una función de acotamiento inferior.

Una vez que hemos establecido las reglas de ramificación y acotamiento, describiremos a continuación las principales estrategias de desarrollo del algoritmo de R-A, las más importantes son:

BUSQUEDA A PRIMER PROFUNDIDAD O REGLA DE LA COTA MAS RECIENTE O TAMBIEN LIFO

Es la estrategia de ramificación del nodo activo más reciente (un nodo Y es activo dependiendo de que f(S) > g(Y) donde S es la solución actual, y Y es algún nodo del conjunto total de soluciones), esto es, se escoge el nodo de máxima profundidad entre aquellos nodos que aún no se ramifican. Si hay más de uno, entonces se selecciona aquél que corresponda al valor de cota inferior. Esta regla logra que el árbol llegue a un nodo terminal rápidamente, aunque se podrían requerir más cálculos computacionales, se requiere un mínimo de memoria, ya que es suficiente para acumular problemas correspondientes a cada nodo en una ruta de la raíz del árbol al nodo terminal.

En la terminología de Ramificación y Acotamiento esta estrategia de desarrollo se conoce como búsqueda en la lista de nodos activos, bajo el criterio que el último en entrar es el primero en salir.

PRIMER ALCANCE O REGLA DE LA MENOR COTA O TAMBIEN FIFO

Esta regla también se conoce como método o regla de separación progresiva y evaluación. Aquí se escoge sistemáticamente el nodo con el valor de la menor cota observando que es en éste en el que se tiene intuitivamente mayor oportunidad de que contenga una solución óptima entre sus sucesores. Sin embargo, con este método corremos el riesgo de tener que explorar una gran porción del árbol antes de encontrar una solución. Nuevamente, la primer solución que se encuentra es en general mejor (esto es, de menor costo) que la

que se encuentra por la estrategia de búsqueda de "primer profundidad".

Esta regla se usa también en la programación dinámica y es la opuesta a la de primera profundidad en la estrategia de ramificar un "mínimo nodo activo recientemente creado". También se conoce como FIFO porque si se tiene una lista de nodos activos es el primero en entrar y el primero en salir.

COSTO UNIFORME

Esta regla consiste en escoger un nodo X del árbol parcialmente expandido, para el cual:

$$g(X) \leq g(Y) \quad \text{para todo nodo Y}$$

los empates se resuelven por alguna regla subsidiaria. Esta es la ramificación de la menor cota o estrategia de costo uniforme (UC) y es computacionalmente óptima en el sentido de que efectivamente requiere un mínimo número de nodos a ser desarrollados. Aunque produce árboles pequeños, puede caer en dejar un mínimo tiempo de cómputo y también sufre el defecto de producir primero una solución factible en o cerca del fin de los cálculos.

Tres estrategias se describen en la figura 5.5 para el problema de búsqueda para encontrar la trayectoria mas corta de L a R. de la gráfica a). En b) se usa primer profundidad, en c) primer amplitud y en d) costo uniforme.

DOMINACION

En algunas aplicaciones de ramificación y acotamiento puede suceder que hay caminos fuera de la estructura de R-A que permiten establecer que para algún par de conjuntos X,Y contenidos en T la mejor solución en X es al menos tan buena como la mejor solución en Y. Entonces se dice que X domina a Y , y el nodo de Y en el árbol no necesita desarrollarse (Si X domina a Y y Y domina a X entonces uno de ellos puede estar inactivo).

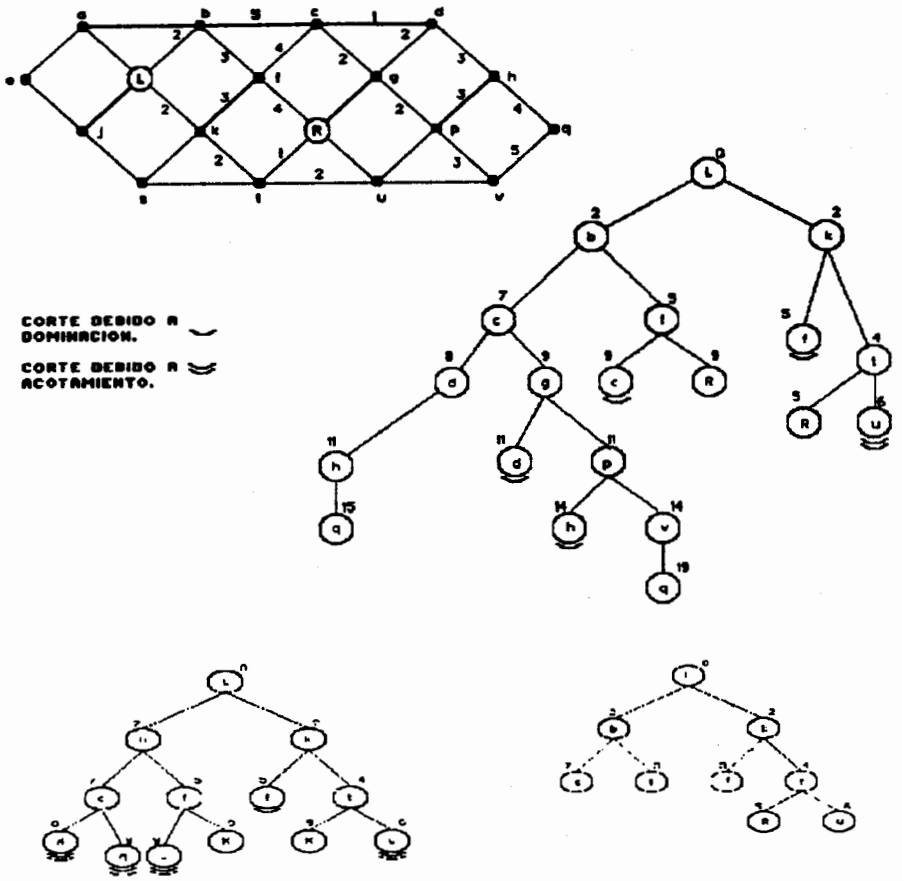


figura 5.5

Finalmente, puede notarse que las técnicas de ramificación y acotamiento son exactas en el sentido de que garantizan soluciones óptimas. Sin embargo, involucran una cantidad considerable de cálculos y si una buena solución subóptima es aceptable entonces a menudo es mejor usar ramificación y acotamiento "heurísticamente" en donde la mejor solución factible generada, en un tiempo dado, es la que se considera como la mejor.

5.5 EJEMPLOS ILUSTRATIVOS.

En los ejemplos siguientes se exhibe la forma en que se pueden emplear las reglas de ramificación y acotación. El último ejemplo es el famoso caso de las cuatro reinas en donde ilustramos las reglas de desarrollo a primer profundidad y de primer alcance

Ejemplo 5.1 (Ramificación)

En un experimento de comparación apareada sobre los elementos de un conjunto {a,b,c,d} las preferencias se resumen en la siguiente matriz, donde 1 indica que existe discrepancia y 0 que no la hay. Si aceptamos que las preferencias queden como a > b > c > d el costo en que se incurre es de 2 calculado como sigue: en el renglón b columna a se lee, si a > b no se tiene discrepancia y no se paga nada, b > c se lee en el renglón c columna b y se tiene una discrepancia y se paga una unidad, c > d no se tiene discrepancia a > c no se paga nada, a > d se paga una unidad y finalmente b > d no se paga nada. Se requiere encontrar un ordenamiento que minimice el número de discrepancias.

	a	b	c	d	puntuación total
a	-	1	1	0	2
b	0	-	0	1	1
c	0	1	-	1	2
d	1	0	0	-	1

Empezaremos con el conjunto {a, b, c, d} donde la primera decisión que se presentará es: ¿Podría a ordenarse por encima de b, o no ? y aplicamos la regla de ramificación del sucesor.

En la figura 5.6 se muestran tres etapas del desarrollo del árbol de búsqueda en ésta solución. Las cotas inferiores se muestran a un lado de cada nodo. De los dos nodos aumentados al árbol de búsqueda, el primero con a > b (esto es a ordenada por encima de b) es el mas prometedor, donde a > b no implica -inmediatamente- discrepancias. De la matriz de preferencias se observa que c, tambien es notablemente mejor que b, y la segunda ramificacion corresponde de un modo u otro a c > b.

Después de tres ramificaciones el árbol de búsqueda desarrollado se muestra en la figura 5.6c). Ya que el ordenamiento a > c > b > d involucra solo una discrepancia y cualquier solución

correspondiente al subconjunto de soluciones en cualquier otro nodo implica al menos una discrepancia, a > c > b > d debe de ser un ordenamiento óptimo. En ésta etapa todavía no se demuestra que éstasolución óptima sea única. Para establecer esto, o encontrar otras soluciones óptimas, es necesario desarrollar todos los nodos que consisten de sólo una discrepancia.

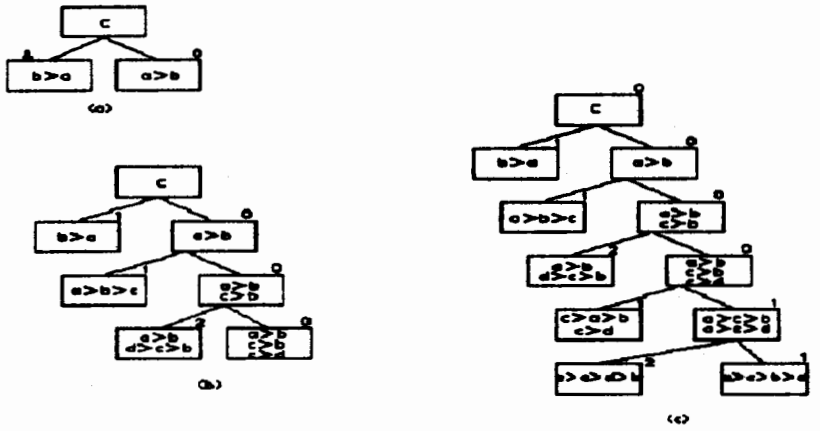


figura 5.6

Este método proporciona fácilmente una solución para este ejemplo, en cierto modo, debido a la manera juiciosa de escoger la ramificación.

Otra forma de resolver éste problema consiste en usar la regla de ramificación ordenada, con $r(X)=p+1$. Sea entonces $x_1 > x_2 > x_3 > x_4$ un ordenamiento, donde x_i con $i=1,\ldots,4$ puede ser un elemento cualquiera del conjunto {a,b,c,d}. Hay $4^4 = 256$ cuartetas distintas (x_1, x_2, x_3, x_4) de las que solo $4! = 24$ corresponden a ordenamientos válidos. Esto es, de los 256 candidatos a solución, 232 no son factibles y deben de ser eliminados.

La regla de ramificación será adoptada de la siguiente manera: La primer ramificación corresponde a la selección de x_1, el elemento del ordenamiento mas alto, la segunda y tercera ramificaciones corresponden a la selección de x_2 y x_3 respectivamente (si se especifican x_1, x_2, y x_3 entonces x_4 queda automáticamente determinada). El árbol de búsqueda se desarrolla en forma similar que en el primer método, como se muestra en la figura 5.7. Como vimos anteriormente a > c > b > d se encuentra que es óptima.

En este caso, ambos métodos de solución nos conducen a un ordenamiento óptimo. En general no es éste el caso y cierto retroceso será necesario. La solución que se obtiene con ambos métodos es a > c > b > d con una discrepancia.

Ejemplo 5.2 Problema de Asignación Lineal (Acotamiento)

Suponga que se desean usar cuatro barcos cargueros para

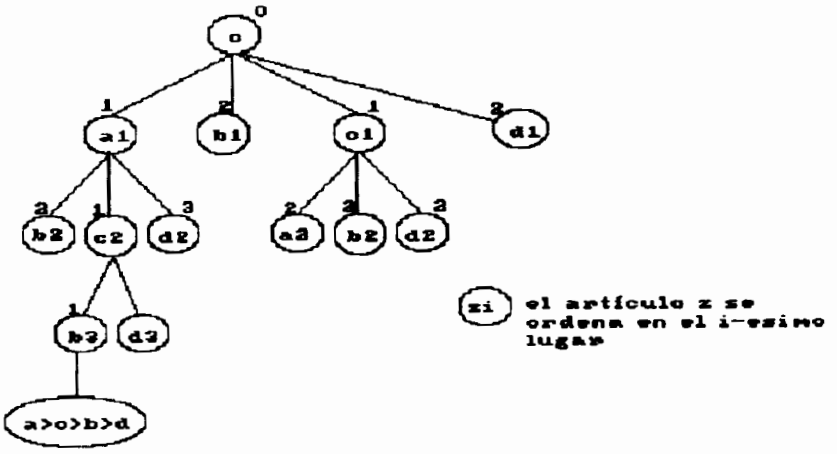


figura 5.7

transportar bienes de un puerto base a otros cuatro puertos (numerados 1, 2, 3 y 4). Se puede usar cualquier barco para hacer cualquiera de los cuatro viajes. Sin embargo, dadas las diferencias entre barcos y las cargas a llevar, el costo total de carga, transporte y descarga de bienes para distintas combinaciones de barcos y puertos varía. Los costos se muestran en la siguiente tabla donde el componente (1,1) de dicha matriz es 15, que es el costo de transportar carga con el barco 1 al puerto 1 (desde el puerto base).

		PUERTOS				
BARCOS		1	2	3	4	mínimo por renglón
	1	15	5	9	5	5
	2	17	14	4	7	4
	3	9	13	13	7	7
		13	10	6	15	6

Se desea asignar los barcos a los puertos, en una correspondencia uno a uno de manera que se minimice el costo total. Si definimos

$$T = \{x \mid x = (i,j,k,l) \text{ e } i,j,k,l \in \{1, 2, 3, 4\}\}$$

como el conjunto de soluciones posibles, el problema de asignación (lineal) P consiste en

$$\text{minimizar } z = c_{1i} + c_{2j} + c_{3k} + c_{4l}$$

donde c_{ij} son los costos de transportar carga con el barco i al puerto j y donde (i,j,k,l) son las permutaciones de (1,2,3,4)

Ejemplos de algunas cotas de costo son :

$$g(T) = \min_{1\le\alpha\le4} c_{1\alpha} + \min_{1\le\beta\le4} c_{2\beta} + \min_{1\le\Gamma\le4} c_{3\Gamma} + \min_{1\le\delta\le4} c_{4\delta} = 5+4+7+6=22$$

$$g(T[i=3]) = c_{13} + \min_{1\le\beta\le4} c_{2\beta} + \min_{1\le\Gamma\le4} c_{3\Gamma} + \min_{1\le\delta\le4} c_{4\delta} = 9+4+7+6=26$$

$$g(T[i=3],[j=2]) = c_{13} + c_{22} + \min_{1\le\Gamma\le4} c_{3\Gamma} + \min_{1\le\delta\le4} c_{4\delta} = 9+14+7+6=36$$

donde T[i=3] denota el subconjunto de T con el índice i fijo en 3. Se puede ver que si i es el conjunto evaluado en 3 podemos insistir que los índices usados en la minimización sean β, Γ y δ = 3. Esto permite perfeccionar la función de acotamiento g'

$$g'(T) = g(T) = 22$$

$$g'(T[i=3]) = c_{13} + \min_{\beta=1,2,4} c_{2\beta} + \min_{\Gamma=1,2,4} c_{3\Gamma} + \min_{\delta=1,2,4} c_{4\delta} =$$

$$= 9 + \min\{17,14,7\} + \min\{9,13,7\} + \min\{13,10,15\}$$
$$= 9 + 7 + 7 + 10 = 33$$

$$g'(T[i=3],[j=2]) = c_{13} + c_{22} + \min_{\Gamma=1,2,4} c_{3\Gamma} + \min_{\delta=1,2,4} c_{4\delta} =$$

$$= 9 + 14 + \min\{9,7\} + \min\{13,15\} = 43$$

Observe que g' satisface la ecuación (5.1) sin modificación y nótese también que $g'(X) \ge g(X)$ para toda X en T; se dirá entonces que g'es mas fuerte que g.

El siguiente problema es un ejemplo clásico del uso de los métodos de tanteo y de los algoritmos de rastreo inverso. Se dice que fue investigado por Gauss en 1850, pero no lo resolvió completamente.

Ejemplo 5.3 (El Problema de las Cuatro Reinas).

-¿Sabes sumar?-le preguntó la reina blanca- ¿Cuánto es uno y uno y uno y uno y uno y uno y uno y uno?
-No sé -dijo Alicia- he perdido la cuenta...
-¡No tiene ni idea de matemáticas! -sentenciaron enfáticamente ambas reinas a la vez.
Lewis Carroll. Alicia a través del Espejo.

Cuatro reinas se deben colocar en un tablero de cuatro por cuatro, de forma tal que ninguna reina pueda atacar a otra. En otras palabras, la colocación debe ser tal que en cada renglón, columna o diagonal del tablero, a lo mas contenga a una reina. El problema puede considerarse como una sucesión de problemas, en donde por ejemplo se coloca una reina en el primer cuadro del renglón de arriba, entonces se coloca otra reina en el segundo renglón de forma tal que no pueda ser atacada por la primera, y de igual forma se procede con la tercera y la cuarta reina.

Podemos asociar posiciones con nodos de un árbol, donde el nodo raíz T corresponde a la posición inicial sin reinas, los siguientes nodos corresponden a las posiciones posibles de una reina en el primer renglón (Es suficiente considerar como posiciones iniciales los primeros dos cuadros, ya que los otros dos dejan posiciones simétricas), conectados con el nodo T, por arcos de longitud cero, las siguientes ramas del árbol se construyen a partir de éstos nodos, considerando las posibles posiciones de una reina en el segundo renglón, sin que ésta ataque a la reina que se ha colocado con anterioridad, procediendo a unir éstos nodos con arcos de longitud cero, así se continúa hasta colocar a las cuatro reinas en sus posiciones respectivas.

Resolveremos primero éste problema usando la estrategia de búsqueda de primer profundidad. Numeramos los renglones y las columnas del tablero del 1 al 4, las reinas también pueden numerarse del 1 al 4. Ya que cada reina debe estar en diferente renglón, podemos suponer sin pérdida de generalidad que la reina i debe encontrarse en el renglón i. Todas las soluciones al problema deben presentarse entonces como cuartetas (x_1, x_2, x_3, x_4) donde x_i es la columna donde se pone la reina i. Las restricciones explícitas usando ésta formulación son $S_i = \{1,2,3,4\}$, $1 \leq i \leq n$. Entonces el espacio de soluciones consiste de 4^4 cuartetas. Las restricciones implícitas para éste problema es que no puede haber dos xi's que sean iguales(esto es, todas las reinas deben estar en columnas distintas) y dos reinas no pueden estar en la misma diagonal. La primera de éstas dos restricciones implica que todas las soluciones son permutaciones de las cuartetas (1,2,3,4). Esta realización reduce el tamaño del espacio de soluciones de 4^4 a 4!.

Mas adelante veremos como formular la segunda restricción en términos de las x_i. Expresada como una solución la cuarteta de la figura 5.8 se tiene (2,4,1,3).

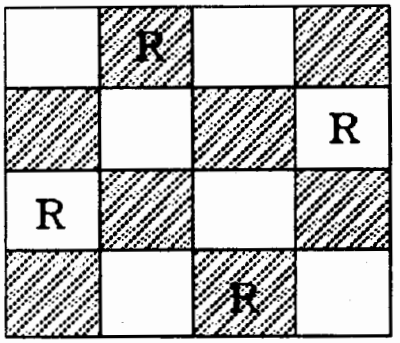


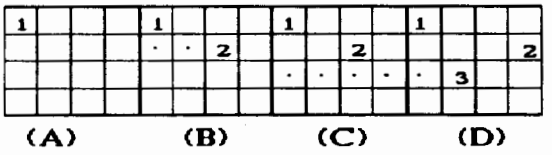
figura 5.8

Como una función de acotamiento usamos el criterio obvio de que si $(x_1, x_2, \ldots, x_i)$ es la trayectoria del T-nodo actual entonces todos los nodos hijos con etiquetas padre-hijo x_{i+1} tales que $(x_1, \ldots, x_{i+1})$ representa una configuración en donde no se pueden atacar dos reinas.

Comenzamos con el nodo raíz como el único nodo vivo, éste es el T-nodo y la trayectoria es (0). Se genera un hijo. Suponemos que generamos hijos en orden ascendente, así el nodo numerado 2 de la figura 5.11 se genera y la trayectoria es ahora (1). Esto corresponde a poner la reina 1 en la columna 1. El nodo 2 se convierte en el T-nodo. El nodo 3 se genera e inmediatamente se muere. El siguiente nodo generado es el nodo 8 y la trayectoria se convierte en (1,3). El nodo 8 se convierte en el T-nodo, sin embargo se muere ya que sus hijos representan configuraciones que no producen una solución. Regresamos al nodo 2 y generamos otro hijo, el nodo 13. La trayectoria es ahora (1,4). La figura 5.9 muestra las configuraciones en el tablero como procedimientos recursivos, y muestra gráficamente los pasos que sigue el algoritmo al tratar de encontrar la solución, los puntos indican los lugares de una reina que se probaron y rechazaron porque otra reina estaba atacando.

En B) la segunda reina se pone en las columnas 1, 2 y finalmente se fija en la columna 3. En C) el algoritmo trata las cuatro columnas y no es posible colocar en ninguna a la próxima reina. Regresando se tiene en D) que la segunda reina se mueve a la próxima columna posible, la columna 4 y se pone la tercer reina en la columna 2. Estos tableros muestran los pasos que el algoritmo sigue hasta encontrar la solución.

La figura 5.10 muestra la parte del árbol que se ha generado usando la estrategia de primera profundidad. Los nodos que se han muerto como resultado de las funciones de acotamiento tienen una M



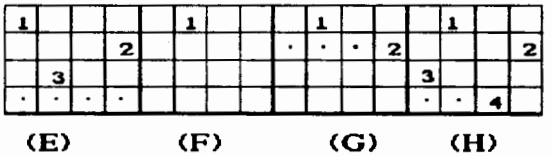


figura 5.9

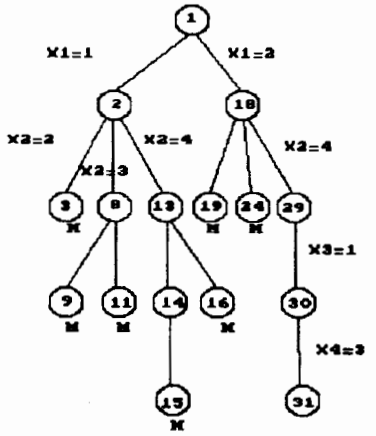


figura 5.10

debajo de ellos.

El árbol completo, usando búsqueda de primer profundidad y sin considerar las funciones de acotamiento se muestra en la figura 5.11

Otra forma de resolverlo es usando la estrategia de primer amplitud o FIFO. Como ya se decía inicialmente hay un solo nodo vivo, el T-nodo y ahora le asignamos el número 1, (este es el caso en que no se tienen reinas en el tablero), Este nodo se particiona

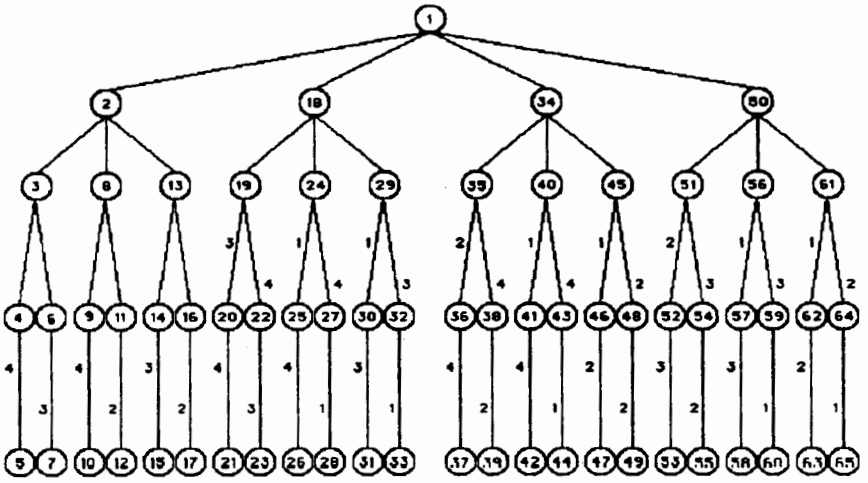


figura 5.11

y sus hijos los nodos 2, 18, 34 y 50 se generan. éstos nodos representan el tablero con una reina en el renglón 1 y columnas 1,2,3 y 4 respectivamente. Ahora los nodos vivos son los nodos 2,18,34 y 50. Si los nodos se generaron en éste orden el próximo T-nodo sería el nodo 2, éste se ramifica y los nodos 3, 8 y 13 se generan. El nodo 3 se mata inmediatamente usando la función de acotamiento. Los nodos 8 y 13 se aumentan a la cola de nodos vivos. El nodo 18 se convierte en el nuevo T-nodo. Los nodos 19, 24 y 29 se generan. Los nodos 19 y 24 se mueren como resultado de la función de acotamiento, que es la misma que se usó en la solución por primer profundidad. El nodo 29 se aumenta a la cola de nodos vivos. El próximo T-nodo es el nodo 34. La figura 5.12 muestra el árbol generado por la estrategia FIFO (primero en entrar primero en salir).

Los números dentro del nodo son los mismos se la estrategia de primer profundidad y los números fuera del nodo corresponden al orden en que fueron generados por FIFO, en el momento en que se alcanza el nodo solución 31, los únicos nodos que permanecen vivos son los nodos 38 y 54. Una comparación de las figuras 5.10 y 5.12 indican que la estrategia de primer profundidad es un mejor método para éste problema.

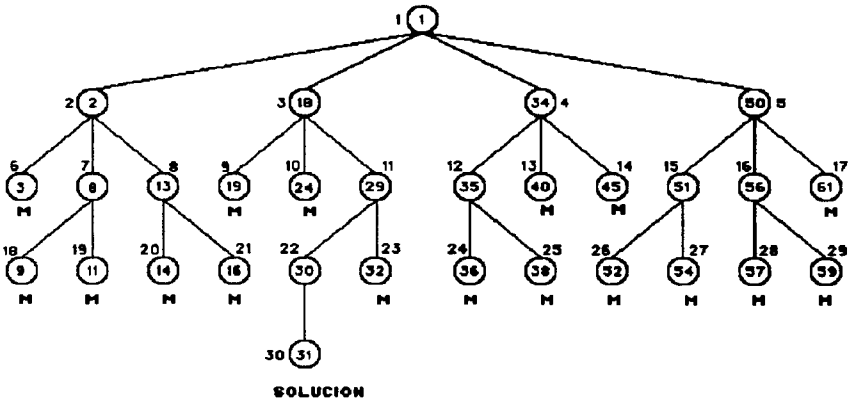
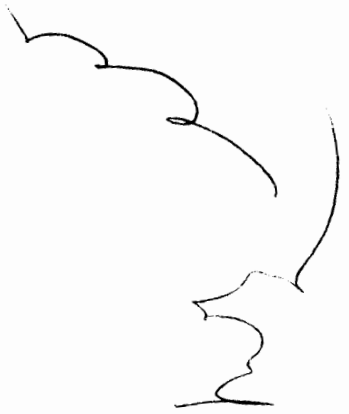

1 1
2 2
3 18
34 4
80 5
6 3
7 8
8 13
9 19
10 24
11 29
12 35
13 40
14 45
15 51
16 56
17 61
18 9
19 11
20 14
21 16
22 30
23 32
24 36
25 38
26 52
27 54
28 57
29 59
30 31
SOLUCION

figura 5.12



CAPITULO 6
METODOS DE RAMIFICACION Y ACOTAMIENTO PARA PROGRAMACION ENTERA MIXTA

"Primero, deben abrazarse de una ojeada todas las ideas particulares desparramadas acá y allá, y reunirlas bajo una sola idea general, para hacer comprender, por una definición exacta, el objeto que se quiere tratar."
" Después se debe saber dividir de nuevo la idea general en sus elementos, como otras tantas articulaciones naturales, guardándose, sin embargo, de mutilar ninguno de éstos elementos primitivos, como acostumbra un mal cocinero cuando trincha".
Fedro, Platón.

El origen de los métodos de solución de la programación entera se remonta al año 1958, cuando Gomory propuso una técnica para generar soluciones enteras usando programación lineal. Como consecuencia de esto se desarrollaron diferentes métodos y se han escrito diversos códigos de computación a nivel comercial. Entre los métodos de solución actuales, están los de ramificación y acotamiento.

El objetivo de éste capítulo es describir el desarrollo de los métodos clásicos de solución de programación entera mixta y la forma en que se han ido depurando. El primer método que se presenta es el que desarrollaron A. H. Land y A.G. Doig, en 1960. Posteriormente R. J. Dakin propone un nuevo algoritmo, basado en el anterior pero con modificaciones que lo hacen mas rápido y mas tarde Norman J. Driebeek y Stanley Zionts presentan un nuevo método que plantea una alternativa distinta a las dos anteriores. Cabe señalar que aunque todos estos algoritmos se refieren a problemas de programación lineal mixta se pueden aplicar a problemas de programación entera pura.

6.1 EL ALGORITMO DE LAND Y DOIG.

Considere el problema:

$$\text{maximizar } z = cx + dy \tag{6.1}$$
$$\text{sujeto a}$$
$$Ax + Dy \leq b \tag{6.2}$$
$$x \text{ vector columna con componentes en } Z^+ \tag{6.3}$$
$$x \geq 0 \tag{6.3'}$$
$$y \geq 0 \tag{6.4}$$

donde z es un escalar; b es un vector columna de m componentes c^t, x son vectores columna de n_I renglones; d^t, y son vectores columna de $(n-n_I)$ componentes; A es una matriz de orden m x n_I; y D es una matriz de orden m x $(n - n_I)$. Una solución factible al problema es aquella que satisface (6.2), (6.3) y (6.4).

A continuación se desarrolla un ejemplo sencillo del Algoritmo de Land y Doig: Se aplica el algoritmo a un problema de programación lineal en dos variables sin restricción de variable discreta.

En la figura 6.1 se puede ver que la función (representada por la familia de rectas paralelas) alcanza el máximo en (x_1^0, x_2^0)

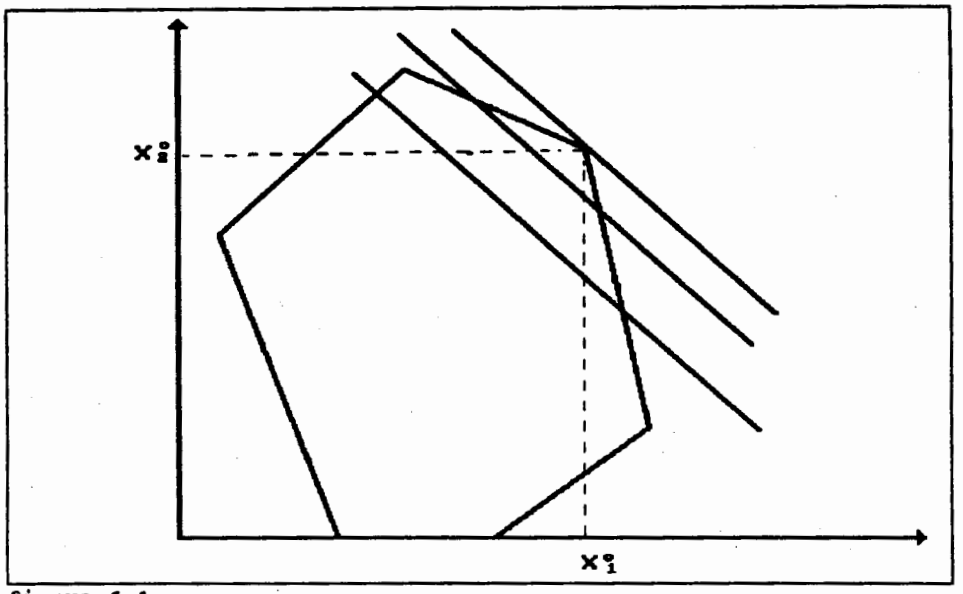


figura 6.1

Las restricciones de variable discreta limitan el conjunto de soluciones factibles a los puntos que están en el conjunto original para los cuales ambas coordenadas son enteras; en la figura 6.2 se muestra el conjunto completo de soluciones factibles para el problema de programación discreta el cual consta de 10 puntos, es fácil ver que en éste caso la solución máxima es $x_1 = 3$, $x_2 = 3$ (indicada por el punto E). El procedimiento para llegar a ella se puede describir como "mover hacia abajo la función lineal hasta encontrar un punto entero".

DESCRIPCION DEL METODO

Este método usa sistemáticamente paralelas en el hiperplano en términos de reducir la función objetivo hasta encontrar un punto en el conjunto de soluciones factibles con coordenadas enteras, en las dimensiones que se especifican. Esto es en principio, ya que en la práctica no se tiene la facultad de "ver" un hiperplano en un espacio de n dimensiones, en términos de determinar si contiene un

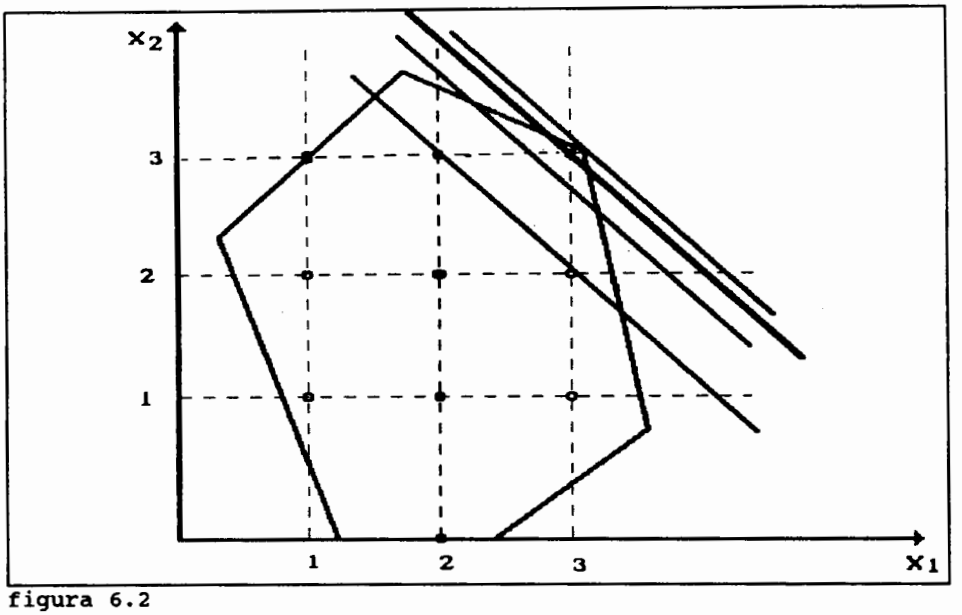


figura 6.2

punto cuyas n coordenadas son enteras. Los métodos numéricos sólo pueden examinar un punto a la vez, pero al pasar el hiperplano de una región a otra no se debe omitir ningún punto entero.

Si la cota superior de la función en cualquier etapa es z^K entonces se ha demostrado que hay una solución no discreta con un valor mas alto del maximando que z^K

Considere el conjunto convexo de soluciones de un problema de programación lineal y z^K cualquier valor factible de la función objetivo, que está únicamente asociado con una posición definida del hiperplano, que en general corta al conjunto convexo de n dimensiones y en el caso especial del valor óptimo toca al conjunto convexo de n-1 dimensiones.

Por ejemplo la figura 6.3 representa una intersección con un conjunto tridimensional.

Los puntos x_1, x_2 y x_3 son las intercepciones del hiperplano con los tres ejes respectivamente y el área sombreada representa el conjunto convexo que está dentro de las restricciones del problema de programación lineal ordinario. En tal caso el conjunto de n-1 dimensiones tendría un mínimo y un máximo para cada variable como

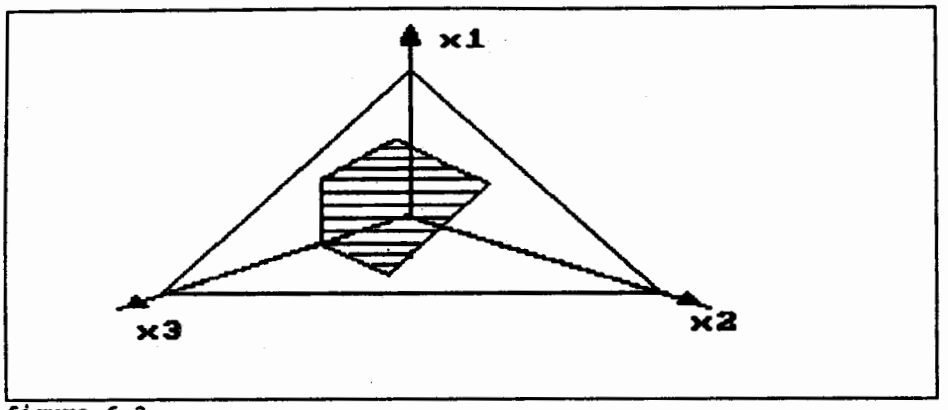


figura 6.3

se muestra en la figura 6.4

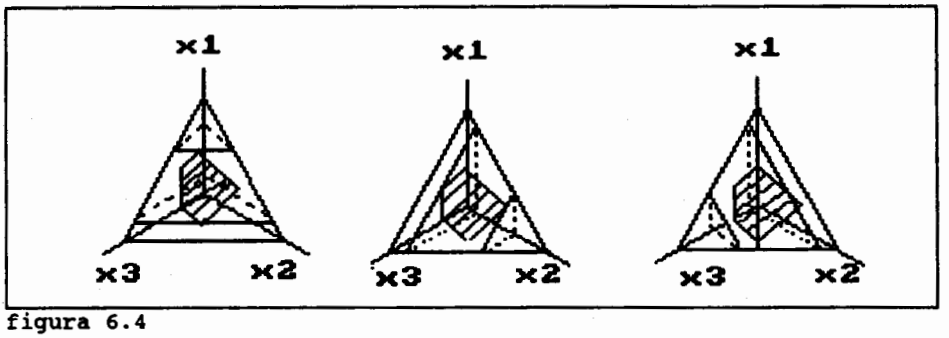


figura 6.4

Si $z^t = z^0$ esto es, el hiperplano es el asociado con la solución óptima del problema de programación lineal ordinario,este conjunto convexo consistiría normalmente de un único punto (a menos que tuviera soluciones óptimas múltiples). Es también cierto que, para cualquier valor del maximando (posición del hiperplano) hay un valor mínimo y uno máximo (que puede coincidir) para cada variable, consistente con las restricciones del problema de programación lineal ordinario.

Conociendo un valor particular de z, los valores mínimo y máximo para cada variable se conocen así como un posible valor entero de cada variable para la posición del hiperplano. Si no hay al menos un posible valor entero para cada variable entonces se puede decir inmediatamente que no hay solución al problema en el

valor del maximando. Desafortunadamente lo inverso no es cierto, el hecho de que cada variable tome un valor entero en algún punto o puntos en el conjunto no es condición suficiente para que haya una intersección de esas coordenadas enteras en el conjunto y por lo tanto una solución factible.

Ya que hay un valor mínimo y un valor máximo para cualquier variable x_k en un valor particular de z, de aquí se sigue que, se pueden definir dos funciones min x_k y max x_k entre x_k y el hiperplano (valor de z).

La condición entre éstas dos funciones y el problema fundamental puede demostrarse a través del siguiente sistema de desigualdades.

$$Cx + dy - z = 0 \qquad (6.1')$$
$$Ax + Dy \leq b \qquad (6.2')$$
$$x \geq 0 \qquad (6.3')$$
$$y \geq 0 \qquad (6.4')$$
$$z \geq 0 \qquad (6.5')$$

Este sistema define un conjunto poliédrico convexo en un espacio de n+1 dimensiones y como el plano proyectado de un conjunto convexo es convexo, la proyección de éste conjunto en el plano (x_k, z) producirá un polígono convexo cuya cota superior (inferior) es una función cóncava (convexa) de la abscisa. Tal polígono se muestra en la figura 6.6.

El valor z^0 que toma z en el punto mas alto del polígono es el máximo valor que puede tomar sujeto al sistema de desigualdades dado. Esto es equivalente a decir que z^0 es el valor óptimo de la función z en el problema de programación lineal con restricciones (6.2), (6.3') y (6.4). Más aún x_k toma el valor x_{k0} en ésta solución. (si xk no estuviera en la base óptima, el pico del polígono podría estar en el eje de z). Las cotas del polígono son las funciones min x_k y max x_k. Estas funciones se pueden calcular resolviendo las dos familias de problemas de programación lineal definidas por (6.1'), (6.2), (6.3'), (6.4) y (6.5) con funciones "minimizar x_k" y "maximizar x_k" para todos los posibles valores de z usando programación lineal paramétrica. Un polígono de dos dimensiones de éste tipo podría determinarse para cada variable x del problema original discreto.

En la figura 6.5, x_k^0 cae entre los valores 2 y 3. Si el min x_k y el max x_k se trazan reduciendo z de z^0 hasta que se encuentra un primer valor entero de x_k en cada uno. Los dos valores z_k (2) y z_k (3) se obtienen de ésta forma en la figura 6.5 Esto es equivalente a examinar la trayectoria que se traza por dos puntos específicos de la intersección del hiperplano funcional con el

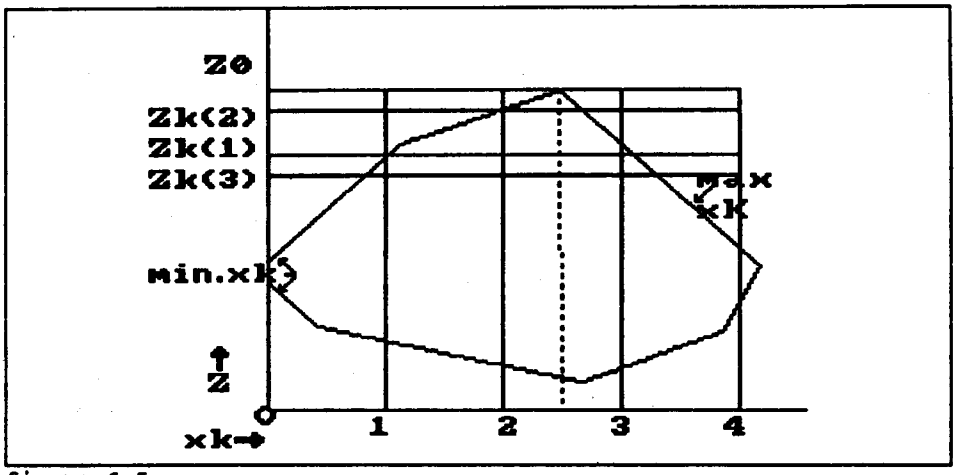


figura 6.5

conjunto convexo de soluciones factibles, cuando el hiperplano sistemáticamente se mueve hacia atrás de su posición máxima z^0 . Se puede decir de z_k (2) que es una cota superior del maximando ya que al no haber un valor mas alto de z puede x_k tomar un valor entero

La solución del problema de programación discreta se desarrollará usando sistemáticamente éste argumento. En aras de facilitar la exposición definimos un conjunto de problemas subsidiarios P(j) como sigue :

P(j): Maximizar z sujeto a las restricciones (6.2), (6.3′) y (6.4) y las restricciones adicionales en el sentido de que j de las variables x son enteros no negativos (j=0, 1, 2,...,n_1).

Sea Sj el conjunto de todas las soluciones factibles a problemas del tipo P(j) y sea S^* el conjunto de soluciones no factibles a cualquier problema del tipo P(j). El problema particular en que por ejemplo, x_2 y x_4 se restringen a ser enteros no negativos, se escribirá P(2; 2, 4) y el conjunto de sus soluciones factibles se escribe S_2(2, 4). En ésta notación, la solución requerida es el elemento de Sn1 para el cual z se maximiza. El valor de z de ésta solución está acotado por el máximo de z sobre el conjunto S_{n1-1} el cuál está acotado por el máximo de z sobre S_{n1-2} etc. La última cota superior alcanzada de ésta manera es z^0, el máximo valor de z sobre S_0.

REGLAS PARA AUMENTAR ARCOS Y NODOS

En el Paso 0, el árbol consiste de un sólo vértice etiquetado z^0 que es un elemento de S_0.

La decisión que ahora se toma es: dado que x_r está en la base de z^0 se restringe a tomar valores enteros no negativos, esto es, la atención se centra en el problema P(1;r). Sea $x_r = x_r^0$ en z^0. Las funciones min x_r y max x_r se trazan hasta los puntos $([x_r^0], z_{rm})$ y $([x_r^0]+1, z_{rM})$ respectivamente, donde:

$[x_r^0]$ es el mayor entero menor o igual a x_r^0

z_{rm} es el máximo valor de z sujeto a P(0) y la condición $x_r = [x_r^0]$, y

z_{rM} es el máximo valor de z sujeto a P(0) y la condición $x_r = [x_r^0] + 1$.

Los nodos z_{rm} y z_{rM}, si existen, son elementos de S_1. Se ha visto, así mismo, que ambos pueden evaluarse resolviendo el problema de programación estándar en (n-1) variables. Si uno de esos problemas dice "maximizar z sujeto a P(0) y a $x_r = [x_r^0]$"y no es factible, entonces no existe z_{rm} y se aumenta un nodo a S*. Esto implica que S_1 no contiene un elemento para el cual $x_r = [x_r^0]$, por lo cual sólo los valores de x_r que satisfacen $x_r \geq [x_r^0]+1$ necesitan considerarse en el desarrollo del árbol. Si no existen z_{rm} y z_{rM}, entonces x_r no se puede restringir a un valor entero y por lo tanto el problema no posee solución factible. Se adicionan entonces los dos arcos y nodos al Paso 1

Del Paso 2, $z_1 = \max(z_{rm}, z_{rM})$ al calcular z_{rm} y z_{rM}, se ha examinado cada valor de z entre z^0 y z^1, y se ha demostrado que z^1 es el máximo valor de z que es compatible con un valor entero de x_r. Suponga que $x_r = v$ en z^1, ésta ecuación se aumenta al problema como una nueva restricción que se aplica a cualquier rama del árbol que se traza después de z^1. La variable x_r se elimina de los cálculos, reduciéndose de ésta manera, las dimensiones del conjunto de soluciones exploradas a (n-1), y cada cota b_i se reduce por una cantidad $a_{ir}v$. Esta nueva restricción es válida con respecto a las soluciones de P(1;r) sobre el rango de valores de z para el que el único valor entero permisible de x_r es v. La extremidad superior de éste rango es z^1 y, ya que la cota superior del polígono de la figura 6.5 es una función cóncava de la abscisa, el extremo

inferior debe ser $z_r(v-1)$ o $z_r(v+1)$. El mayor de éstos dos es la segunda mejor solución de $P(1;r)^*$. Uno de los valores vecinos ya se conoce (es el $\min(z_{rm}, z_{rM})$) y una cota superior para el otro se puede encontrar por extrapolación de una función apropiada para x_r.

Si ésta cota superior excede el otro valor vecino, el valor z correspondiente se podría determinar en forma exacta. Los arcos y nodos del Paso 3 se aumentan al árbol.

Para ilustrar éste argumento, considere el problema de dos dimensiones mostrado en la figura 6.2.

Paso 0. La solución en A proporciona el primer vértice del árbol, z^0

Paso 1 Se selecciona la variable x_2 y se determinan los puntos B ($x_2 = 2$, $z = z_{2m} = z_2(2)$) y C ($x_2 = 3$, $z = z_{2m} = z_2(3)$).

Paso 2 $z^1 = z_{2M}$ para $v = 3$

Paso 3 z^1 no es la solución final, ya que $x_2 = 4$ está completamente afuera del conjunto convexo S_0, $z_2(4)$ está en S^*. El único posible valor entero de x_2 para todos los valores de z que satisfacen $z_2(2) < z \leq z_2(3)$ es $x_2 = 3$; ésta restricción es válida para la recta CD que es un subconjunto unidimensional del conjunto de soluciones factibles de P(0).

Paso 4 x_1 está ahora restringido a valores enteros. Uno de los nuevos vértices (z_{1m}) está en S_2 y el otro en S^*

Paso 2 $z^2 = z_{1m} > z_{2M}$

Paso 3 z^2 es un elemento de S_2; entonces el punto E, correspondiente a z_2 es la solución requerida. En éste punto, $x_1 = 2$, $x_2 = 3$

Claramente en cualquier etapa de la solución, las únicas variables x que pueden violar las restricciones discretas son las que están en la base actual. Entonces, restringiendo sucesivamente cada variable en la forma que se describió anteriormente, se encontrará una solución factible de $P(n_1)$ o se demostrará que tal solución no existe.

En el caso general, la primer cota superior en la función z0 es la solución óptima a P(0). La segunda cota superior es $z^1 \leq z^0$; z^1 es la solución óptima de P(1;r). Esta segunda cota superior se puede afinar encontrando la solución óptima a todos los problemas de tipo P(1) y seleccionando al menos uno de ellos como z^1. En éste análisis es más económico en términos de esfuerzo computacional trazar las funciones min y max de una sola variable en cada etapa.

Si z^1 no es una solución para $P(n_1)$, se escoge una nueva variable x entre aquellas que están en la base de z^1. El Paso 4 consiste en trazar las funciones max y min de ésta variable a los respectivos valores enteros mas cercanos por arriba y por debajo de su valor en z^1. Esto aumenta dos nuevos nodos al árbol, y del paso 2, z^2 es el máximo valor de z tomado sobre ellos y los valores vecinos $z_r(v-1)$ y $z_r(v+1)$ de z^1. El argumento completo se puede repetir en z^2 reemplazando z^1 como la cota superior actual en el valor óptimo de z. Continuando éste proceso, se forma un árbol donde cada uno de sus nodos representa un conjunto conocido de restricciones enteras (z^1, por ejemplo, representa $x_r = v$). Una ramificación termina si alcanza un nodo de S^* Podemos concluir de aquí que: si todas las ramas terminan en S^* el problema no tiene solución factible o si se alcanza un nodo z^F, todas las variables x son enteros no negativos. En principio, restricciones apropiadas se aplican a cualquier variable x que no se ha restringido por un valor entero, y un nodo z^F en $S_{n1}2$ se alcanza con el mismo valor z que el nodo etiquetado z^F. Ya que los nodos etiquetados son cotas superiores en la función zF debe ser la solución óptima requerida.

ALGORITMO 6.1 Land-Doig

PROPOSITO: Resolver el problema de programación entera mixta.

DESCRIPCION

La solución se construirá en forma de gráfica de árbol cuyos nodos son elementos de uno de los conjuntos S_j $j=0, 1,\ldots,n_1$ o del conjunto S^*. Los pasos de ésta construcción son

Paso 0. El primer nodo del árbol es la solución óptima de P(0), que está dada por la etiqueta z^0. Si ésta solución satisface también $P(n_1)$, terminamos se tiene la solución óptima.

Paso 1 Si z^0 no es el valor óptimo se traza un arco a cada uno de los dos puntos en S_1 (o posiblemente en S^*). Se evalúa z en esos puntos, si están en S_1. Si están en S^* es inmediato que no necesitan calcularse.

Paso 2 Si los nodos $z^0, z^1,\ldots,z^{k-1}$ ya se han etiquetado, el nodo más alto no etiquetado (de acuerdo al valor de z que no está en S^*) se etiqueta z^k.

Paso 3 La solución final se ha alcanzado cuando por primera vez un nodo etiquetado es un elemento de S_{a1}, esto ocurre tan pronto como todas las variables de una solución son enteros no negativos. Si z^k no es tal solución se supone que es un elemento de S_j. Se dibuja un nuevo arco originando así el nodo que está inmediatamente abajo de z^k (etiquetado, que no necesariamente z^{k-1}) y terminando en un punto en el mismo subconjunto de soluciones (S_j) como z^k o en S^*. Si este nuevo nodo está en S_j, su valor z es necesariamente menor o igual que z^k.

Paso 4 Se dibujan dos arcos de z^k a los puntos en S_{j+1} o en S^*. Nuevamente, si esos puntos están en S_{j+1} sus valores z son menores o iguales a z^k. Los últimos dos pasos aumentan tres nuevos arcos y nodos al árbol y regresamos al Paso 2.

Los nodos etiquetados forman una sucesión de cotas superiores no crecientes en el valor z de la solución final. Además el valor z de un nodo no etiquetado del conjunto S_j no puede exceder al del nodo etiquetado y entonces a la vez que z^k está etiquetado, es la mejor cota superior actual en el valor óptimo de z. Así, el método usado para aumentar arcos y nodos a la gráfica cubre todas las posibilidades, el algoritmo debe alcanzar la solución óptima de $P(n_1)$, siempre que tal solución exista.

Como se podrá observar los algoritmos enumerativos son usualmente más sencillos de entender si se ilustran geométricamente a través de un árbol compuesto de nodos y arcos. Un nodo también se puede hacer corresponder a un punto x^k y un arco une a ese nodo x^k con otro nodo x^{k+1}, donde definimos el último punto del anterior haciendo una de sus n-k variables libres igual a un entero no negativo. Como una variable libre x_i puede tener usualmente varios valores, es posible tener muchos arcos que emanen desde el nodo x^k. Como se puede ver en la figura 6.1 los nodos numerados 8, 9 y 10 se crearon haciendo x_1, una variable libre en el nodo 5 igual a 3, 4 y 2 respectivamente. Los nodos que no se han usado para crear otros nodos, o que no tienen arcos se llaman activos. Por ejemplo, el nodo 5 no es un nodo activo. También el número de nivel 1 se refiere al número de variables fijas.

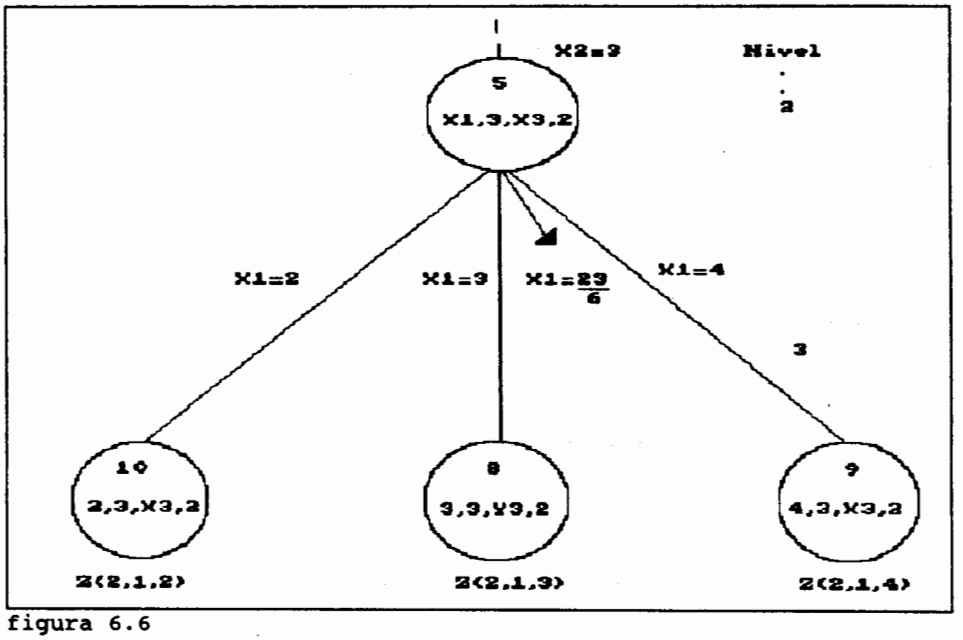


figura 6.6

Ejemplo 6.1

Considere el problema

$$\text{Max } z = 77.9\,x_1 + 76.8\,x_2 + 89.6\,x_3 + 97.1\,y_1 + 31.3\,y_2$$

Sujeto a:

$$\begin{aligned} 10.9\,x_1 + 3.6\,x_2 - 40.8\,x_3 + 43.9\,y_1 + 7.1\,y_2 + y_3 &= 82.3 \\ -86.8\,x_1 + 32.7\,x_2 + 24.3\,x_3 + 13.8\,y_1 - 12.6\,y_2 + y_4 &= 77.3 \\ 60.9\,x_1 + 68.9\,x_2 + 69\,x_3 - 56.9\,y_1 + 22.5\,y_2 &= 86.5 \end{aligned}$$

con $x \geq 0$, $y \geq 0$, x entero

Resuelva usando el método de Land-Doig.

ITERACION 1

Paso 0 Resolvemos el problema original sin incluir las restricciones de variables enteras, aplicando el método simplex y obtenemos:

VALOR DE LA FUNCION OBJETIVO $z^0 = 1165.50600$

Con valores en las variables de decisión:

$x_1 = 1.496009$
$x_2 = 0$
$x_3 = 5.021035$
$y_1 = 6.169743$
$y_2 = y_3 = y_4 = 0$

Paso 1. Las variables no enteras, son x_1 y x_3. Elegimos como variable de ramificación la que tenga el valor más apartado al entero más cercano; x_1 es la más lejana a un valor entero. Calculamos el primer valor entero en las funciones Min x_1 y Max x_1, es inmediato que 1 = Min x_1 y 2 = Max x_1

1.1 Min x_1. Sea $x_1 = 1$, el valor óptimo de la función objetivo es entonces

$z = 1051.89500$

con los siguientes valores en las variables

$x_1 = 1$
$x_2 = 0$
$x_3 = 4.408884$
$y_1 = 5.483788$
$y_2 = 1.485092$
$y_3 = y_4 = 0$

1.2 Max x_1 . Sea $x_1 = 2$. El valor de la función objetivo es

$$z = 770.698300$$

con los siguientes valores en las variables

$x_1 = 2$
$x_2 = 0$
$x_3 = 2.675005$
$y_1 = 3.864242$
$y_2 = y_3 = 0$
$y_4 = 132.570900$

Paso 2 El mayor de los dos valores objetivo es z = 1051.8952 con $x_1 = 1$, y se le asigna la etiqueta z. Para completar ésta etapa es necesario saber los valores de z para los enteros "cercanos".

Paso 3 Se calcula el mínimo de x_1 hasta su segundo valor entero: $x_1 = 0$ y se tiene que el valor de la función objetivo es:

$$z = 822.845600$$

Con valores en las variables

$x_1 = 0$
$x_2 = 0$
$x_3 = 3.174731$
$y_1 = 4.100841$
$y_2 = 4.479172$
$y_3 = y_4 = 0$

Estos tres valores de z se guardan en una lista en orden decreciente, para cada paso, así al inicio de cada Paso 2, se tomará un valor de la lista y al final de los pasos 3 y 4 se aumentarán 3 valores (inferiores). Los cálculos se terminan cuando todos los valores restantes de la lista son menores que un valor z asociado a una solución entera. En ésta etapa la lista está dada por:

z		obtenida en el paso
1051.8952	z^1	1
770.6983		1
822.8457		3

el valor más alto es 1051.8952 que se ha etiquetado como z^1 (Paso 2). La solución y restricciones en este punto son $z^1 = 1051.8952$, $x_1 = 1$, $x_2 = 0$, $x_3 = 4.4089$, $y_1 = 5.4838$, $y_2 = 1.4851$, $y_3 = 0$

Paso 4

4.1 Se calcula Min x_3. Se hace $x_3 = 4$ entonces

$$z = 990.638200$$

con valores en las variables:

$x_1 = 1$
$x_2 = 0.276776$
$x_3 = 4$
$y_1 = 5.151334$
$y_2 = 1.050713$
$y_3 = y_4 = 0$

4.2 Se calcula Máx x_3. Entonces $x_3 = 5$, y $x_1 = 1$. Sin embargo se obtiene una solución no factible.

Nota: Este resultado aparentemente anormal se produce porque max x3 es una función fallida de Z. Lo que implica que el punto para el cual $x_1=1$, $x_3=5$ cae en S*, ya que el problema con tales restricciones no es factible.

La lista de valores de Z es:

z		obtenida en el paso
1051.8952	z^1	1
770.6983		1
822.8457		3
990.6382		4

ITERACION 2

Paso 2 El mejor valor no etiquetado es 990.6382, que será z^2

Paso 3 Llevamos a min x_3 un paso mas adelante. Sea $x_3 = 3$ y se obtiene el siguiente valor de la función objetivo:

$$z = 840.659200$$

Con valores en las variables

$x_1 = 1$
$x_2 = 0.948665$
$x_3 = 3$
$y_1 = 4.336784$
$y_2 = y_3 = 0$
$y_4 = 0.331045$

Como en $x_3 = 3$ $z = 840.8229$, en éste punto la mejor solución no etiquetada es 990.6382 por lo cual la solución y restricciones son ahora: $z^2 = 990.6382$, $x_1 = 1$, $x_2 = 0.276$, $x_3 = 4$, $y_1 = 5.1513$, $y_2 = 1.0507$

Paso 4

4.1 Se calcula Mín x_2 y se hace $x_2 = 0$, con valor en la función objetivo:

$$z = 981.602300$$

y valores de las variables:

$x_1 = 1$
$x_2 = 0$
$x_3 = 4$
$y_1 = 5.070157$
$y_2 = 1.692975$
$y_3 = 0$
$y_4 = 18.263330$

4.2 Se calcula Máx x_2 y se hace $x_2 = 1$ Aquí se agrega otro punto a S^*, puesto que se tiene una solución no factible.

Entonces se tiene una solución en

$z = 981.6023$
$x_1 = 1$
$x_2 = 0$
$x_3 = 4$
$y_1 = 5.0702$
$y_2 = 1.0930$

Como éste valor de z es el mayor de los restantes en la lista, ésta es la solución óptima

El árbol de expansión que ilustra los cálculos anteriores se presenta a continuación

Algunas características importantes de éste algoritmo son las siguientes:

Cuando un subproblema en el nivel n es factible, se produce una solución al Problema Entero Mixto. Por lo tanto el algoritmo converge, ya que en el peor de los casos todos los nodos del árbol se enumeran explícitamente, y los nodos en el último renglón representan todas las combinaciones posibles de enteros.

Para reducir el esfuerzo computacional es probable que valga la pena introducir explícitamente la desigualdad $cx^1 + dy \geq z^0 + \epsilon$

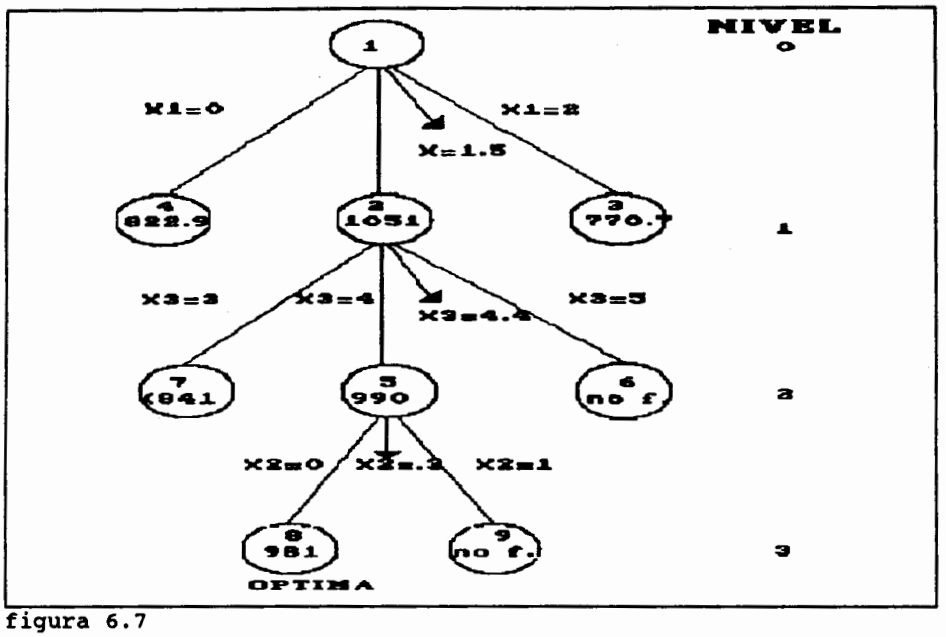


figura 6.7

para cada problema lineal PL^1, donde es un número pequeño positivo. Haciendo esto un problema lineal factible siempre indica un nodo activo y se tienen entonces, mejores soluciones del Problema Entero Mixto.

En éste algoritmo se selecciona para ramificar el nodo con la solución lineal mayor. Con ésta estrategia se pretende encontrar rápidamente una buena solución entera mixta, sin embargo éste procedimiento de selección puede requerir mucho almacenamiento computacional.

La variable que se selecciona para ramificar es aquella con componente más fraccionario

En éste algoritmo, la mayoría (si no todos) los problemas lineales se resuelven completamente. esto resulta costoso desde el punto de vista computacional, especialmente cuando la única información necesaria en un nodo puede ser el valor de la función objetivo.

El proceso continuo de ramificar y acotar, sugiere la designación de Ramificación y Acotamiento al algoritmo.

Algunos inconvenientes de éste algoritmo son:

Es posible que un número grande de arcos se puedan originar del mismo nodo. El problema es que esto no se puede predecir por anticipado, lo que complica el árbol de búsqueda y puede proporcionar un aumento severo en la memoria de la computadora.

Si el algoritmo se toma literalmente, será necesario resolver un problema lineal para cada ramificación en aras de determinar la cota superior propia. Esto lo hace aparecer como un método costoso, especialmente porque algunos de los nodos resultantes probablemente nunca se ramifiquen.

La determinación de una cota inferior sucede sólo como resultado de resolver el programa lineal en los diferentes nodos. Lo que significa que no obstante la importancia de cálculos truncados, no hay un método sistemático que asegure una cota inferior "estrecha" en una etapa temprana del procedimiento.

6.2 ALGORITMO DE DAKIN

Un algoritmo similar pero mas sencillo de instrumentar que el algoritmo de Land y Doig, fue propuesto por Dakin. El algoritmo es aplicable a problemas tanto lineales como no lineales, sin embargo sólo se tiene experiencia computacional para el caso de programación lineal. Para explicar las bases de tal algoritmo considere nuevamente el problema:

$$\text{Maximizar } z = cx + dy \tag{6.6}$$

sujeto a

$$Ax + Dy = b \tag{6.7}$$

$$x \text{ vector entero} \tag{6.8}$$

$$x \geq 0,\ y \geq 0 \tag{6.9}$$

El problema (6.6), (6.7), y (6.9) (sin la condición de que x sea entero) se denomina el problema continuo. Una solución al problema continuo que también satisfaga (6.8) se llama una solución entera; en caso contrario se dice que es una solución no entera. Suponga que existe un algoritmo (que llamaremos subalgoritmo) para encontrar soluciones a los problemas resultantes del problema continuo con la adición de cotas superiores o inferiores en alguna de las variables enteras. La existencia de tal algoritmo limita nuestro problema donde z es cóncava y (6.7) y (6.9) se especifican como una región convexa.

Se demostrará que si el subalgoritmo es finito, entendiendo por esto que resuelve cada subproblema en tiempo finito, y las variables enteras acotadas, entonces se alcanzará una solución óptima entera después de un número finito de cálculos.

Suponga que x_j es una variable entera y k es un entero. Dado que el rango $k < x_j < k + 1$ es inaceptable, podemos dividir todas las soluciones a las restricciones (6.7), (6.8) y (6.9) en dos grupos separados

$$\text{i) Soluciones en que } x_j \leq k \tag{6.10}$$

$$\text{ii) Soluciones en que } x_j \geq k + 1 \tag{6.11}$$

Al igual que el algoritmo de Land y Doig, en éste algoritmo se comienza con una solución al problema continuo. Si ésta solución es entera entonces es la solución del problema, de otra manera al menos una variable entera - digamos x_j- es no entera y toma un valor b_j en ésta solución. Se divide b_j en dos partes, entera y fraccional $[b_j] + f_j$ respectivamente, definida por:

$$b_j = [b_j] + f_j \tag{6.12}$$

donde $[b_j]$ es un entero y $0 < f_j < 1$ Sustituimos

$$k = [b_j] \tag{6.13}$$

en (6.10) y (6.11). Aparentemente no se satisface ninguna de esas relaciones con la actual solución no entera, por lo que aumentamos ésta restricción al problema continuo y resolvemos los problemas resultantes, Se repite el procedimiento para cada una de las dos soluciones obtenidas. La estructura lógica de éste proceso es la de un árbol, como el que se muestra en la figura 6.8

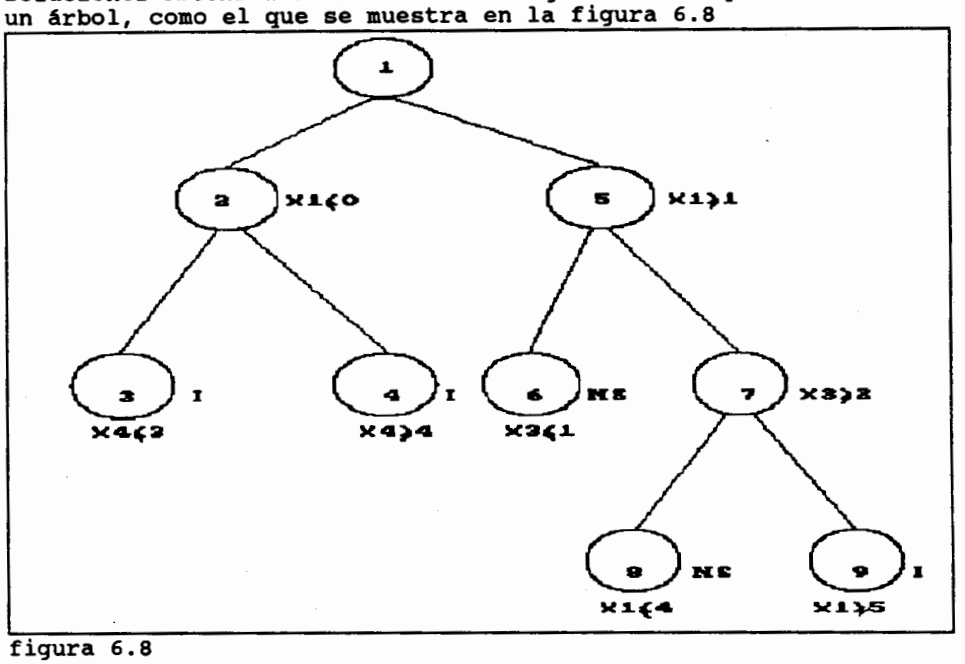


figura 6.8

El árbol siempre terminará en uno de dos caminos, puede alcanzar una solución entera (denotada por "I" en la figura) o se puede encontrar que el conjunto actual de restricciones no tiene solución (denotado por "NS"). La solución al problema completo será la mejor solución entera alcanzada de ésta manera.

El árbol -en general- no es único para un problema dado, ya que en cualquier etapa se tiene libertad de formar la siguiente restricción usando cualquier xi que no sea entera; la selección de diferentes xj producirá diferentes árboles. Como se podrá ver la selección puede tener un gran efecto en la cantidad de cálculos requeridos.

Note que sólo hay dos ramas y por lo tanto dos subárboles que salen de cada nodo -en contraste con el algoritmo de Land-Doig, donde puede salir cualquier número de ramas.

Se puede operar mas de una restricción en una variable al mismo tiempo. Por ejemplo, en el nodo 9 de la figura 6.8 se fijan las restricciones $x_1 \geq 1$ y $x_1 \geq 5$ a la vez. El número de restricciones que pueden operar al mismo tiempo en una variable está limitado por el rango de valores que puede tomar la variable. Así, si la restricción original (2) limita a x_j a un rango de $0 \leq x_j \leq v$ entonces el entero v acota a x_j en éste rango y es consistente con $x_j = k$; por ejemplo

$x_j \geq 1,\ x_j \geq 2,\ldots,x_j \geq k,\ x_j \leq k,\ x_j \leq k+1,\ldots,x_j \leq v-1.$

ALGORITMO 6.2: Dakin

PROPOSITO: Resolver el problema de programación entera mixta

DESCRIPCION

Paso 1 Se resuelve el problema entero por medio del método simplex de programación lineal. Si la solución es entera, pare, se ha encontrado la solución óptima.

Paso 2 Se escoge arbitrariamente una variable entera x_j cuyo resultado en el Paso 1 sea fraccionario e igual a f_j.

Paso 3 Se resuelven dos nuevos problemas similares al problema original, uno con la restricción adicional $x_{jk} \leq k$ y el otro con la restricción adicional $x_j \geq k+1$.

Paso 4 De los programas lineales resueltos en el paso 3 se incluyen en el análisis a seguir sólo aquellos programas cuya solución (entera o fraccional) sea mejor a cualquiera de las soluciones enteras conocidas.

Paso 5 Seleccione aquél programa lineal que tenga el mínimo valor de la función objetivo. Si las variables enteras tienen valor entero, se ha encontrado la solución óptima. Si no, regrese al paso 2 con la estructura del problema lineal resuelto en éste paso.

En los programas de cómputo que se han escrito para programación lineal, usando el método simplex como subalgoritmo y guardando el tableau completo, el requerimiento total de almacenamiento de datos es

$$\Sigma\, v_j + (m + 2)(n' + 2) \text{ palabras}$$

donde m es el número de restricciones en el problema continuo (excluyendo restricciones de cota superior si se usa el procedimiento de variables acotadas), y n'es el número de variables no básicas, v_j es el rango de valores que puede tomar la variable x_j, de donde la suma de estos posibles valores acotan los

requerimientos de almacenamiento.

Ejemplo 6.2

Suponga que se desea resolver el problema:

$$\begin{aligned} \text{Max } z &= 5x_1 + 2x_2 \\ \text{sujeto a} & \\ 2x_1 + 2x_2 + x_3 &= 9 \\ 3x_1 + x_2 + x_4 &= 11 \end{aligned}$$

Usando el algoritmo de Dakin

ITERACION 1

Paso 1 La solución óptima del problema continuo correspondiente tiene como valor de la función objetivo:

$$z = 18.75$$

Y los valores de las variables son:

$$x_1 = 3.25$$
$$x_2 = 1.25$$
$$x_3 = x_4 = 0$$

Pasos 2 - 3 Se escoge arbitrariamente $x_2 = 1.25$ y se resuelven dos problemas lineales distintos, uno con la restricción adicional $x_2 \leq [1.25] = 1$ y el otro con la restricción adicional $x_2 \geq [1.25] + 1 = 2$ Es decir:

Problema (1) : Consiste en resolver

$$\text{Max } z = 5x_1 + 2x_2$$

sujeto a

$$\begin{aligned} 2x_1 + 2x_2 + x_3 &= 9 \\ 3x_1 + x_2 + x_4 &= 11 \\ x_2 + x_5 &= 1 \end{aligned}$$

cuyo valor óptimo de la función objetivo es

$$z = 18.66667$$

Y los valores de las variables son:

$$x_1 = 3.333333$$
$$x_2 = 1$$
$$x_3 = .333333$$
$$x_4 = x_5 = 0$$

Problema (2) : Consiste en resolver

$$\text{Max } z = 5x_1 + 2x_2$$

sujeto a

$$\begin{aligned} 2x_1 + 2x_2 + x_3 &= 9 \\ 3x_1 + x_2 + x_4 &= 11 \\ x_2 - x_5 &= 2 \end{aligned}$$

cuyo valor óptimo de la función objetivo es igual a:

$$z = 16.5$$

y los valores de las variables son:

$$\begin{aligned} x_1 &= 2.5 \\ x_2 &= 2 \\ x_3 &= x_5 = 0 \\ x_4 &= 1.5 \end{aligned}$$

Paso 4. Como no ha habido ninguna solución entera en todo el proceso, se incluyen ambos problemas en el análisis

Paso 5 Como la mejor función objetivo hasta el momento corresponde a una solución no entera ($z = 18.67$), se regresa al paso 2 con el problema (1) como candidato

ITERACION 2

Paso 2 En el Problema (1) se escoge arbitrariamente la variable $x_1 = 3.33$ y se resuelven dos nuevos problemas. Uno que es igual al problema (1) más la restricción $x_1 \leq [3.33] = 3$. El otro que es igual al problema (1) más la restricción $x_1 \geq [3.33] + 1 = 4$. Específicamente:

Problema (3) : Consiste en resolver

$$\text{Max } z = 5x_1 + 2x_2$$

sujeto a

$$\begin{aligned} 2x_1 + 2x_2 + x_3 &= 9 \\ 3x_1 + x_2 + x_4 &= 11 \\ x_2 + x_5 &= 1 \\ x_1 + x_6 &= 3 \end{aligned}$$

Paso 3 Se obtienen las soluciones óptimas al problema lineal (3). El valor óptimo de la función objetivo es:

$z = 17$

y los valores de las variables son :

$$x_1 = 3$$
$$x_2 = x_3 = x_4 = 1$$
$$x_5 = x_6 = 0$$

El problema (4) no tiene solución factible y no se incluye en el listado.

Paso 4 Por tener una solución entera se incluye en el análisis.

Paso 5 Por ser el mejor valor de la función objetivo y además, ser entero, es la solución óptima.

El árbol que muestra el desarrollo del método se describe en la figura 6.9

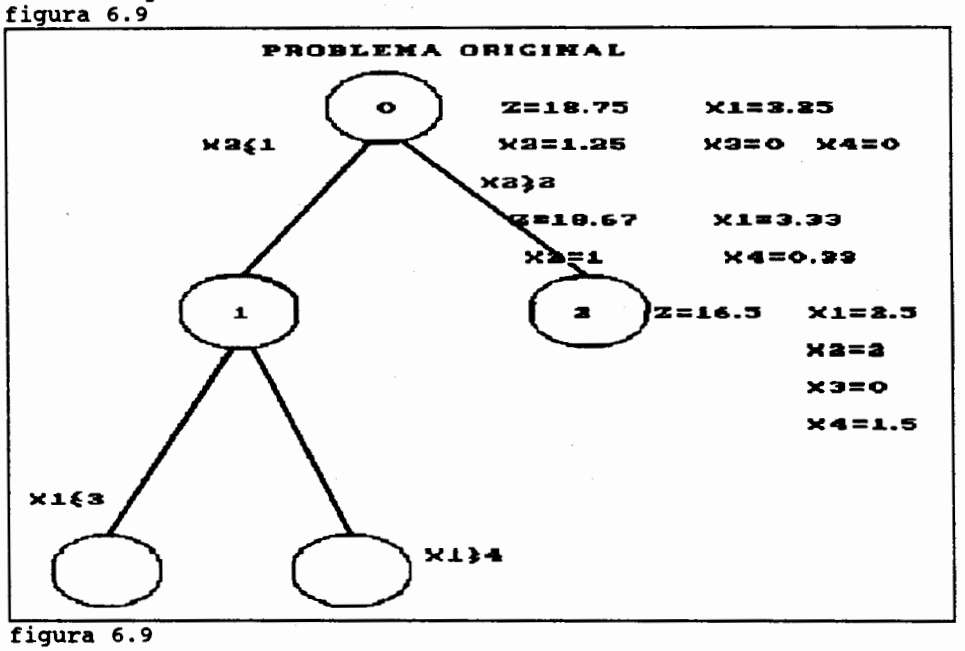


figura 6.9

6.3 COMPARACION CON EL METODO DE LAND Y DOIG

La cercana conección entre el algoritmo descrito y el propuesto por Land-Doig (1960) es evidente. La diferencia básica es que el algoritmo de L-D forza a las variables enteras a tomar valores enteros en lugar de aplicar cotas, de tal forma que es necesario buscar sobre el rango de valores enteros para los cuales z es mayor que el valor que toma en la mejor solución entera conocida. Si todas las variables enteras se restringen a los valores 0 y 1 los dos métodos son iguales y la única contribución del método presentado consiste en proporcionar un procedimiento computacional conveniente para llevarlo a cabo.

La convergencia de ambos métodos depende de la selección de la variable entera que se maneja en cada etapa. Sin embargo hay indicios que hacen suponer una convergencia más rápida para éste algoritmo en muchos casos, ya que hará una búsqueda sobre un rango más pequeño de valores para una variable entera que el algoritmo de Land y Doig, pues podemos detenernos tan pronto como se alcance una solución entera. Más aún, este método en algunos casos se saltará valores que alguna variable entera pueda tomar con el algoritmo de Land y Doig, y como se ha visto, se hace la ramificación en solo dos ramas y no en mas como el método anterior.

A continuación se desarrolla un ejemplo usando tanto el algoritmo de Land y Doig, como el de Dakin para observar claramente la diferencia entre ambos

Ejemplo 6.3

Considere el problema entero

$$\text{Max } z = -4x_1 - 5x_2 \qquad (6.14)$$

Sujeto a

$$-x_1 - 4x_2 \leq -5 \qquad (6.15)$$

$$-3x_1 - 2x_2 \leq -7 \qquad (6.16)$$

$$x_1, x_2 \text{ enteros no negativos} \qquad (6.17)$$

$$x_1 \geq 0, \quad x_2 \geq 0 \qquad (6.18)$$

Aplique el algoritmo de Land-Doig:

Paso 0 La solución al problema definido por (6.14), (6.15),(6.16) y (6.17) tiene como valor óptimo para su función objetivo:

$$z = -11.20$$

Cuyos valores en las variables son :

$$x_1 = 1.80$$
$$x_2 = 0.80$$

Paso 1 La cota superior de z y la solución actual al iniciar este paso es

$$z^0 = -11.2$$
$$x_1 = 1.8$$
$$x_2 = 0.8$$

Se escoge cualquiera de las dos variables, pues ambas están igual de lejanas al entero más próximo, sea $x_1 = 1.8$ y se hace max x_1 y min x_1

1.1 min x_1 implica x_1=1, cuyo valor óptimo de la función objetivo es igual a:z = -14. Y los valores en las variables:$x_1 = 1, x_2 = 2$

1.2 Máx x_1 implica x_1=2, con valor óptimo de la función objetivo igual a:z = -11.75, y los valores de las variables son: $x_1 = 2$, $x_2 = .75$

Paso 2 El mayor de los dos es z=-11.75 con x_1=2, y se le asigna la etiqueta z^1. Para completar ésta etapa es necesario saber los valores de z para los enteros que se pueden alcanzar "mas cercanos"

Paso 3 Se continua con max x_1 hasta su segundo valor entero; x_1=3 y se tiene que el valor óptimo de la función objetivo es: z = -14.5000000, y los valores de las variables: $x_1 = 3$, $x_2 = .5$

Estos tres valores de z se guardan en una lista y se tiene

z		obtenida en el paso
-11.75	z^1	1
-14		1
-14.5		3

El valor más alto es -11.75 que se ha etiquetado como z^1. La solución y restricciones en éste punto son:

restricciones solución	$x_1 = 2$
	$z^1 = -11.75$
	$x_1 = 2$
	$x_2 = .75$

Paso 4

4.1 Min x_2 implica x_2=0, no se tiene solución factible

4.2 Max x_2 implica x_2=1, cuyo valor óptimo de la función objetivo es: z = -13, y los valores de las variables son : $x_1 = 2$, $x_2 = 1$

La figura 6.10 ilustra el árbol de búsqueda asociado a éste problema

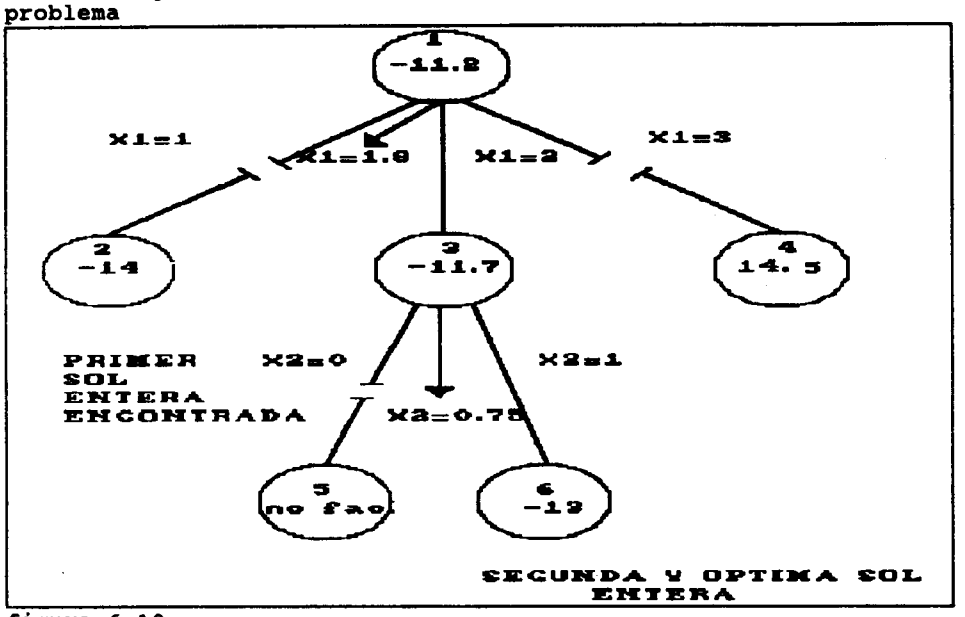


figura 6.10

Ahora desarrollamos el mismo ejemplo usando el algoritmo de Dakin

$$\max z = -4x_1 - 5x_2$$
sujeto a
$$-x_1 - 4x_2 \leq -5$$
$$-3x_1 - 2x_2 \leq -7$$

ITERACION 1

Paso 1 La solución óptima del problema lineal correspondiente tiene como valor óptimo de la función objetivo: $z = -11.2$ y con valores en las variables: $x_1 = 1.8$, $x_2 = .80$

Paso 2 Se escoge arbitrariamente $x_1 = 1.8$, y se resuelven los dos problemas lineales distintos uno con la restricción adicional $x_1 \leq [1.8]=1$ y el otro con la restricción adicional $x_1 \geq [1.8]+1=2$.

Paso 3 Se tienen los tableaus óptimos

Problema (1) Con $x_1 \leq 1$, se tiene que el óptimo en la función objetivo es z = -14. Y con valores en las variables: $x_1 = 1$, $x_2 = 2$

Problema (2) Con $x_1 \geq 2$, se tiene que el óptimo en la función objetivo es z = -11.75. Y los valores de las variables: $x_1 = 2$, $x_2 = .75$

Paso 4 Hay una solución entera sin embargo la mejor función objetivo corresponde a una solución no entera, y nos regresamos al Paso 2

ITERACION 2

Paso 2 Escogemos $x_2 = .75$ y se resuelven los dos nuevos problemas. Uno igual al problema 1 más la restricción $x_2 \leq [.75] = 0$ y el otro que es igual al problema 1 más la restricción $x_2 \geq [.75]+1 = 1$ es decir:

Problema (3) $x_2 \leq 0$, se tiene que no hay solución factible
Problema (4) Con $x_2 \geq 1$, se tiene que el valor óptimo de la función objetivo es igual a : z = -13. Cuyos valores en las variables son : $x_1 = 2$, $x_2 = 1$

La figura 6.11 ilustra el árbol de búsqueda correspondiente a este problema.

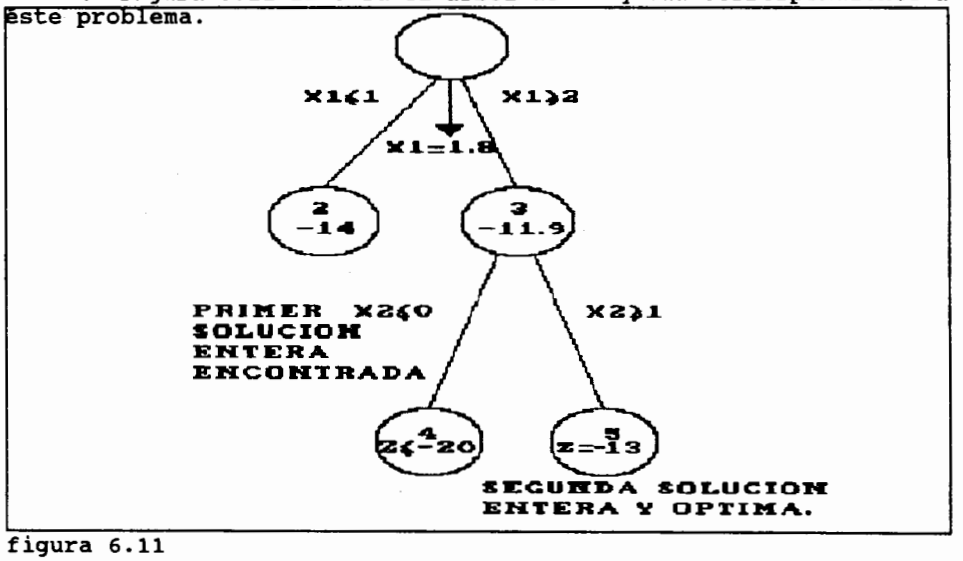


figura 6.11

6.4 ALGORITMO PARA LA SOLUCION DE PROBLEMAS DE PROGRAMACION ENTERA MIXTA DE DRIEBEEK

Este algoritmo, basado en el método de Land-Doig convierte a todas las variables enteras en binarias (cero-uno). Una vez realizado esto se sustituyen en el problema original y se encuentra la solución óptima utilizando programación lineal con ciertos ajustes, que se denominan costos penalizados. Dichos costos sirven para seleccionar la mejor variable entera que genera una bifurcación, y por lo tanto aceleran al método de R-A, es decir, lo hacen converger mas rápidamente a la solución óptima. El algoritmo fue propuesto por Norman J. Driebeek en marzo de 1966 y posteriormente en marzo de 1968 Stanley Zionts propuso algunas modificaciones al algoritmo que permiten considerar una restricción y n+1 variables para cada variable entera en lugar de n+1 restricciones y 2n+1 variables como se hacía originalmente. A continuación se presenta el algoritmo modificado por Zionts:

ALGORITMO 6.3: Driebeek

PROPOSITO: Resolver el problema entero mixto de programación lineal.

DESCRIPCION

PASO 1 Se convierte a todas las variables enteras en binarias de la siguiente manera: Sea x_i una variable entera, entonces

$$x_i = \sum_{k=1}^{n} k\ a_{ik}$$

donde $a_{ik} = 0$ o 1, para toda i, k

$$\Sigma a_{ik} = 1.$$

Si $a_{ik} = 1$, se implica que la variable entera es $x_i = k$ y se resuelve el problema no entero. Si se encuentra una solución óptima entera, parar.

PASO 2 Se calculan las penalizaciones de la siguiente manera:

a) Por cada variable a_{ik} no básica, el costo penalizado por alcanzar un valor $a_{ik} = 1$, es decir $x_i = k$, es precisamente el "precio sombra"de a_{ik} es decir el valor $z_{ik}-c_{ik}$ del tableau correspondiente.

b) Por cada variable a_{ik} básica, el costo penalizado por incrementar a_{ik} al valor 1, es decir, hacer $x_i = k$, es el mínimo incremento en la función objetivo por hacer $a_{ik} = 1$ (o sea, hay que usar análisis de sensibilidad al hacer el término x_{Bi} correspondiente al vector básico a_{ik} igual

a uno). El cambio mínimo en el valor de la función objetivo es la cantidad en la que cambia la función objetivo en la primer iteración del método dual simplex, al hacer el cambio correspondiente en el lado derecho del tableau. Este cambio es igual a

$(x_{Br}-1)$ Máx $\{(z_j-c_j)/Y_{rj} \mid Y_{rj}\leq 0\}$

donde x_{Br} es el valor del vector básico a_k y Y_{rj} son los valores del tableau correspondiente

PASO 3 De los costos penalizados calculados en a) se selecciona el mínimo. Si hay un empate, se escogen todos. Se combinan entonces con el costo mínimo penalizado calculado en b) y se selecciona aquella solución factible cuya suma de costos penalizados (los calculados en a) y en b)) sea mínima.

CONVERGENCIA

La experiencia computacional con el algoritmo ha sido buena, pues el algoritmo busca esencialmente una reducción mínima en el valor de la función objetivo cuando se activan las restricciones enteras. Porque las soluciones prueba se inician en la base de las penalizaciones, y porque aquellos puntos que tienen menores penalizaciones se prueban primero, se espera que la mejor o al menos una muy buena solución entera, se encuentre en una de las primeras pruebas; de los problemas que se han resuelto se ha encontrado la solución óptima en las primeras pruebas. En un problema que contiene 21 variables cero-uno o (2) 21 puntos, la solución óptima se encontró en la segunda prueba. Después de que una prueba se inicia, ésta sólo termina si:

a) En una iteración dual, el valor de la función objetivo disminuye por debajo de la solución encontrada.

b) Se encuentra una condición dual no acotado (primal no factible).

c) Una solución entera válida, mejor que la que se encontró previamente.

El tiempo total de corrida del algoritmo depende principalmente del número de puntos de la red en el problema y menos del significado de las variables enteras. Cuando las variables se usan para representar costos iniciales, la solución se encuentra con mayor facilidad que cuando las variables enteras se usen para decidir entre parejas de variables muy similares. La mayoría de los problemas enteros se han resuelto de k a 3k iteraciones, donde k es el número de iteraciones requeridas para alcanzar una solución continua.

Ejemplo 6.4

Considere el problema entero-mixto

$$\text{Máx } z = 5x_1+2x_2$$

sujeto a

$$\begin{aligned} 2x_1+2x_2+x_3 &= 9 \\ 3x_1+ x_2 +x_4 &= 11 \end{aligned}$$

$$x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0,\ x_4 \geq 0$$
$$x_1,\ x_2 \text{ enteros}$$

Resuelva usando el método de Driebeek.

Empezamos por hacer:

$$x_1 = 0\alpha_{10} + 1\alpha_{11} + 2\alpha_{12} + 3\alpha_{13} + \ldots$$

$$x_2 = 0\alpha_{20} + 1\alpha_{21} + 2\alpha_{22} + 3\alpha_{23} + \ldots$$

con

$$\alpha_{ij} = 0 \text{ o } 1,$$

y

$$\begin{aligned} \alpha_{10} + \alpha_{11} + \alpha_{12} + \ldots &= 1 \\ \alpha_{20} + \alpha_{21} + \alpha_{22} + \ldots &= 1 \end{aligned}$$

Se tiene que como x_1 puede alcanzar un valor máximo factible de 3, y x_2 de 4, se toman los primeros cuatro términos de α_{1j} y los primeros cinco términos de α_{2j}, y el problema original se convierte en:

$$\max \quad 5\alpha_{11} + 10\alpha_{12} + 15\alpha_{13} + 2\alpha_{21} + 4\alpha_{22} + 6\alpha_{23} + 8\alpha_{24}$$

sujeto a

$$\begin{aligned} 2\alpha_{11} + 4\alpha_{12} + 6\alpha_{13} + 2\alpha_{21} + 4\alpha_{22} + 6\alpha_{23} + 8\alpha_{24} + x_3 &= 9 \\ 3\alpha_{11} + 6\alpha_{12} + 9\alpha_{13} + \alpha_{21} + 2\alpha_{22} + 3\alpha_{23} + 4\alpha_{24} + x_4 &= 11 \\ \alpha_{11} + \alpha_{12} + \alpha_{13} + \alpha_{10} &= 1 \\ \alpha_{21} + \alpha_{22} + \alpha_{23} + \alpha_{24} + \alpha_{20} &= 1 \end{aligned}$$

EL TABLEAU

REN (BASIS)	α_{11}	α_{12}	α_{13}	α_{21}	α_{22}
1 ART	-5.0	-10.0	-15.0	-2.0	-4.0
2 ART	2.0	4.0	6.0	2.0	4.0
3 ART	3.0	6.0	9.0	1.0	2.0
4 ART	1.0	1.0	1.0	.0	.0
5 ART	.0	.0	.0	1.0	1.0

REN	α_{23}	α_{24}	x_3	x_4	α_{10}	α_{20}	x_B
1	-6.0	-8.0	.0	.0	.0	.0	.0
2	6.0	8.0	1.0	.0	.0	.0	9.0
3	3.0	4.0	.0	1.0	.0	.0	11.0
4	.0	.0	.0	.0	1.0	.0	1.0
5	1.0	1.0	.0	.0	.0	1.0	1.0

Después de dos iteraciones del método simplex, se obtiene la siguiente solución óptima no entera:

EL TABLEAU

REN (BASE)	α_{11}	α_{12}	α_{13}	α_{21}	α_{22}
1 ART	6.0	3.0	.0	.0	.0
2 α_{24}	-.50	-.25	.0	.25	.50
3 ART	-4.0	-2.0	.0	.0	.0
4 α_{13}	1.0	1.0	1.0	.0	.0
5 ART	.50	.25	.0	.75	.50

REN	α_{23}	α_{24}	x_3	x_4	α_{10}	α_{20}	x_B	
1	.0	.0	1.0	.0	9.0	.0	18.0	
2	.75	1.0	.125	.0	-.75	.0	.375	
3	.0	.0	-.500	1.0	-6.0	.0	.50	
4	.0	.0	.0	.0	1.0	.0	1.0	
5	.25	.0	-.125	.0	.75	1.0	.625	

El cálculo de costos penalizados al pasar de α_{ik} distinto de 1 a $\alpha_{ik} = 1$ es el siguiente: α_{ij} no básicas

α_{ij}	$x_i = j$	costo penalizado $= (z_{ij}-c_{ij})$
α_{10}	$x_1 = 0$	9
α_{11}	$x_1 = 1$	6
α_{12}	$x_1 = 2$	3
α_{21}	$x_2 = 1$	0
α_{22}	$x_2 = 2$	0
α_{23}	$x_3 = 3$	

α_{ij} básicos

α_{ij} x_i = j costo penalizado $(x_{Br}-1)$ $\underset{j=1,\ldots,n}{\text{Máx}} \{z_j-c_j/Y_{rj} \mid Y_{rj} \leq 0\}$

α_{13} $x_1 = 3$ (1-1) Máx {no hay candidatos} = 0
α_{20} $x_2 = 0$ (5/8 - 1) Máx {1/(-1/8)} = (-3/8)(-8) = 3
α_{24} $x_2 = 4$ (3/8 - 1) Máx {9/(-3/4),6/(-1.2),3/(-1/4)} = 15/2

Los costos penalizados mínimos se obtienen cuando:

$x_1 = 3$ y $x_2 = 1$ (solución factible)
con costo penalizado total 0+0 = 0

$x_1 = 3$ y $x_2 = 2$ (solución no factible)
con costo penalizado total 0+0 = 0

$x_1 = 3$ y $x_2 = 3$ (solución no factible)
con costo penalizado total 0+0 = 0

Como las dos últimas soluciones no son factibles, pues violan alguna de las restricciones y la primera solución es factible, es también óptima por lo que la solución queda:

$$z = 17, \quad x_1 = 3, \quad x_2 = 1$$

Como se podrá observar el algoritmo determina una cota superior (inferior) de la función objetivo en el caso de maximización (minimización), y por medio de los costos penalizados se ajusta con el costo penalizado mínimo, la solución óptima del problema entero. Este algoritmo puede dejar de iterar cuando la solución factible que se tiene se aproxima en un 80%-90% a la solución óptima.

Como veremos en el siguiente ejemplo, los costos penalizados sirven para seleccionar la mejor variable entera. Dada una variable básica $x_r=x_{Br}$ cuyo resultado final debe ser entero, y por el momento todavía es fraccionario, se tiene que el costo penalizado de hacer $x_r=[x_{Br}]$ (donde [x] es el número entero z más grande, menor o igual a x y <x> el número entero z más pequeño, mayor o igual a x) es:

$$(x_{Br}-[x_{Br}]) \underset{j=1,\ldots,n}{\text{Mín}} \{(z_j - c_j)/ Y_{rj} \; Y_{rj} \geq 0\}$$

mientras que el costo penalizado de hacer $x_r = \langle x_{Br} \rangle$ es:

$$(1 - x_{Br} + [x_{Br}])\ \text{Mín}\ \{(z_j - c_j)/ - Y_{rj}\ \ Y_{rj} \geq 0\}_{j=1,\ldots,n}$$

Entonces, asociada a cada variable básica x_r, que aún es fraccionaria, pero que en la solución óptima debe ser entera, se tienen dos costos penalizados, uno asociado con el cambio $x_r = [x_{Br}]$ y el otro con $x_r = \langle x_{Br} \rangle$. Para identificación, denote al primer costo penalizado por $[CP]_r$ y al segundo por $\langle CP \rangle_r$. La variable que se selecciona para ramificarse es aquella cuyo $[CP]_r$ ó $\langle CP \rangle_r$ es el máximo entre todas las variables básicas x_r, descartando por supuesto a las de holgura, que siendo aún fraccionarias, deben ser enteras en la solución óptima. Este proceso acelera al método, pues al elegir una variable básica con costo penalizado alto, se evita aumentar el número de iteraciones al eliminar implícitamente soluciones peores a las actuales.

Ejemplo 6.5

Resuelva el siguiente problema de programación entera usando el procedimiento descrito anteriormente para acelerar la convergencia

$$\max \quad z = 5x_1 + 2x_2$$

sujeto a

$$\begin{aligned} 2x_1 + 2x_2 + x_3 &= 9 \\ 3x_1 + x_2 + x_4 &= 11 \\ 20x_1 - 10x_2 - x_5 &= 51 \end{aligned}$$

$$x_i \geq 0 \text{ enteros, } i = 1,\ldots,5$$

La solución óptima del problema lineal asociado tiene como valor óptimo de la función objetivo

$$z = 18.750$$

Y los valores de las variables son:

$$\begin{aligned} x_1 &= 3.25 \\ x_2 &= 1.25 \\ x_3 &= x_4 = 0 \\ x_5 &= 1.5 \end{aligned}$$

El cálculo de los costos penalizados $[CP]_i$, $\langle CP \rangle_i$, i = 1,2 para las variables básicas (x_1, x_2) es:

$$[CP]_1 = (3.25-3)\ \text{Mín}\ \{0/1,\ 1.5/.5\} = 0$$
$$\langle CP \rangle_1 = (1-3.25+3\ \text{Mín}\ \{-.25/-.25\} = .75$$

$[CP]_2 = (1.25-1)$ Mín $\{0/1, .25/.75\} = 0$
$\langle CP \rangle_2 = (1-1.25+1)$ Mín $\{1.5/-(-.5)\} = 0.75(3) = 2.25$

La variable que se selecciona es la x_2 y las dos nuevas ramificaciones estarán dadas por las nuevas restricciones $x_2 \leq 1$ y $x_2 \geq 2$.

1. Sea x2 ≤ 1

La solución óptima del problema lineal asociado es

(BASIS)	x_1	x_2	x_3	x_4	x_5	x_6	x_B
ART	.0	.0	.0	1.667	.0	.333	18.667
x_3	.0	.0	1.0	-.667	.0	-1.333	.333
x_5	.0	.0	.0	6.667	1.0	-16.667	5.667
x_1	1.0	.0	.0	.333	.0	-.333	3.333
x_2	.0	1.0	.0	.0	.0	1.0	1.0

Y el valor óptimo de la función objetivo es:

$$Z = 18.66667$$

Con valores en las variables

$x_1 = 3.333333$
$x_2 = 1$
$x_3 = .333333$
$x_4 = x6 = 0$
$x_5 = 5.666667$

Se calculan nuevamente los costos penalizados $[CP]_i$, $\langle CP \rangle_i$ para las variables básicas x_1 y x_2

$[CP]_1 = (3.333-3)$ Mín $\{0, 1.666/.333\} = 0$
$\langle CP \rangle_1 = (1-3.333+3)$ Mín $\{-.333/-.333\} = .66666$
$[CP]_2 = (1-1)$ Mín $\{0, .333/1\} = 0$
$\langle CP \rangle_2 = (1-1+1)$ Mín { No hay disponibles} = 0

Ahora vemos si hay solución para $x_2 \geq 2$

2. Con $x_2 \geq 2$ no se tiene solución factible, entonces escogemos del problema con $x_2 \leq 1$ la variable, con el mayor costo de penalización que es x_1, entonces planteamos los dos nuevos problemas: con $x_1 \leq 3$ y $x_1 \geq 4$ y se tiene 3. Sea $x_1 \leq 3$ cuyo valor óptimo de la función objetivo es :

$$z = 16.8$$

Y los valores de las variables:

$$x_1 = 3$$
$$x_2 = 0.90$$
$$x_3 = 1.20$$
$$x_4 = 1.10$$
$$x_5 = x_7 = 0$$
$$x_6 = 0.10$$

4. Con $x_1 \geq 4$, no hay solución factible

Como en éste problema no hay solución factible entonces se tendría que ramificar sobre el problema 3, sin embargo los costos penalizados asociados son iguales a cero por lo que la solución óptima está dada por $z = 17$, $x_1 = 3$, $x_2 = 1$ En la figura 6.12 se muestra el árbol de búsqueda asociado a éste problema

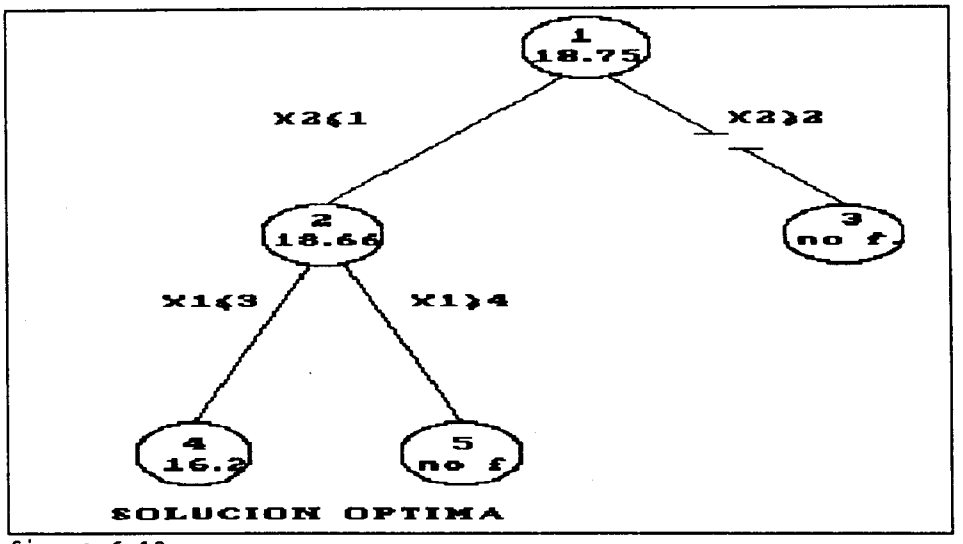


figura 6.12

Estos costos penalizados tienen la grandísima ventaja que permiten en cualquier instante del algoritmo de bifurcación y acotación (aunque no se haya obtenido aún la solución óptima), calcular una cota superior en el caso de maximización y una cota inferior en el caso de minimización entre la diferencia del valor de la función objetivo de la mejor solución entera obtenida hasta ese momento y el valor óptimo de la función objetivo. Esta cota se calcula de

$$\frac{z_j - c_j}{Y_{rj}} (x_{Br} - [x_{Br}]), \text{ para } Y_{rj} > 0 \quad \forall\, r,j$$

$$\text{Máx}\{\text{Mín}_{r\in I,\, j\in N}$$

$$\frac{z_j - c_j}{-Y_{rj}} (1 - x_{Br} + [x_{Br}]), \text{ para } Y_{rj} < 0$$

donde I es el conjunto de todos los vectores básicos en la iteración en cuestión (cuando se para el algoritmo) y N es el conjunto de todos los vectores no-básicos.

Ejemplo 6.6

Considere el siguiente problema entero

$$\text{Mín } z = 7x_1 + 3x_2 + 4x_3$$

sujeto a

$$\begin{aligned} x_1 + 2x_2 + 3x_3 - x_4 &= 8 \\ 3x_1 + x_2 + x_3 - x_5 &= 5 \end{aligned}$$

$$x_1,\ x_2,\ x_3,\ x_4,\ x_5 \geq 0, \quad \text{enteros}$$

Se supone que después de ramificarse como lo indica la figura 6.13 se ha obtenido una solución entera y se para el algoritmo, se desea conocer cuan alejada se encuentra esa solución de la solución óptima. El tableau asociado al nodo 2 es:

		z	x_1	x_2	x_3	x_4	x_5	x_B
	z	1	5	1	0	0	4	-17
vectores en	x_3	0	3	1	1	0	-1	2
la base	x_4	0	8	1	0	1	-3	4

Calculando los costos penalizados $[CP]_r$ y $\langle CP \rangle_r$, para toda variable básica (sin incluir a las de holgura) (x_3) se tiene

$$\text{Máx } \{ \text{Min } \{(2-2)(5/3),\ (2-2)(1/1) \},\ \text{Mín}\{(1-2+2)(4/-(-1))\} \}$$
$$= \text{Máx}\{\text{Mín } (0,0),\ \text{Min}\{4\}\}$$
$$= \text{Máx}\{0,4\} = 4$$

Lo que quiere decir que la función objetivo óptima no puede tener un valor mayor a -17+4 = -13 (puesto que se está minimizando

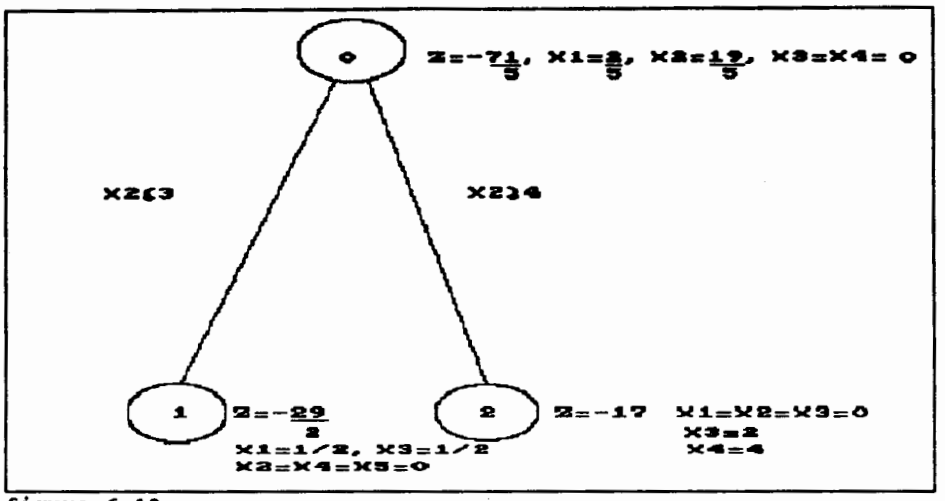


figura 6.13

se obtiene una cota inferior, en el caso de maximización se obtiene una cota superior, dada por el valor de la función objetivo al momento de parar, menos el valor del costo penalizado). Es decir 13 es una cota inferior del valor óptimo de la función objetivo del problema entero. De hecho el valor óptimo es:

$$z = 15$$
$$x_2 = 5$$
$$x_4 = 2$$
$$x_1 = x_3 = x_5 = 0.$$

CAPITULO 7
EL PROBLEMA DEL AGENTE VIAJERO
Y
EL PROBLEMA DE LA MOCHILA.

"Tiresias me ordenó que recorriera muchísimas ciudades, llevando en la mano un manejable remo, hasta llegar a aquellos hombres que nunca vieron el mar, ni comen manjares sazonados con sal, ni conocen las naves de purpúreos flancos, ni tienen noticia de los manejables remos que son como las alas de los bajeles."
Homero. La Odisea.

En el pasado reciente, los algoritmos de ramificación y acotamiento se han formulado para resolver una amplia variedad de problemas de optimización combinatoria. Entendiendo por problema combinatorio a aquél que asigna valores numéricos discretos a algún conjunto finito de variables X, de tal forma que satisfaga un conjunto de restricciones y minimice alguna función objetivo. Dos de tales problemas, tales como el del agente viajero y el de la mochila los proporcionamos como representativos no del rango de todas los posibles problemas combinatorios, pero si del tipo de problemas a donde se puede aplicar ramificación y acotamiento. La función objetivo que puede ser no lineal, discontinua o no necesariamente definida matemáticamente, se puede resolver con R-A.

Problemas como el del agente viajero o el de la mochila se conocen como no polinomiales completos o NP-completos. Un problema se dice polinomial si existe un algoritmo para el cual el tiempo requerido para su solución, está acotado por una función polinomial del tamaño del problema (donde entendemos el tamaño del problema como la longitud de un código, por ejemplo binario de los datos del problema). Se tiene así por ejemplo que en una gráfica G = [X, A] con N=X nodos y M = A arcos, una ruta más corta se encuentra a lo mas en un tiempo O(MN), un flujo máximo en un tiempo $O(N^3)$, un árbol de peso mínimo en un tiempo $O(N^2)$.

Un algoritmo se dice no determinístico si contiene afirmaciones que permiten una selección además de las afirmaciones usuales (deterministicas). La clase de problemas que se pueden resolver en tiempo polinomial por algoritmos no determinísticos se llaman de clase NP, en otras palabras, si para cada instrucción seleccionada, la selección es correcta, entonces el tiempo de computadora es polinomial. Por otro lado si se enumeran las posibles selecciones el algoritmo se vuelve determinístico pero con tiempo de computadora exponencial.Un problema NP-completo es aquel para el cual las únicas soluciones conocidas consisten, en esencia, en intentar todas las posibilidades. Tanto en el caso del problema del agente viajero como en el de la mochila los algoritmos que se basan en el método de R-A tienen un tiempo de computadora exponencial.

7.1 EL PROBLEMA DEL AGENTE VIAJERO

El problema del agente viajero es un problema clásico en optimización combinatoria, y uno de los problemas más viejos en la optimización de redes que ha atraído mucho la atención en los últimos cuarenta años. Cuenta con numerosas aplicaciones, por ejemplo, en problemas de distribución donde se tiene que el almacén central de una compañía desea distribuir materia prima a cada una de sus sucursales a un costo mínimo o problemas de carteros, ¿ cómo debe el cartero atravesar la ciudad para repartir la correspondencia minimizando el tiempo de viaje?. O considere otra situación: una fábrica produce n artículos y se debe cambiar el montaje siempre que cambia la producción de artículos, conociendo el costo de montaje entre los artículos se desea encontrar una secuencia de producción óptima.

Plantear el problema es muy sencillo y lo haremos a continuación:

Un agente viajero debe visitar cada ciudad de su territorio exactamente una vez y regresar a su punto de partida. Dado el costo del viaje entre cada par de ciudades, ¿como debe planear su itinerario de tal forma que visite cada ciudad una sola vez y el costo total del recorrido sea mínimo?

En términos de la teoría de redes, el problema consiste en encontrar en una gráfica completa de n nodos, un circuito de peso mínimo de longitud n. Recuérdese que una gráfica completa es aquella en la cual entre cualquier par de nodos existe un arco que los une. Considere por ejemplo, un problema con cinco ciudades, como se muestra en la figura 7.1

El peso del circuito o costo de la ruta (a, b, c, d, e, a) es 206, mientras que el circuito de peso mínimo es (a, b, e, c, d, a) con peso 89.

Hay varias variantes a éste problema, pero primero comenzaremos por precisar conceptos:

Sea $W = [w_{ij}]$ el peso de la matriz de la gráfica, entonces se puede tener que los pesos en los arcos pueden no ser simétricos $w_{ij} \neq w_{ji}$, en cuyo caso estamos tratando con una gráfica dirigida y resolviendo un problema del agente viajero asimétrico. O algunas veces la gráfica dada no es completa, lo que significa que existen parejas de nodos que no están directamente conectados por arcos. No

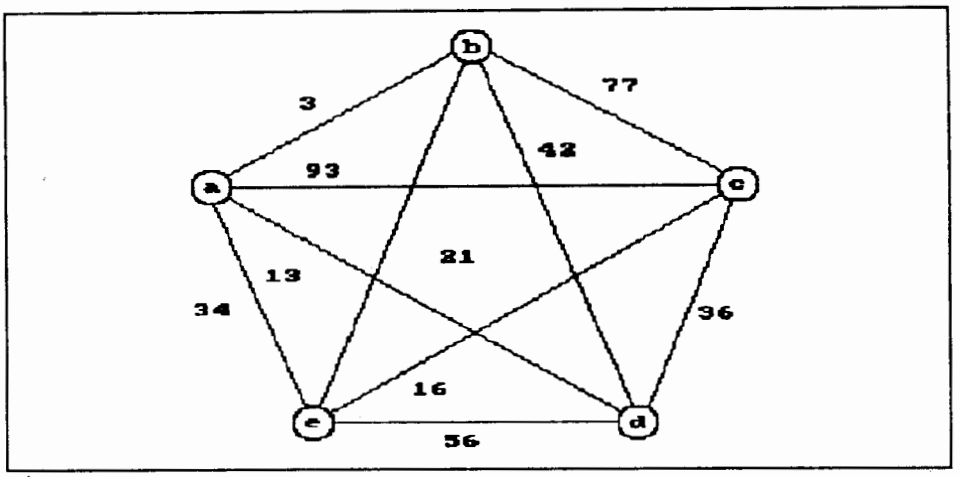


figura 7.1

obstante, que toda gráfica completa siempre tiene circuitos de longitud n, una que no lo es puede no tener circuitos de longitud n. Por ejemplo la gráfica en la figura 7.2 no tiene circuitos de longitud 5, en tal caso el problema del agente viajero no tiene solución.

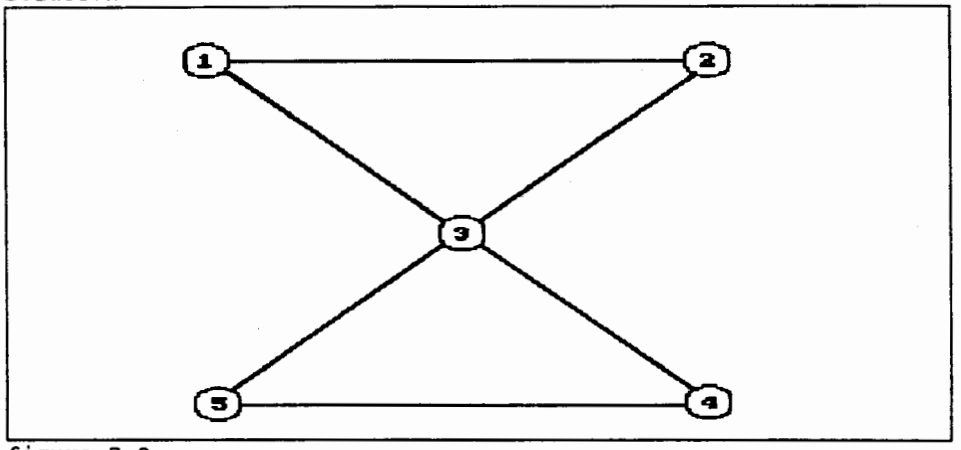


figura 7.2

El problema de determinar si una gráfica dada con n nodos tiene un circuito de longitud n se conoce como un problema de circuito hamiltoniano, originado en un juego inventado por el matemático irlandés Sir William Rowan Hamilton en 1859. Trataba sobre un viaje alrededor del mundo, el cual se representaba en forma simplificada por un dodecaedro (poliedro de 12 caras pentagonales con 20 vértices), y se requiere que se pase una sola vez por cada vértice o ciudad, usando solamente las caras del dodecaedro y se regrese al punto inicial. Este juego es equivalente a buscar un circuito hamiltoniano en la figura 7.3

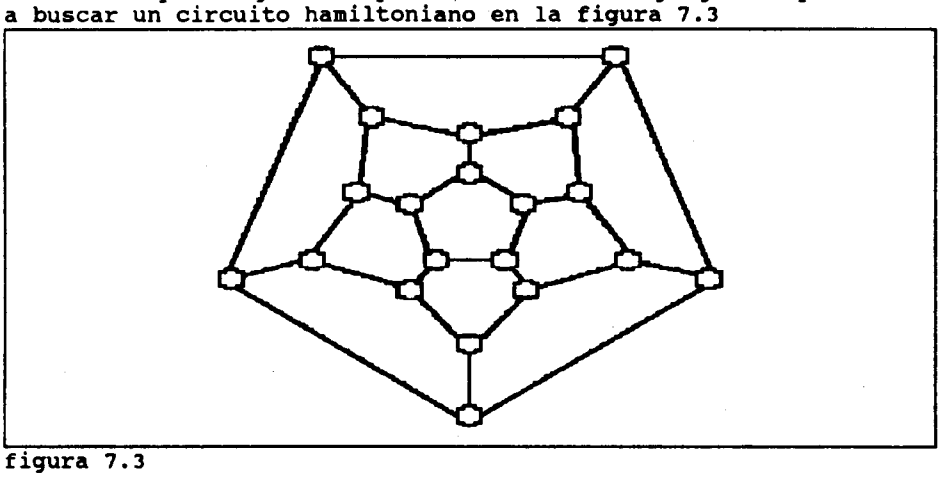


figura 7.3

Un circuito que pasa por cada nodo en la gráfica es llamado un circuito hamiltoniano. Si el agente viajero no quiere regresar al nodo inicial, después de visitar todos los nodos en la gráfica exactamente una vez, el problema se convierte entonces en encontrar la trayectoria de peso mínimo de longitud (n-1) y no un circuito de longitud n.

Se podrá notar un cambio muy drástico si al resolver éste problema se permite que el agente viajero visite un nodo mas de una vez (si ésto es más barato). Por ejemplo en la figura 7.4 se tiene que si se permite que visite un nodo mas de una vez la ruta óptima (a, b, c, d, e, c, a) que además no es un circuito tiene un peso de 60, pero si no se permite que visite un nodo mas de una vez, la ruta (a, b, c, d, e, a) es la única solución posible, con un peso de 140.

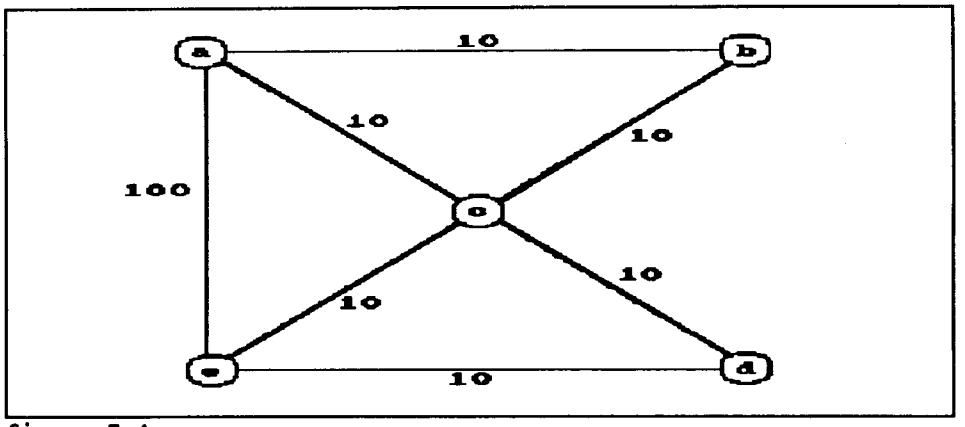


figura 7.4

La condición bajo la cual al menos uno de los circuitos hamiltonianos tenga un circuito de peso mínimo es que los pesos satisfagan la desigualdad del triángulo $w_{ij} \leq w_{ik} + w_{kj}$.

Una forma de resolver el problema del agente viajero es haciendo una enumeración exhaustiva de todas las soluciones posibles y entre ellas escoger la mejor. Hay n! permutaciones de los n nodos, pero solo (n-1)! de ellas son circuitos hamiltonianos distintos (en una gráfica dirigida completa), porque podemos fijar alguno de los n nodos, y posteriormente hay (n-1)! formas de acomodar los (n-1) nodos restantes en el circuito. En caso de que la red dada no sea una red dirigida entonces hay (n-1)!/2 circuitos distintos.

Teóricamente, el problema se puede resolver generando los (n-1)! circuitos y comparando sus pesos, sin embargo como método éste es muy ineficiente. Por ejemplo, dada una gráfica de 20 nodos se tienen 19! > 10^{17} circuitos, por lo que se requieren años de cálculos continuos para resolver el problema.

Como se podrá ver el problema del agente viajero es uno de los problemas de optimización combinatoria para los cuales no se conoce un algoritmo eficiente de tiempo polinomial. Todos los algoritmos conocidos requieren de un tiempo de computadora exponencial en el número n de ciudades.

Para salvar ésta dificultad computacional hay dos formas de resolverla:

1. Usando técnicas refinadas tales como ramificación y acotamiento o programación dinámica, que reducen drásticamente el efecto que se da en enumeración exhaustiva. Tales técnicas refinadas enumerativas son buenas para encontrar una solución óptima, pero en el peor de los casos si se tiene un problema muy grande pueden requerir un número exponencial de cálculos que se hacen prohibitivamente grandes.

2. Empleando métodos de solución aproximados pero rápidos (en tiempo polinomial), que no producen una solución óptima, pero si soluciones subóptimas que estén aceptablemente cercanas a la óptima.

Ambos métodos son útiles, sin embargo para nuestros propósitos sólo discutiremos el caso de ramificación y acotamiento.

7.2 ALGORITMO DE RAMIFICACION Y ACOTAMIENTO

"Entre éstos dos caminos vacilo continuamente, y cuando siento que he explorado al máximo las posibilidades de uno, me lanzo al otro y viceversa"
Italo Calvino. Seis Propuestas para el Próximo Milenio.

Un método de ramificación y acotamiento para un problema en particular queda definido al especificar sus operaciones de R-A. En el caso del problema del viajero, existen tres métodos de ramificación y acotamiento :

1. Construcción del circuito según la matriz reducida.
2. Eliminación de subcircuitos a partir de la solución de problemas de asignación.
3. Construir la trayectoria basándose en el árbol de expansión mínima.

En nuestro caso solo desarrollaremos el primero, pues no es nuestra intención hacer un análisis exhaustivo del problema del agente viajero, quienes deseen ver mas sobre el mismo pueden consultar [15]

METODO DE RAMIFICACION Y ACOTAMIENTO BASADO EN LA MATRIZ REDUCIDA

El funcionamiento de éste método se basa en la construcción del circuito hamiltoniano según la matriz reducida. Para ilustrar la idea de éste método, veamos el siguiente ejemplo:

Ejemplo 7.1

Sea la matriz de tiempos D que se presenta en la Tabla 7.1, entonces teniendo el tiempo de duración de cada uno de los viajes entre las distintas paradas, se desea encontrar el recorrido hamiltoniano de menor duración.

TABLA 7.1

	A	B	C	D	E	F
A	∞	27	43	16	30	26
B	7	∞	16	1	30	25
C	20	13	∞	35	5	0
D	21	16	25	∞	18	18
E	12	46	27	28	∞	5
F	23	5	5	9	5	∞

Si T es la duración de un recorrido hamiltoniano asociado a la matriz de tiempos de la tabla 7.1 entonces la duración de ese mismo recorrido con la matriz de tiempos obtenida de restar un escalar h_i al renglón i está dado por $T - h_i$, porque cada circuito contiene uno y solo un elemento de éste renglón. Lo mismo sucede si restamos un escalar h^j de una columna j (j = A, B, C, D, E, F). Una cota inferior del recorrido hamiltoniano óptimo es sencilla de obtener si restamos de cada renglón de la matriz de tiempos D, el mínimo elemento correspondiente. O bien

$$h_i = \min_i d_{ij} \geq 0$$

TABLA 7.2

	A	B	C	D	E	F
A	∞	11	27	0	14	10
B	6	∞	15	0	29	24
C	20	13	∞	35	5	0
D	5	0	9	∞	2	2
E	7	41	22	23	∞	0
F	18	0	0	4	0	∞

y entonces, en la matriz D' así reducida, restamos de cada columna j la constante

$$h^j = \min_j d_{ij}' \geq 0$$

La nueva matriz D^1, tiene elementos no-negativos, y contiene al menos un elemento cero en cada renglón y cada columna. Como se muestra en la tabla 7.3

TABLA 7.3

	A	B	C	D	E	F
A	∞	11	27	0	14	10
B	1	∞	15	0	29	24
C	15	13	∞	35	5	0
D	0	0	9	∞	2	2
E	2	41	22	23	∞	0
F	13	0	0	4	0	∞

Tomamos $h = \sum_i h_i + \sum_j h^j$

La duración $z(t)$ de un circuito hamiltoniano t de la matriz D es entonces igual a $z(t) = h + z'(t)$, donde $z'(t)$ es la duración del circuito t de la matriz D'. Ya que $z'(t)$ (porque $dij^1 \geq 0$), tenemos $z(t) \geq h$ donde h es una evaluación por cota inferior del conjunto Ω de circuitos hamiltonianos y se denota ev(Ω).
De la matriz dada D^1 obtenemos

$h_1 = 16$, $h_2 = 1$, $h_3 = 0$, $h_4 = 16$, $h_5 = 5$, $h_6 = 5$
y $h^1 = 5$, $h^2 = h^3 = h^4 = h^5 = h^6 = 0$

Un aspecto importante es que cada recorrido hamiltoniano asociado a la Tabla 7.3 tiene una duración no negativa y que difiere en tiempo del recorrido original en 48 unidades de tiempo. De donde una cota inferior de la duración del recorrido mínimo es 48 o bien

$$ev(\Omega) = \sum_i h_i + \sum_j h^j = 16 + 1 + 16 + 5 + 5 + 5 = 48$$

Una manera de proceder a la determinación del recorrido hamiltoniano de mínima duración consiste en particionar el conjunto de recorridos hamiltonianos como sigue:

a) Recorridos hamiltonianos que usan el arco AD
b) Recorridos hamiltonianos que no usan el arco AD

En el primer caso la matriz de tiempos entre localidades se reduce a una nueva matriz en donde se elimina al renglón A y la columna D. Asimismo, el tiempo entre la localidad D a la A se hace igual a infinito o a un número muy grande para evitar usar el arco DA; pues sabemos que no forma parte del recorrido hamiltoniano mínimo. Si no hacemos éste tiempo infinito, existe la posibilidad de la aparición de subcircuitos. La tabla 7.4 muestra la matriz reducida donde se han eliminado el renglón A y la columna D, el arco DA se hace igual a infinito

TABLA 7.4

	A	B	C	E	F
B	1	∞	15	29	24
C	15	13	∞	5	0
D	∞	0	9	2	2
E	2	41	22	∞	0
F	13	0	0	0	∞

Una cota inferior de la duración de los recorridos hamiltonianos asociados con esta tabla es sencilla de obtener si restamos una unidad a cada elemento del renglón uno como se muestra en la tabla 7.5

TABLA 7.5

	A	B	C	E	F
B	0	∞	14	28	23
C	15	13	∞	5	0
D	∞	0	9	2	2
E	2	41	22	∞	0
F	13	0	0	0	∞

Cabe hacer notar que como resultado de las manipulaciones anteriores podemos decir que los recorridos hamiltonianos asociados a la tabla 7.1, que usen el arco AD tiene una duración no menor de 49 unidades de tiempo, 48 acumuladas hasta la obtención de la tabla 7.3 y una unidad de tiempo al pasar de la tabla 7.4 a la tabla 7.5, por lo que 49 unidades es una cota inferior a la duración de los recorridos hamiltonianos que usan el arco AD.

Una cota inferior a la duración de los recorridos hamiltonianos que no usan el arco AD es sencilla de obtener, suponga que en general un arco (i,j) con $d_{ij}' = 0$ no se escoge. Ya que debemos dejar el punto i, debemos usar algún arco comenzando en

i y penalizarlo por α_i. También debemos usar algún arco en j, y podríamos penalizarlo por β_j, con lo cual $\theta_{ij} = \alpha_i + \beta_j$ es la penalización de no escoger (i, j). Dicha penalización θ_{ij} se puede interpretar en éste caso como un retardo.

Si ev (S) es la evaluación obtenida reduciendo la matriz por el nodo S, entonces ev (S) + θ_{ij} es una primer evaluación por cota inferior del nodo IJ, obtenido de S por no escoger el arco (i, j)

Calculamos el retardo θ_{ij} correspondiente a $d_{ij}' = 0$

$$\theta_{AD} = \alpha_A + \beta_D = 10 + 0 = 10$$
$$\theta_{BD} = \alpha_B + \beta_D = 1 + 0 = 1$$
$$\theta_{CF} = \alpha_C + \beta_F = 5 + 0 = 5$$
$$\theta_{DA} = \alpha_D + \beta_A = 0 + 1 = 1$$
$$\theta_{DB} = \alpha_D + \beta_B = 0 + 0 = 0$$
$$\theta_{EF} = \alpha_E + \beta_F = 2 + 0 = 2$$
$$\theta_{FB} = \alpha_F + \beta_B = 0 + 0 = 0$$
$$\theta_{FC} = \alpha_F + \beta_C = 0 + 9 = 9$$
$$\theta_{FE} = \alpha_F + \beta_E = 0 + 2 = 2$$

Separamos la pareja (i, j) que maximiza el retardo θ_{ij} para garantizar una mayor cota inferior de la duración de los recorridos hamiltonianos: $\theta_{AD} = 10$

Entonces el nodo AD tiene una evaluacion por cota inferior igual a 48 + 10 = 58

Una cuestión que conviene analizar en éste momento es: ¿por qué se efectuó la ramificación sobre el arco AD y no sobre otro que tuviese el tiempo entre localidades igual a cero en la tabla 7.2?

La razón es la siguiente: estrictamente se debió analizar para cada arco (i, j) con tiempo entre localidades igual a cero en la tabla 7.2 el retardo correspondiente asociado al eliminar ese arco y elegir el que tenga máximo retardo (que es el arco AD) para garantizar una mayor cota inferior de la duración de los recorridos hamiltonianos.

El siguiente paso consiste en calcular los retardos correspondientes a la tabla 7.5

$$\theta_{BA} = \alpha_B + \beta_A = 14 + 2 = 16$$
$$\theta_{CF} = \alpha_C + \beta_F = 5 + 0 = 5$$
$$\theta_{DB} = \alpha_D + \beta_B = 2 + 0 = 2$$
$$\theta_{EF} = \alpha_E + \beta_F = 2 + 0 = 2$$
$$\theta_{FB} = \alpha_F + \beta_B = 0 + 0 = 0$$
$$\theta_{FC} = \alpha_F + \beta_C = 0 + 9 = 9$$
$$\theta_{FE} = \alpha_F + \beta_E = 0 + 2 = 2$$

Se escoge el arco BA para efectuar la ramificación como sigue:
a) Recorridos hamiltonianos que usan el arco BA
b) Recorridos hamiltonianos que no usan el arco BA

En el primer caso partimos de la tabla 7.5 eliminando el renglón B y la columna A y haciendo el tiempo de A a B igual a infinito para evitar circuitos innecesarios. En éste momento, como AD y BA se han seleccionado para formar el recorrido hamiltoniano hacemos que el tiempo de DB sea infinito, la tabla 7.6 exhibe ésta situación

TABLA 7.6

	B	C	E	F
C	13	∞	5	0
D	∞	9	2	2
E	41	22	∞	0
F	0	0	0	∞

Con el propósito de tener un elemento cero en cada renglón y columna así como mejorar la cota inferior de los recorridos hamiltonianos procedemos a restar dos unidades del renglón dos en la tabla 7.6 para obtener:

TABLA 7.7

	B	C	E	F
C	13	∞	5	0
D	∞	7	0	0
E	41	22	∞	0
F	0	0	0	∞

Y se puede observar que una cota inferior de los recorridos hamiltonianos que usan BA es 51 unidades de tiempo.

Nuevamente calculamos las penalizaciones para saber que arco es el que se va a usar:

$$\theta_{CF} = \alpha_C + \beta_F = 5 + 0 = 5$$
$$\theta_{DE} = \alpha_D + \beta_E = 0 + 0 = 0$$
$$\theta_{DF} = \alpha_D + \beta_F = 0 + 0 = 0$$
$$\theta_{EF} = \alpha_E + \beta_F = 22 + 0 = 22$$
$$\theta_{FB} = \alpha_F + \beta_B = 0 + 13 = 13$$
$$\theta_{FC} = \alpha_F + \beta_C = 0 + 9 = 9$$
$$\theta_{FE} = \alpha_F + \beta_E = 0 + 2 = 2$$

Como se tiene 22 para EF entonces se incluye al arco EF y si no se incluye se tiene una penalización de 51 + 22 = 73 y la tabla 7.8 exhibe el resultado

TABLA 7.8

	B	C	E
C	13	∞	5
D	∞	7	0
F	0	0	∞

Nos fijamos en el primer renglón y restamos 5 para tener cero quedando

TABLA 7.9

	B	C	E
C	8	∞	0
D	∞	7	0
F	0	0	∞

lo que nos da un costo de 56 unidades nuevamente calculamos las penalizaciones:

$$\theta_{CE} = \alpha_C + \beta_E = 8 + 0 = 8$$
$$\theta_{DE} = \alpha_D + \beta_E = 7 + 0 = 7$$
$$\theta_{FB} = \alpha_F + \beta_B = 0 + 8 = 8$$
$$\theta_{FC} = \alpha_F + \beta_C = 0 + 7 = 7$$

de acuerdo a esto podemos elegir como arco a usar el CE o el FB, escogemos FB y obtenemos la tabla

TABLA 7.10

	C	E
C	∞	0
D	7	0

restamos siete en el renglón de D lo que nos da un costo de 63 unidades por usar el arco FB y obtenemos

TABLA 7.11

	C	E
C	∞	0
D	0	0

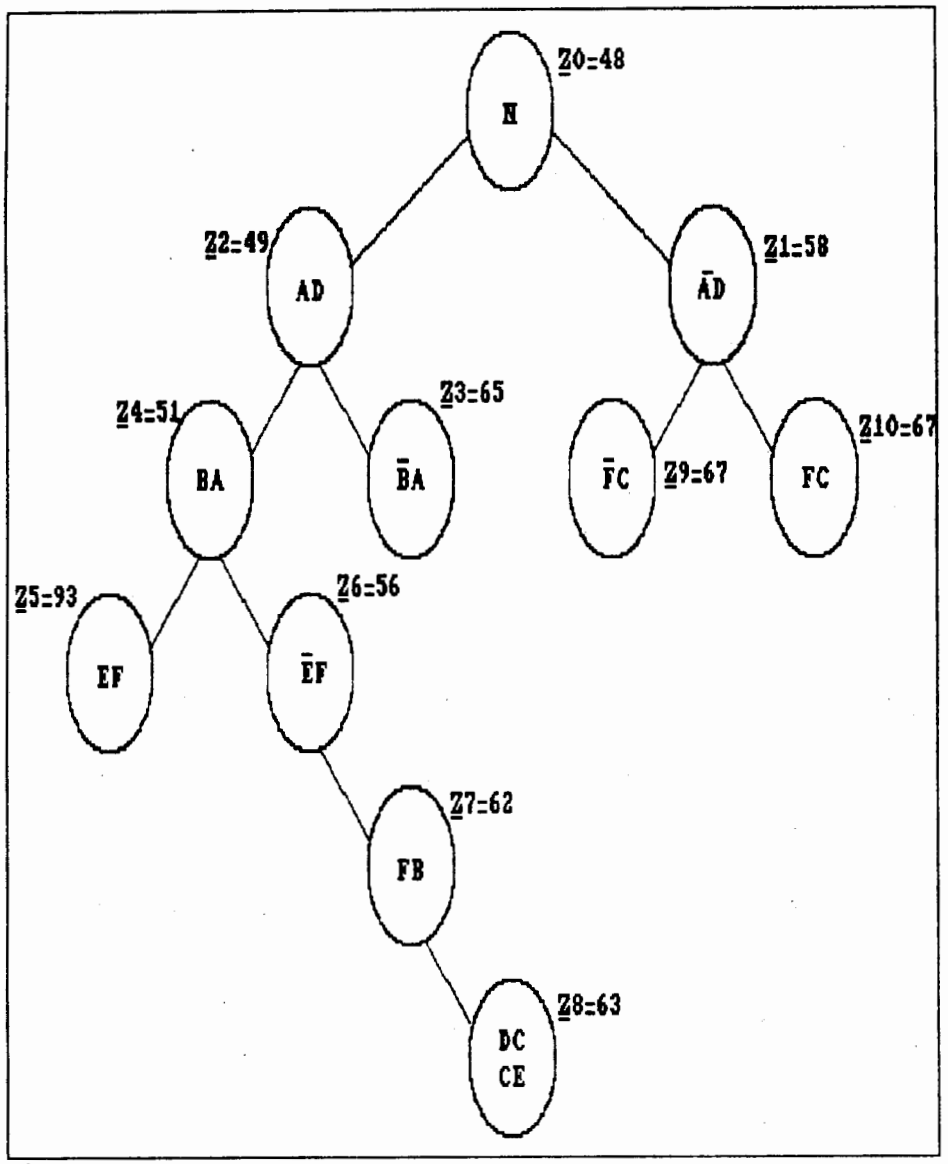
de donde los únicos arcos que quedan que se pueden usar son CE, DC y DE de donde escogemos CE y DC y asignamos infinito a DE para evitar posibles subcircuitos con lo que nos queda la tabla

TABLA 7.12

	C	E
C	∞	0
D	0	∞

De donde se concluye que el recorrido hamiltoniano óptimo es A-D-C-E-F-B-A con una duración de 63 unidades de tiempo.

Para verificar que ésta solución es óptima, debemos examinar el vértice AD cuya evaluación por cota inferior es 58 (< 63). La separación del vértice AD produce: usando FC una cota de 63 y sin usar FC una cota de 67, por lo cual el circuito propuesto es óptimo.


Z0=48
N
Z2=49
AD
Z1=58
$\bar{A}D$
Z4=51
BA
Z3=65
$\bar{B}A$
$\bar{F}C$
Z9=67
FC
Z10=67
Z5=93
EF
Z6=56
$\bar{E}F$
Z7=62
FB
Z8=63
DC
CE
figura 7.5

ALGORITMO 7.1: LITTLE-MURTY

PROPOSITO: Determinar el circuito hamiltoniano de costo mínimo dada una matriz de costos. El método utiliza la propiedad de la matriz de costos reducida para probar la inclusión o exclusión de un arco en el circuito.

DESCRIPCION

Paso 1 Dada la matriz de costos D, se efectúan sustracciones en los renglones y las columnas de la matriz D, sin permitir que aparezcan valores negativos. Con ésto obtenemos una cota inferior del problema del viajero al sumar los elementos que se restaron a los renglones y a las columnas. La matriz D' es la matriz reducida de D.

Paso 2 Sea S un nodo del árbol y ev(S) la cota inferior de éste nodo S, con cada arco (i, j) con $d_{ij}' = 0$, debemos usar algún arco comenzando en i y penalizarlo por α_i. También debemos usar algún arco en j, y podríamos penalizarlo por β_j, con lo cual $\theta_{ij} = \alpha_i + \beta_j$ es la penalización de no escoger (i,j). Se selecciona el arco que tiene el máximo de los θ_{ij}

Paso 3 Si el arco (i, j) no se selecciona, ev (S) + θ_{ij} es una cota inferior. Si el arco (i, j) se seleccionó, entonces la matriz se reduce omitiendo el renglón i y la columna j. Buscar la condición adicional sobre D' para excluir subcircuitos posibles.

Paso 4 Seleccione el vértice que tiene menor costo y vaya al paso 1.

7.3 EL PROBLEMA DE LA MOCHILA

Muchos problemas industriales se pueden formular como problemas de tipo mochila, por ejemplo problemas de cargo fijo, selección de proyectos, corte en inventarios, control de presupuestos, etc. La versión más popular del problema contiene sólo una restricción lineal, pero casi cualquier problema lineal entero y muchos otros problemas combinatorios se pueden reducir a él. El problema de la mochila se presenta también como un subproblema en varios algoritmos de programación lineal pura y entera.

Hay muchas versiones distintas del problema de la mochila, en nuestro caso consideraremos el problema de la mochila 0-1 que se expresa de la siguiente manera:

$$\text{Maximizar} \sum_{i=1}^{n} p_i x_i \qquad (7.1)$$

Sujeto a

$$\sum_{i=1}^{n} w_i x_i \leq W \qquad (7.2)$$

$$x_i = 0 \text{ o } 1 \qquad i=1,2,\ldots,n \qquad (7.3)$$

donde p_i, wi (i=1,2,...,n) y W son números enteros. En otros términos, suponga que se tiene que llenar una mochila con diferentes objetos con un beneficio p_i y peso w_i sin exceder un peso total dado W. El problema consiste en encontrar una asignación factible de objetos para que el valor total de los objetos en la mochila sea el máximo.

El problema de la mochila 0-1 es un caso especial del problema de la mochila acotado, que se define igual que el anterior y solo difiere en la restricción (7.3) ya que en éste caso se tiene $0 \leq x_i \leq b_i$, donde x_i es entero, i=1,2,...,n

En el problema de la mochila acotado, la mochila se puede llenar con a lo más b_i objetos del tipo i. En el problema general de la mochila, que a veces se denomina no acotado, la restricción (7.3) se relaja a $x_i \geq 0$, x_i entero, i=1,2,...,n

Sin pérdida de generalidad podemos suponer que los parámetros p_i, w_i y W en los problemas anteriores satisfacen las condiciones:

p_i y w_i son enteros positivos $\qquad i=1,2,\ldots,n$

$w_i \leq W$

$$\sum_{i=1}^{n} w_i > W$$

Los problemas de la mochila 0-1, acotado y generalizado a veces se conocen como problemas unidimensionales, donde el uno se refiere al número de restricciones lineales en el problema. Los más populares son loa de valor independiente (cuando $w_i=p_i$) y el problema de hacer cambios, éste problema consiste en encontrar el menor número de monedas de tipos o valores especificados w_i que constituyen exactamente un cambio dado V. Suponemos que se dispone de cada tipo de moneda en una cantidad ilimitada, formalmente el problema es

$$\text{Minimizar} \sum_{i=1}^{n} x_i$$

Sujeto a

$$\sum_{i=1}^{n} x_i w_i = W$$

$$x_i \geq 0,\ x_i \text{ entero } i=1,\ldots,n$$

Observe que como la restricción en éste problema es de igualdad no siempre existe solución, a menos que alguna de las monedas disponibles valga 1.

Los problemas tipo mochila unidimensionales se pueden generalizar de muchas maneras, la generalización más natural es aquella en que los objetos que tenemos que guardar pueden ponerse en m mochilas, cada una con capacidad Wj (j=1,2,...,m 1). Sea x_ij una variable 0-1 tal que $x_ij=1$ si el i-ésimo objeto se asigna a la j-ésima mochila. El problema 0-1 multimochila, se expresa como

$$\text{Maximizar} \sum_{i=1}^{n}\sum_{j=1}^{m} p_i x_{ij}$$

Sujeto a

$$\sum_{i=1}^{n} w_i x_{ij} \leq W_j \qquad j=1,2,\ldots,m$$

$$\sum x_{ij} \leq 1 \qquad i=1,2,\ldots,n$$

$$x_{ij} = 0 \text{ o } 1 \qquad i=1,\ldots,n;\ j=1,\ldots,m$$

La primer restricción significa que en una restricción factible de objetos no se sobrecarga ninguna mochila y la segunda, que cada objeto puede asignarse a lo más a una mochila, pueden formularse de aquí las versiones acotada y no acotada de éste problema.
Si antes de poner los objetos en una mochila se tienen que comprar, a un costo ci para el i-ésimo objeto, y se tiene una cantidad limitada de dinero C, se tiene entonces el problema de asignar objetos a la mochila que no pesen mas de W y no cuesten mas de C, entonces el problema se convierte en

$$\text{Maximizar} \quad \sum_{i=1}^{n} p_i x_i$$

Sujeto a

$$\sum_{i=1}^{n} w_i x_i \leq W$$

$$\sum_{i=1}^{n} c_i x_i \leq C$$

$$x_i = 0 \text{ o } 1 \qquad i=1,2,\ldots,n$$

En general, se pueden introducir muchas restricciones al asignar objetos a una mochila, el problema se convierte entonces en un problema de la mochila multidimensional, donde evidentemente se pueden considerar los casos acotado y no acotado.

Los problemas de tipo mochila a menudo se refieren como problemas de cargo, pero de hecho, el problema de cargo estándar consiste en asignar objetos dados con volúmenes conocidos a cajas, que tienen restricciones de capacidad, con el objeto de minimizar el número de cajas usadas. Sea k_j la capacidad de la j-ésima caja y w_i el volumen del i-ésimo objeto. El problema de cargo se define como sigue:

$$\text{Minimizar} \quad \sum_{j=1}^{m} y_j$$

Sujeto a

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad i=1,2,\ldots,n$$

$$\sum_{i=1}^{n} w_i x_{ij} \leq k_j y_j \qquad j=1,2,\ldots,m$$

$$y_j,\ x_{ij} = 0 \text{ o } 1;\ i=1,\ldots,n;\ j=1,\ldots,m$$

En un cargo factible, tenemos $x_{ij}=1$ si el i-ésimo objeto se pone en la j-ésima caja, y $y_j=1$ si se usa la j-ésima caja.

El problema de la mochila acotado o no acotado puede también representar el problema de cortar objetos unidimensionales (por ejemplo la longitud de un papel, vidrio y acero) en piezas pequeñas de valores y tamaños dados para maximizar el valor total de las piezas o minimizar el material que sobra.

Este pequeño panorama de las posibles generalizaciones y modificaciones del problema de la mochila 0-1 indica la variedad de problemas del mundo real que se pueden modelar por problemas que provienen de la familia de problemas de tipo mochila.

REDUCCION DEL PROBLEMA LINEAL ENTERO AL PROBLEMA DE LA MOCHILA.

Cada sistema de ecuaciones lineales con coeficientes enteros se pueden transformar en una sola ecuación lineal que tiene el mismo conjunto de soluciones enteras no negativas que el correspondiente sistema lineal entero. Entonces, las restricciones de un problema lineal entero se pueden primero transformar en una única restricción y así resolverlo como un problema de tipo mochila.

El paso básico del proceso de transformación agrega dos ecuaciones de tal forma que el conjunto de soluciones enteras no negativas no se altera. Describiremos un método que supone todas las variables acotadas.

Sean las dos ecuaciones enteras en variables acotadas

$$g_j(x) = b_j - \sum_{i=1}^{n} a_{ji}\, x_i = 0 \qquad j=1,2 \tag{7.4}$$

$$0 \le x_i \le u_i,\ x_i \text{ es entero} \qquad i=1,2,\ldots,n$$

Determinaremos ahora los multiplicadores válidos α_1 y α_2 tales que son distintos de cero y la ecuación

$$\alpha_1\, g_1(x) + \alpha_2\, g_2(x) = 0 \tag{7.5}$$

implica que:

$$g_1(x) = g_2(x) = 0$$

ya que (7.5) es equivalente a $g_1(x) + (\alpha_2 / \alpha_1)\, g_2(x) = 0$, podemos preguntar por sólo un multiplicador α tal que:

$$g_1(x) + \alpha\, g_2(x) = 0 \tag{7.6}$$

implica que:

$$g_1(x) = g_2(x) = 0$$

Usando cotas superiores en las variables, $g_j(x)$ $(j=1,2)$ se pueden acotar de la siguiente manera:

$$b_j - \sum_{i=1}^{n} a_{ji}^{+} u_i \leq g_j(x) \leq b_j - \sum_{i=1}^{n} a_{ji}^{-} u_i$$

donde $a_{ji}^{+} = \max(a_{ji}, 0)$ y $a_{ji}^{-} = \min(a_{ji}, 0)$.

Teorema 7.1

El vector entero x que satisface $0 \leq x \leq u$ es una solución de (7.4) si y sólo si es una solución de (7.6) con α que satisface.

$$\alpha > \max \{b_1 - a_{1i}^{-} u_i, -b_1 + a_{1i}^{+} u_i\}$$

Ejemplo 7.2

Aplicaremos el Teorema 7.1 al siguiente sistema de ecuaciones:

$$\begin{aligned} 4x_1 - x_2 + 2x_3 + x_4 + x_5 &= 0 \\ x_1 - x_3 - x_4 - x_6 &= -1 \end{aligned}$$

$$x_i = 0 \text{ o } 1 \qquad i = 1,2,\ldots,6$$

Por la enumeración de todos los posibles vectores 0-1 $(x_1,x_2,x_3,x_4,x_5,x_6)$ encontramos las únicas soluciones de éste sistema $(0,1,0,1,0,0)$, $(0,1,0,0,1,1)$ y $(0,0,0,0,0,1)$. Primero tratamos con dos multiplicadores arbitrarios $\alpha_1 = 1$ y $\alpha_2 = 1$. La ecuación agregada es de la forma

$$5x_1 - x_2 + x_3 + x_5 - x_6 = -1$$

y se tienen las tres soluciones anteriores mas $(0,1,0,0,0,0)$, $(0,0,0,1,0,1)$, $(0,1,1,0,0,1)$, $(0,1,1,1,0,1)$ y $(0,1,0,1,1,1)$. Sin embargo aplicando el Teorema 7.1 se encuentra que α debe ser mayor que 8. Para $\alpha = 9$, la ecuación válida agregada es de la forma

$$13x_1 - x_2 - 7x_3 - 8x_4 + x_5 - 9x_6 = -9$$

Una vez que sabemos como agregar dos ecuaciones en una, podemos aplicar éste proceso iterativamente a un sistema de ecuaciones arbitrario. Este método es de uso limitado para programas lineales enteros debido al rápido crecimiento en los valores de los coeficientes conforme se van agregando restricciones.

METODOS COMPUTACIONALES

Desde el punto de vista computacional, el problema de la mochila es un problema difícil. No se ha encontrado un algoritmo de tiempo polinomial para el, y es muy poco probable que tal algoritmo exista. Se han propuesto varias técnicas como métodos de solución para éstos problemas como las siguientes:

1. Métodos de Reducción y Aproximación (del tipo gradiente y Lagrangeano).

2. Métodos exactos
   a. Redes
   b. Programación dinámica
   c. Métodos de Enumeración (Ramificación y Acotamiento)

Los métodos del tipo gradiente generalizado son eficientes computacionalmente sólo cuando se requieren soluciones aproximadas.

Los algoritmos de programación dinámica son eficientes cuando el valor de W es pequeño, cuando éste crece tiende a ser ineficiente porque se requiere un almacenamiento excesivo, mientras que las técnicas de enumeración resultan menos afectadas por ésta desventaja. El caso de la solución a través de redes formulan el problema de la mochila como un problema de ruta más corta, resultan también ineficientes porque se producen redes de tamaño muy grande

En éste trabajo sólo analizaremos el problema a través de el método de ramificación y acotamiento como veremos a continuación.

7.4 METODO DE RAMIFICACION Y ACOTAMIENTO

El primer método de ramificación y acotamiento que se usó para resolver el problema de la mochila es debido a Peter J. Kolesar (1967), éste método usa la estrategia de búsqueda por primer amplitud; el gran uso de memoria de computadora y los requerimientos de tiempo de éste algoritmo se redujeron mucho con el método de Greenberg y Hegerich (1970) que usa la estrategia de primer profundidad. Posteriormente Horowitz y Sahni(1974) propusieron un procedimiento de ramificación y acotamiento altamente eficiente, basado en el esquema de Greenberg-Hegerich, este mismo algoritmo se obtuvo independientemente por Ahrens y Finke (1975), y se han presentado más perfeccionamientos por Barr y Ross (1975), Fayard y Plateau (1975), Zoltners (1978), Nauss (1976), Suhl (1977), Martello y Toth (1977).

ALGORITMO DE KOLESAR

El algoritmo que se propone es un algoritmo de ramificación y acotamiento en el sentido de Little, et al.

Este método considera que un nodo lleva un índice i si el articulo se incluye y un índice i* si no se incluye. Un nodo con índice (i,j) significa que se incluye el artículo i primero y después el artículo j, mientras que el índice (i*,j) significa que el artículo i no se incluye pero el j si como se muestra en la figura 7.6

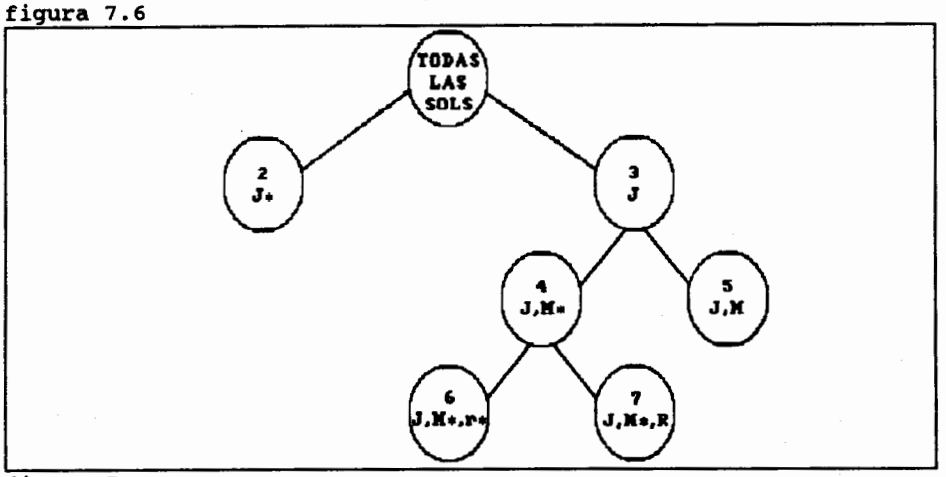


figura 7.6

Se denota por B(n) la cota superior asociada con el nodo n. Dado el nodo n tenemos tres categorías de artículos:

Los artículos incluidos In es el conjunto de artículos que están incluidos explícitamente en las soluciones contenidas en el nodo n; los artículos excluidos En como su nombre lo dice están explícitamente excluidos de las soluciones contenidas en el nodo n y finalmente los artículos no asignados son aquellos que no están incluidos en ninguna de las dos categorías anteriores, cuando el conjunto de artículos no asignados en un nodo es vacío, el nodo contiene una solución y ya no es posible ramificarlo más. En el mismo árbol de la figura 7.6, el nodo 1 representa la clase de todas las soluciones factibles, el nodo 2 representa todas las soluciones factibles que no incluyen al artículo j, mientras que el nodo 3 representa a las que si incluyen al artículo j, el nodo 7 representa a todas las que incluyen a j y r pero no a m. Los conjuntos In, En para todos los nodos de la figura 7.6 son:

$I_1 = \phi$	$E_1 = \phi$
$I_2 = \phi$	$E_2 = (j)$
$I_3 = (j)$	$E_3 = \phi$
$I_4 = (j)$	$E_4 = (m)$
$I_5 = (j,m)$	$E_5 = (m)$
$I_6 = (j)$	$E_6 = (m,r)$
$I_7 = (j,r)$	$E_7 = (m)$

El cálculo de cotas superiores se basa en dos observaciones. La primera es que la relajación de la restricción de indivisibilidad en uno o más de los artículos conlleva a un nuevo problema de optimización cuyo valor óptimo de la función objetivo no puede ser menor que el valor óptimo de la función objetivo para el problema original. La segunda observación es que para un problema de cargo en que todos los artículos son perfectamente divisibles, esto es :

$$\text{Maximizar} \quad \sum_{i=1}^{m} p_i y_i \tag{7.7}$$

Sujeto a

$$\sum_{i=1}^{m} y_i w_i \leq W \tag{7.8}$$

$$0 \leq y_i \leq 1 \tag{7.9}$$

Se obtiene una solución óptima arreglando los artículos en orden decreciente de magnitud de la razón p_i/w_i, y entonces, se comienza con el primer artículo de la lista, secuencialmente se va cargando cada artículo, hasta que se alcanza un peso límite W.

Aplicamos ahora éstos dos hechos al cálculo de cotas superiores. Supongamos que estamos en el nodo n, con los conjuntos In y En como se definieron anteriormente, antes de calcular una cota superior en el nodo n, probamos la factibilidad de la clase de soluciones contenidas en el nodo verificando si

$$\sum_{i \in In} w_i \leq W \tag{7.10}$$

Los nodos que no satisfagan (7.10) no se incluyen en los cálculos. Para calcular la cota superior en el nodo n, decimos que es suficiente con resolver el siguiente problema:

$$\text{Maximizar} \quad \sum_{i=1}^{a} p_i x_i$$

Sujeto a

$$\sum_{i=1}^{a} x_i w_i \leq W$$

$$\begin{array}{lll} x_i = 1 & \text{si} & i \in I_a \\ x_i = 0 & \text{si} & i \in E_a \\ 0 \leq x_i \leq 1 & \text{si} & i \in (I_a \cup E_a) \end{array} \qquad (7.11)$$

La solución de éste sistema es una cota superior en el nodo n aplicando la primer observación, ya que en (7.11) hemos relajado la restricción para los artículos no asignados en el nodo n, la segunda observación da una solución directa a éste sistema ya que por una simple transformación se puede poner en la forma (7.7),(7.8),(7.9). La solución está dada por:

a. Todos los artículos en $(I_a \cup E_a)'$ se ordenan decrecientemente p_i/w_i y se comienza con el primer artículo en la lista y se continua hasta que el peso total de los artículos sea igual a $W - \sum_{i \in I_a} w_i$.

b. La cota superior en el nodo n, B(n) es igual al valor total cargado en a. **más** $\sum_{i \in I_a} p_i$.

Para desarrollar la operación de ramificación, se deben tomar dos decisiones. Primero se debe seleccionar el nodo terminal en el que se hace la siguiente ramificación, y segunda, se debe seleccionar el artículo que se deberá sumar a la lista de artículos incluidos y excluidos en los dos nodos resultantes, llamamos a éste al artículo "pivote". La selección del nodo a ramificar está dada por el método de ramificación y acotamiento, de asignar el nodo terminal al que tenga la mayor cota B(n). Por otro lado la selección del elemento pivote es arbitraria, se desea que la selección sea de tal forma que el algoritmo alcance una solución óptima más rápidamente, resolviendo (7.7),(7.8),(7.9) se tiene una buena selección heurística. Suponiendo que los artículos originales están arreglados en orden decreciente de la razón p_i/w_i, seleccionamos el elemento pivote como la variable no asignada con el mayor p_i/w_i. En caso de haber empate se selecciona el primer artículo que aparece en la secuencia.

ALGORITMO 7.2: Kolesar

PROPOSITO: Resolver el problema tipo mochila 0-1 usando ramificación y acotamiento.

DESCRIPCION

Paso 1

1.1 Pruebe la factibilidad no trivial del problema verificando al menos uno de los índices i=1,2...,n donde $w_i \leq W$ si el problema es factible no trivial, siga a 1.2 si no, pare.

1.2 Si $\Sigma w_i \leq W$, entonces se pueden vaciar todos los artículos y el problema es trivial, si no vaya a 1.3

1.3 Ordene los artículos en orden decreciente de p_i/w_i, en lo que sigue se supondrá que los artículos están ordenados, vaya a 1.4

1.4 Para el nodo 1 haga B(1)=0, $I_i=\phi$, $E_i = \phi$ y vaya al paso 2

Paso 2 Selección del nodo para ramificar

2.1 Encuentre el nodo terminal con el mayor valor de B(n). Este es el nodo desde el cual se va a ramificar.

2.2 Pruebe si el nodo actual k, $(I_k \cup E_k)' = \phi$ si es cierto, hay una solución óptima dada por los índices contenidos en I_k. Si no seleccione el nuevo artículo pivote i^* por $i^* = i$ tal que p_i/w_i es máximo para $i \in (I_k \cup E_k)'$, y vaya al paso 3.

Paso 3 Cálculo de las cotas superiores.

3.1 Si n = n+1, $I_n = I_k$, $E_n = E_k \cup (i^*)$ vaya a 3.3

3.2 Si n = n+1, $I_n = I_k \cup (i^*)$, $E_n = E_k$ vaya a 3.3

3.3 Pruebe la factibilidad de las soluciones contenidas en el nodo n verificando si:

$$\sum_{k \in n} w_i \leq W,$$

si la prueba falla haga B(n) = -999, de otra forma proceda a calcular la cota superior metiendo todos los artículos en el conjunto In y procediendo entonces en forma ordenada i=1,2,...,n,cargando tantos artículos como sea posible del conjunto $(I_n \cup E_n)'$ hasta que el peso total cargado sea igual a W. El valor total es B(n). Pruebe si el nodo con índice n es par. Si lo es proceda con 3.2 si no vaya al Paso 2.

Ejemplo 7.3

Considere el siguiente problema tipo mochila con siete artículos cuyos pesos y valores se dan a continuación:

ARTICULO No.	PESO	VALOR
1	40	40
2	50	60
3	30	10
4	10	10
5	10	3
6	40	20
7	30	60

El peso total permitido es W = 100. Se desea maximizar el valor de los artículos que se van a llevar en la mochila.
Una prueba preliminar revela que el problema posee una solución factible y no es trivial, $\Sigma w_i > 100$. Calculamos las razones p_i/w_i y reordenamos los artículos, como se muestra a continuación con un nuevo índice:

NUEVO INDICE	ARTICULO No.	PESO	VALOR	RAZON
1	7	30	60	2
2	2	50	60	6/5
3	1	40	40	1
4	4	10	10	1
5	6	40	20	1/2
6	3	30	10	1/3
7	5	10	3	3/10

El primer nodo es el que incluye todas las soluciones posibles, y la primer ramificación usa el índice 1 como primer pivote, y en el nodo 2 se excluye éste índice de la solución, y la cota superior es igual a :

$$B(2) = p_2 + p_3 + p_4 = 110$$

Mientras en el nodo 3 donde éste índice se incluye tenemos:

$$B(3) = p_1 + p_2 + \tfrac{1}{2} p_3 = 140$$

A continuación se desarrollan los cálculos para cada nodo del árbol: cuando sobra peso se resta el peso sobrante, y en la columna del valor se resta el mismo peso multiplicado por la razón de valor/peso.

B(2) no incluye el artículo = 1°

índice	peso	valor
2	50	60
3	40	40
4	10	10
total	100	110

B(3) incluye el artículo =1

índice	peso	valor
1	30	60
2	50	60
3	40	40
total	120	160
	-20	-20
	100	140

la mejor cota es de 140 por lo que se procede a ramificar desde ahí:

B(4) incluye (1,2°)

índice	peso	valor
1	30	60
3	40	40
4	10	10
5	40	20
total	120	130
	-20	-10
	100	120

B(5) incluye (1,2)

índice	peso	valor
1	30	60
2	50	60
3	40	40
total	120	160
	-20	-20
	100	140

la mejor cota es 140 por lo que se procede a ramificar desde ahí:

B(6) incluye (1,2,3°)

índice	peso	valor
1	30	60
2	50	60
4	10	10
5	40	20
total	130	150
	-30	-15
	100	135

B(7) incluye (1,2,3)

índice	peso	valor
1	30	60
2	50	60
3	40	40
total	120	160

NO FACTIBLE

B(8) incluye (1,2,3°,4)

índice	peso	valor
1	30	60
2	50	60
4	10	10
5	40	20
total	130	150
	-30	-15
	100	135

B(9) incluye (1,2,3°,4°)

índice	peso	valor
1	30	60
2	50	60
5	40	20
total	120	140
	-20	-10
	100	130

B(10) incluye (1,2,3*,4,5)

índice	peso	valor
1	30	60
2	50	60
4	10	10
5	40	20
total	130	150

NO FACTIBLE

B(11) incluye (1,2,3*,4,5*)

índice	peso	valor
1	30	60
2	50	60
5	40	20
total	120	140
	-20	- 6.7
	100	133.3

B(12) incluye (1,2,3*,4,5*,6)

índice	peso	valor
1	30	60
2	50	60
4	10	10
6	30	10
total	120	140

NO FACTIBLE

B(13) incluye (1,2,3*,4,5*,6*)

índice	peso	valor
1	30	60
2	50	60
4	10	10
7	10	3
	100	133

B(14) incluye (1,2,3*,4,5*,6*,7*)

índice	peso	valor
1	30	60
2	50	60
4	10	10
total	90	130

B(15) incluye (1,2,3*,4,5*,6*,7)

índice	peso	valor
1	30	60
2	50	60
4	10	10
7	10	3
	100	133

De donde se observa que el óptimo se obtiene en el nodo (15). El valor total es 133 llevando los artículos 1, 2, 4 y 7, y el árbol se ilustra en la figura 7.7

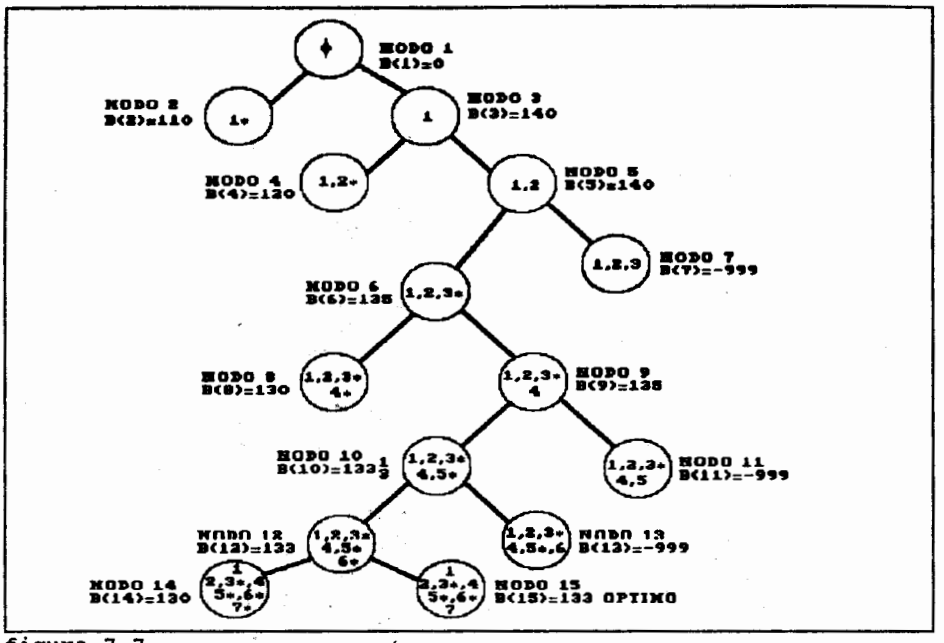


figura 7.7

7.5 NOTAS HISTORICAS

Se tiene noticia de que Euler (1759) y Vandermonde (1771) discutieron el problema del recorrido de un caballo. Este problema es un circuito hamiltoniano en una gráfica cuyos vértices son los 64 cuadros de un ajedrez, con dos vértices adyacentes si y sólo si el caballo se puede mover en un paso de un cuadrado al otro.

El Reverendo T.P. Kirkman (1856) fue probablemente el primero en considerar circuitos hamiltonianos en un contexto mas general. Sin embargo, al mismo tiempo Hamilton (1856) trabajó en el asunto e inventó un sistema de álgebra no conmutativa, para la cual, las acciones de los elementos en la base se pueden interpretar en términos de trayectorias en un dodecaedro regular. Hamilton llemó a ésta álgebra Cálculo Icosiano (los vértices adyacentes en el dodecaedro corresponden a caras adyacentes en un icosaedro) y usó la interpretación gráfica como base para un juego que se vendió con el nombre de "El juego del icosaedro" como se muestra en la figura 7.8. El juego consistía en varios problemas, tales como, terminar un circuito cuando se nos dan las primeras cinco posiciones.

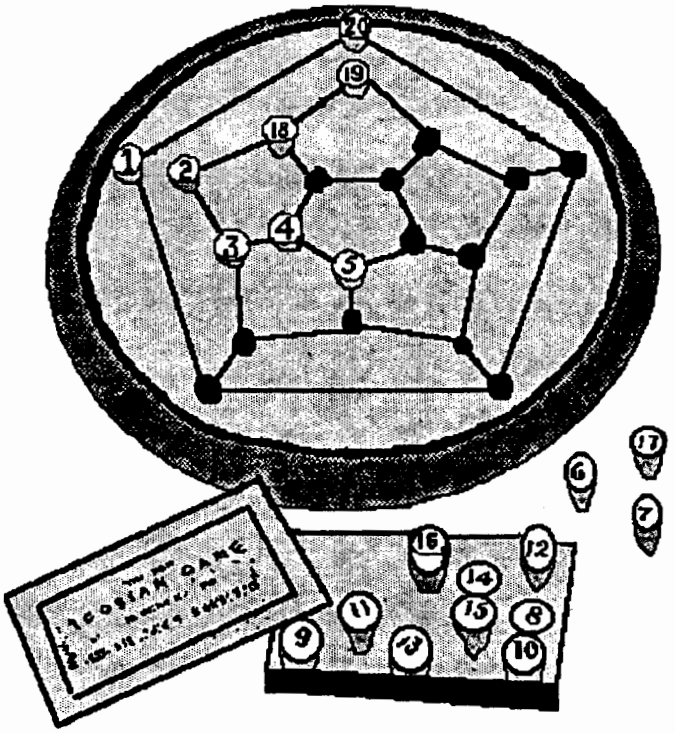


FIGURA 7.8

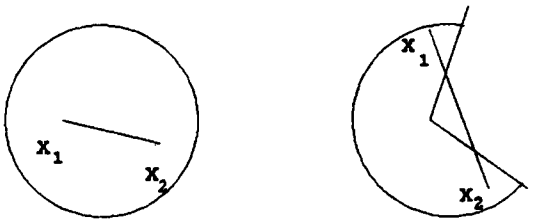
APENDICE A

CONJUNTOS CONVEXOS

A.1 CONJUNTOS CONVEXOS

Definición A.1 Un conjunto C en R^n es convexo para todas x_1, $x_2 \epsilon C$ y todo número real α, $0 < \alpha < 1$ si el punto $\alpha x_1 + (1-\alpha)x_2 \epsilon C$.

Esta definición se puede interpretar geométricamente en el sentido, de que un conjunto es convexo, si dados dos puntos de un conjunto, todo punto del segmento de recta que une estos dos puntos es también un miembro del conjunto, como se muestra en la figura A.



CONVEXO NO CONVEXO

Figura A.1

Ejemplo 3.1 Considere el siguiente conjunto:

$$\begin{aligned} -2x_1 + x_2 &\leq 4 \\ x_1 + x_2 &\leq 3 \\ x_1 &\leq 2 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

La intersección de estos 5 semiespacios es el conjunto sombreado, como se muestra en la figura A.2, que es convexo.

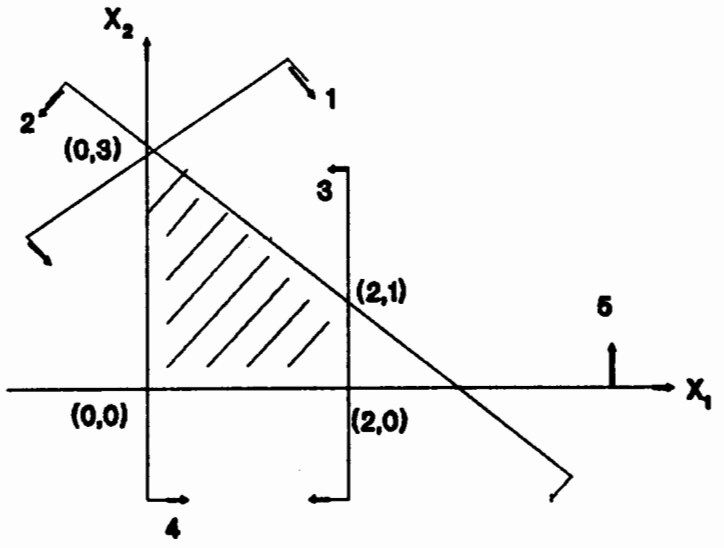


Figura A.2

A.2 HIPERPLANOS Y POLITOPOS

El tipo más importante de conjunto convexo (además de los puntos aislados es el hiperplano. Los hiperplanos predominan en toda la teoría de optimación presentándose en la forma de multiplicadores de Lagrange, teoría de dualidad o cálculos del gradiente.

La definición más natural de un hiperplano es la generalización lógica de propiedades geométricas de un plano en tres dimensiones. Se comienza dando esta definición geométrica. Sin embargo, para los cálculos y para una descripción concreta de los hiperplanos hay una definición algebraica equivalente más útil. Una parte importante de esta sección se dedica a probar esta equivalencia.

Definición A.2 Un conjunto V de E^n se dice que es una *variedad lineal*, si dados cualesquiera x_1, $x_2 \epsilon V$, se tiene $\lambda x_1 + (1-\lambda)x_2 \epsilon V$ para todos los números reales λ.

Obsérvese que la única diferencia entre la definición de una variedad lineal y un conjunto convexo es que en una variedad lineal toda la recta que pasa por dos puntos cualesquiera, en lugar de sólo el segmento de recta entre ellos debe estar en el conjunto. Así, en tres dimensiones, las variedades lineales no vacías son puntos, rectas, planos bidimensionales y todo el espacio. En general, es evidente que se puede hablar de la dimensión de una variedad lineal. Así, por ejemplo, un punto es una variedad lineal de dimensión uno. En el caso general, se puede hallar la dimensión de una variedad lineal de E^n trasladándola (moviendola) de modo que contenga al origen y, después determinando la dimensión del conjunto resultante, que es entonces un subespacio de E^n.

Definición A.3 Un *hiperplano* de E_n es una variedad lineal (n-1)dimensional.

Se observa que los hiperplanos generalizan el concepto de un plano bidimensional en el espacio tridimensional. Se pueden considerar como las variedades lineales más grandes en un espacio que no sea todo el espacio.

Ahora, se relaciona esta definición geométrica abstracta con una algebraica.

Proposición A.1. Sea $\mathbf{a}$ un vector columna de n dimensiones distinto de cero, y sea c un número real. El conjunto

$$H = \{\mathbf{x} \epsilon E^n : \mathbf{a}^T\mathbf{x} = c\}$$

es un hiperplano de E^n.

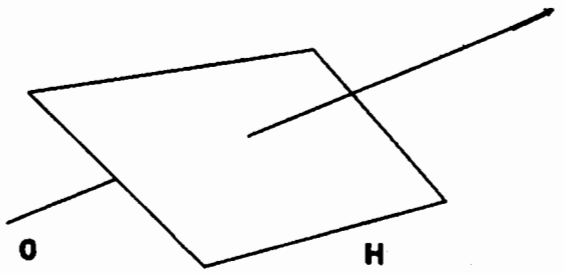
Demostración. Se obtiene directamente de la linealidad de la ecuación $\mathbf{a}^T\mathbf{x}=c$ que H es una variedad lineal. Sea $\mathbf{x}_1$ cualquier vector en H. Trasladando por $-\mathbf{x}_1$ se obtiene el conjunto $M = H - \mathbf{x}_1$, que es un subespacio lineal de E_n. Este subespacio está formado por todos los vectores x que satisfacen $\mathbf{a}^T\mathbf{x}=0$; en otras palabras, todos los vectores ortogonales a $\mathbf{a}$. Evidentemente, esto es un subespacio (n-1)-dimensional

Proposición A.2 Sea H un hiperplano de E^n. Entonces, existen un vector distinto de cero n-dimensional y una constante c tal que

$$H = \{ \mathbf{x} \epsilon E^N : \mathbf{a}^T\mathbf{x} = c\}$$

Demostración. Sea $\mathbf{x}_1 \epsilon H$ y trasládese por $-\mathbf{x}_1$, obteniendo el conjunto $M=H-\mathbf{x}_1$. Como H es un hiperplano, M es un subespacio (n-1)-dimensional-. Sea $\mathbf{a}$ cualquier vector distinto de cero ortogonal a este subespacio, es decir, $\mathbf{a}$ pertenece al subespacio unidimensional $M^{\perp}$. Es evidente que $M=\{\mathbf{x};\mathbf{a}^T\mathbf{x}=0\}$. Al hacer $c=\mathbf{a}^T\mathbf{x}_1$, se observa que si $\mathbf{x}_2 \epsilon M$ y así, $\mathbf{a}^T\mathbf{x}_2-\mathbf{a}^T\mathbf{x}_1=0$, lo cual implica $\mathbf{a}_T\mathbf{x}_2=c$. Así, $H \subset \{\mathbf{x}:\mathbf{a}=c\}$. Como H es, por definición, de dimensión n-1 y $\{\mathbf{x}:\mathbf{a}^T\mathbf{x}=c\}$ es de dimensión n-1 por la proposición 2, estos dos conjuntos deben ser iguales. ▌

Al combinar las proposiciones A.1 y A.2 se observa que un hiperplano es el conjunto de soluciones a una sola ecuación lineal. Esto se ilustra en la figura A.3. Ahora, se utilizan hiperplanos para elaborar otra clase importante de conjuntos convexos.



Definición A.4 Sea $\mathbf{a}$ un vector distinto de cero en E^n y sea c un número real correspondientes al hiperplano $H = \{\mathbf{x}:\mathbf{a}^T\mathbf{x}=c\}$ están los *semiespacios cerrados positivo y negativo*

$$H_+ = \{\mathbf{x}:\mathbf{a}^T\mathbf{x} \geq c\}$$

$$H_- = \{\mathbf{x}:\mathbf{a}^T\mathbf{x} \leq c\}$$

y los ***semiespacios abiertos positivo y negativo***

$$H_+ = \{x : a^T x > c\}$$

$$H_- = \{x : a^T x < c\}$$

Es fácil observar que los semiespacios son conjuntos convexos y que la unión de H_+ y H_- es espacio total.

Definición A.5 Un conjunto que puede expresarse como la intersección de un número finito de semiespacios cerrados es un *polítopo convexo*.

Se observa que los polítopos convexos son los conjuntos obtenidos como la familia de soluciones a un conjunto de desigualdades lineales de la forma

$$\begin{aligned} a^T_1 x &\leq b_1 \\ a^T_2 x &\leq b_2 \\ &\vdots \\ a^T_m x &\leq b_m, \end{aligned}$$

pues cada desigualdad individual define un semiespacio y la familia solución es la intersección de estos semiespacios. (Si alguna $a_i=0$, el resultado todavía se puede expresar, como puede comprobar el lector, como la intersección de un número finito de semiespacios).

En la figura A.4 se representan varios polítopos. Se observa que un polítopo puede ser vacío, acotado o no acotado. Es de especial interés el caso de un polítopo acotado no vacío y se distinguirá este caso por la siguiente

Definición A.6 Un polítopo acotado no vacío se denomina poliedro.

APENDICE B

METODO DUAL SIMPLEX

Paso 1. Empiece con un tableau donde todas las $z_j - c_j \geq 0$ para toda $j \in A$.

Paso 2. Si $x_{Bi} \geq 0 \; \forall \; i=i,\ldots,m$, el tableau actual es óptimo si no, seleccionese como vector de salida de la base, aquél cuyo correspondiente x_{Bi} sea el mas negativo.

Paso 3. El vector a_k de entrada será aquel que satisfaga la siguiente regla

$$\frac{z_k - c_k}{y_{ik}} = \max\left\{\frac{z_j - c_j}{y_{ij}} \mid y_{ij} < 0\right\}$$

Paso 4. La columna a_k se convierte en el vector unitario k_i con el pivote y_{ik} igual a uno. Los cambios se llevan a cabo en operaciones matriciales elementales. Regrese al paso 2 y repita hasta que las condiciones de optimalidad se cumplan.

Si durante la aplicación del paso 3 todos los elementos $y_{ik} \geq 0$ el problema original no tiene solución.

Ejemplo B.1

Resuelva usando el Método Dual Simplex el siguiente problema de programación lineal

$$\min z = 2x_1 + x_2$$

s.a

$$3x_1 + x_2 \geq 3$$

$$4x_1 + 2x_2 \geq 6$$

$$x_1 + 2x_2 \geq 3$$

$$x_1 \geq 0 \quad x_2 \geq 0$$

este programa se reescribe como:

$$\max h = -z = -2x_1 - x_2$$

s.a

$$-3x_1 - x_2 + x_3 = -3$$

$$-4x_1 - 3x_2 + x_4 = -6$$

$$-x_1 - 2x_2 + x_5 = -3$$

$$x_1 \geq 0$$

El primer tableau es:

	h	x_1	x_2	x_3	x_4	x_5	LD
$z_j - c_j$	1	2	1	0	0	0	0
a_3	0	-3	-1	1	0	0	-3
a_4	0	-4	-3	0	1	0	-6 ←
a_5	0	-1	-2	0	0	1	-3

Note que existe factibilidad dual pues todas las $z_j - c_j \geq 0$ para toda $j \epsilon A$. La solución actual no es óptima pues $x_B \geq 0$. Se selecciona al vector a_4 como el vector que sale, pues $x_{B4} = -6$ es el mas negativo. Para seleccionar al vector que entra se usa la regla:

$$\frac{z_k - c_k}{x_{B4k}} = \max \left\{ \frac{z_j - c_j}{x_{B4j}} \mid x_{By,j} < 0 \right\}$$

$$= \max \left\{ \frac{2}{-4}, \frac{1}{-3} \right\}$$

$$= -\frac{1}{3}$$

y por lo tanto a_2 entra a la base. El pivote es el elemento $Y_{B4,2}$ o sea el $X_{2,2}$. El vector a_2 se convierte en la columna unitaria con un uno en la posición del pivote. Esto se logra con una operación que arroja el siguiente tableau.

		x_1	x_2	x_3	x_4	x_5	
	1	2/3	0	0	1/3	0	-2
a_3	0	-5/3	0	1	-1/3	0	-1 ←
a_2	0	4/3	1	0	-1/3	0	2
a_5	0	5/3	0	0	-2/3	1	1

Esta solución no es óptima ya que $x_{B1} = -1 < 0$. El vector a_3 sale de la base y la regla de entrada indica:

$$\max \left\{ \frac{2/3}{-5/3} , \frac{1/3}{-1/3} \right\} = - \frac{2}{5}$$

o sea que a_1 entra en la base. Haciendo el pivote $Y_{B1,1}(Y_{11})$ igual a 1 y el resto de la columna a_1 igual a cero, y se obtiene:

	h	x_1	x_2	x_3	x_4	x_5	
	1	0	0	2/5	1/5	0	-12/5
a_1	0	1	0	-3/5	1/5	0	3/5
a_2	0	0	1	4/5	-3/5	0	6/5
a_5	0	0	0	1	-1	1	0

La solución óptima es $x = (3/5, 6/5, 0,0,0)$

y $h = - z = - 12/5 \rightarrow z = 12/5$

La ventaja de este mÉtodo es que no requiere variables artificiales, requiere menos iteraciones (que el de dos fases y de penalización). La desventaja es que para empezar a iterar se requiere la condición de factibilidad dual, es decir $z_j - c_j \geq 0$ para toda $j \in A$.

16. LAND, A.H. Y DOIG, A.G. "*An Automatic Method of Solving Discrete Programming Problems*", Econométrica 28, 497-520, 1960.

17. LARNDER, H "*The Origin of Operational Research*".Opns Res.32, p.465-475, 1984.

18. LAWLER, E.L Y WOOD, D.E "*Branch-and-Bound Methods: A Survey*" Opns. Res.14, p.669-719, 1966.

19. LAWLER, E.L., LENSTRA, J.K. RINNOOY KAN, A.H.G. Y SHMOYS D.B. " *The Traveling Salesman Problem*" Ed. Wiley, 1985.

20. LITTLE, J.,MURTY, K., SWEENEY, D. Y KAREL, C. "*An Algorithm for the Traveling Salesman Problem*", Opns. Res. 34, p.698-717, 1963.

21. MARTELLO, S. Y TOTH, P. "*Algorithm for the Solution of 0-1 Single Knapsack Problem*", Computing 21, p.81-86, 1978.

22. MITTEN, L.G. "*Branch and Bound Methods: General Formulation and Properties*" Opns. Res. p.24-35, 1969.

23. MORSE, P.M "*The Beginnings of Operations Research in the United States*" Opns Res.34, 10-17, 1986.

24. NEMHAUSER, G Y WOLSEY L. "*Integer and Combinatorial Optimization*" Wiley Ed. 1988.

25. OCHOA-ROSSO F. "*Applications of Discrete Optimization Techniques to Capital Investment and Network Synthesis Problems*" Research Report R68-42 Massachusetts Institute of Technology, Cambridge 1968.

26. PARKER, G. "*Discrete Optimization*" Academic Press 1988.

27. PRAWDA, J. "*Metodos y Modelos de Investigación de Operaciones*" Limusa, 1979.

28. SALKIN, H.M. "*Integer Programming*", Addison-Wesley, 1974.

29. SYSLO, M. "*Discrete Optimization Algorithms*" Ed.

30. TAHA, H.A. "*Integer Programming Theory, Applications and Computations*", Academic Press, 1975.

31. THESEN, A "*Computer Methods in O.R.*" Capítulo 8, Academic Press, 1978.

32. ZIONTS, S."*On An Algorithm for the Solution of Mixed Integer Programming Problems*",Man. Sci.15, 113 -116.

BIBLIOGRAFIA

1. AGIN, N. "*Optimum Seeking with Branch and Bound*", Management Science 13, p B-176-185, 1966.

2. AHO, HOPCROFT & ULLMAN " *Estructura de Datos y Algoritmos*" Ed Sitesa, México 1988.

3. BALAS, E "*An Additive Algorithm for Solving Linear Programs with Zero-One Variables*" Operations Research 13, p.517-546, 1965.

4. BALAS, E. "*A Note on the Branch and Bound Principle*" Operations Research 16, p 442-445, 1968.

5. BALINSKI, M.L. "*Integer Programming: Methods, Uses, Computation*", Man.Sci. 12, No.3 p. 253-313, 1965.

6. BOFFEY, T.B "*Graph Theory in Operations Research*" Capítulo 3, Mac Millan 1982.

7. CHRISTOFIDES, N. "*Combinatorial Optimization*", Ed. John Wiley, 1979.

8. DAKIN, R.J. "*A Tree-Search Algorithm for Mixed Integer Programming Problems*", The Computer Journal 8, p.250-255, 1965.

9. DRIEBEEK, N.J. "*An Algorithm for the Solution of Mixed Integer Programming Problems*", Man.Sci.12, No. 7, 576-587, 1966.

10. FUENTES MAYA, S "*Programación Entera*" DEPFI, 1985

11. GEOFFRION, A.M. y MARSTEN, R. E. "*Integer Programming Algorithms: A Framework and State-of-the Art Survey*", Management Science 18, no 9 p.465-491, 1972.

12. GONDRAN, M. Y MINOUX, M. "*Graphs and Algorithms*", Ed. John Wiley, 1979.

13. HOROWITZ, E Y SAHNI, S "*Fundamentals of Computer Algorithms*" Capítulo 8, Computer Science Press, 1984.

14. JIANYU, Y. "*El Problema del Viajero y sus Extensiones*",Tesis de Maestría DEPFI-UNAM, 1985.

15. KOLESAR, P.J. "*A Branch and Bound Algorithm for the Knapsack Problem*", Management Science 13, p.723-735, 1967.