UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

SECCION DE MATEMATICAS

"TECNICAS DE RAMIFICACION Y ACOTAMIENTO PARA LA SOLUCION DE PROBLEMAS DE OPTIMIZACION DISCRETA"

TESIS PROFESIONAL

QUE PARA OBTENER EL TITULO DE:

A C T U A R I O

P R E S E N T A :

JOSE LUIS FLORES CAMACHO

CIUDAD UNIVERSITARIA ABRIL 1992

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

INTRODUCCION

En el mundo real, la búsqueda de la mejor solución, la máxima, la mínima o en general la óptima, para un amplio rango de problemas, ha intrigado al hombre a través de las épocas. Euclides, por ejemplo, describió métodos para encontrar las líneas rectas más largas y más cortas desde un punto, a la circunferencia de un círculo y cómo determinar el paralelogramo de área máxima con un perímetro dado. Los grandes matemáticos de los siglos XVI y XVIII desarrollaron nuevos procedimientos de optimización para resolver problemas geométricos, dinámicos y físicos, tales como encontrar las curvas de revolución mínima o la curva de descenso más rápido.

Recientemente se ha originado una nueva clase de problemas de optimización, debido a las complejas estructuras organizacionales que rigen la sociedad moderna. Un problema relacionado con esto es, que a medida que se incrementa la complejidad y especialización de una organización, se vuelve cada vez más difícil asignar los recursos disponibles a sus diversas actividades, de manera que sea

lo más efectivo para la organización. Y son problemas que tienen que ver desde el cómo administrar una economía o una fábrica, hasta el cómo distribuir de manera óptima el presupuesto, el personal y las actividades dentro de una organización.

La búsqueda de cómo formular y resolver éstos y muchos otros problemas que surgen en los campos de la agricultura, petroquímica, transporte, programación de producción, control de inventarios, ingeniería y economía entre otros, ha conducido al desarrollo de nuevas e importantes técnicas de optimización, dentro de las que está incluída la optimización discreta -tema de estudio de este trabajo- y que forma parte de una rama científica mayor, que es la investigación de operaciones. Esta se define como el procedimiento para tomar decisiones que tienen que ver con las operaciones de sistemas de organización. Sin embargo, esta descripción es tan grande que puede aplicarse, con igual propiedad, a muchos otros campos. Por lo tanto, tal vez la mejor manera de captar la naturaleza única de la investigación de operaciones es examinar sus características sobresalientes.

Así la investigación de operaciones se aplica a problemas que tienen que ver con la forma de conducir y coordinar las operaciones o actividades dentro de una organización. En particular, el proceso se inicia observando y formulando cuidadosamente el problema y a continuación, se construye un modelo (típicamente matemático) que intenta abstraer la parte esencial del problema real. Entonces se establece la hipótesis de que este modelo es una representación lo suficientemente precisa de las características más sobresalientes del problema, de modo que las conclusiones (soluciones) que se obtengan a partir del modelo son, así mismo, válidas para el problema real. Por tanto, en cierto sentido, la investigación de operaciones comprende una investigación de las propiedades fundamentales de las operaciones.

En resumen, la contribución del procedimiento de Investigación de Operaciones se basa principalmente en:

1.- La estructuración de la situación de la vida real en un modelo matemático, abstrayendo los elementos esenciales de modo que pueda descubrirse la solución pertinente para los objetivos de quienes toman las decisiones.

2.- La exploración de la estructura de tales soluciones y el desarrollo de los procedimientos sistemáticos para obtenerlas.

3.- La determinación de una solución, incluyendo la teoría matemática, si es necesario, que proporciona un valor óptimo de la medida de deseabilidad de un sistema (o posiblemente la comparación de cursos de acción alternativos, evaluando su medida de deseabilidad).

4.- Implantación de medidas de control, para regular los posibles cambios que más adelante puedan surgir en la estructura del problema mismo y evitar su completa reformulación.

Debido a esto, los procedimientos sistemáticos con que cuenta la investigación de operaciones para encontrar la solución a problemas reales, son muchos y muy variados, lo que ha dado lugar al desarrollo de varias ramas de esta ciencia, de la cuales podemos citar a la programación lineal, programación entera, programación dinámica, redes, procesos estocásticos, etc..

En el presente trabajo se hace el estudio de una de las ramas de la investigación de operaciones, esto es, la optimización discreta (básicamente la programación cero-uno). Y es aquí donde presentaremos los tipos de problemas que pueden ser modelados por esta técnica y la forma de obtener la solución al mismo vía la

técnica de Ramificación y Acotamiento (técnica más comúnmente usada en esta área).

En este contexto, **los propósitos de esta tesis son:**

1°.- Ser útil y comprensible para aquéllos que tengan necesidades de plantear y resolver problemas que pueden ser modelados matemáticamente, en donde es necesario maximizar o minimizar una función objetivo sobre un conjunto finito de posibles alternativas.

2°.- Que el lector logre valorar la capacidad con la que los modelos de programación discreta tienden a reflejar la situación real de este tipo de problemas.

3°.- Buscar la tan acostumbrada eficiencia en la utilización de estos modelos, proporcionando algunos elementos (análisis de postoptimalidad y algoritmos especializados) que ayuden a obtener rapidamente una solución, al reducir el tiempo de proceso de los algoritmos, cada vez que se plantea un problema de optimización discreta.

Para ello el trabajo es desarrollado como sigue: En el primer capítulo -**Problemas de Optimización Discreta**- damos paso a la presentación del modelado matemático, mostrando las herramientas y artificios más comúnmente utilizados, cada vez que necesitamos modelar matemáticamente un problema discreto, tratando siempre de no perder el reflejo de todas las situaciones reales del problema; adicional al capítulo, enunciamos las técnicas de solución más comúnmente utilizadas para este modelo. En el segundo capítulo -**La técnica de Ramificación y Acotamiento**- se especifican las ideas generales, en cuanto al funcionamiento de dicha técnica, cada vez que es resuelto un problema discreto, mostrando con ello las reglas

más comunes de ramificación (enumeración de las posibles soluciones), así como algunas reglas de acotamiento al valor de la función objetivo solución enumerada. En el tercer capítulo -**Algoritmos Específicos**- ya se presentan de manera formal los algoritmos de ramificación y acotamiento, utilizados en la solución de una clase particular de problemas de optimización discreta (Problemas de programación cero-uno) y con el propósito de eficientar dichos algoritmos, se anexa una parte de análisis de postoptimalidad, así como los algoritmos para problemas más particulares (asignación y secuenciación). Y finalmente, en el cuarto capítulo -**Modelos Clásicos**- simplemente se enuncian una variedad de problemas, que se formulan y resuelven mediante la técnica ya mostrada, los cuales involucran un conjunto de situaciones que muy a menudo son las que se presentan en la práctica profesional.

1. EL PROBLEMA DE OPTIMIZACION DISCRETA

1.1. INTRODUCCION

En la práctica, existen muchos problemas en los que es muy común encontrar que la solución óptima a éstos, tiene sentido sólo para valores enteros. Por ejemplo es necesario con frecuencia, asignar hombres, máquinas y vehículos a las diversas actividades en cantidades enteras. Más aún, existen problemas, en que el conjunto de todos los posibles candidatos a la solución óptima, són tales que presentan alternativas de decisión "si-no". Por ejemplo, en un problema de contratación de personal, se pretende seleccionar de un conjunto de aspirantes a aquellos que satisfagan las necesidades de la empresa, pero cumpliendo también con ciertas condiciones como minimizar costos, tiempo, etc. y decidir para cada aspirante, si éste será o no contratado. Equivalentemente, un problema restringido a los valores enteros 0 y 1 donde 0=no se contrata y 1=se contrata.

Este tipo de restricciones a los problemas es difícil de manejarse en forma matemática. Sin embargo se han hecho algunos logros en cuanto al desarrollo de los modelos para representar el problema y con los procedimientos para obtener su solución. En este capítulo pretendemos describir los logros en cuanto a la representación de esta clase de problemas. Para ello presentaremos dos problemas que a menudo surgen en la práctica

1) **Un Problema de proyectos de inversión** que permite ver los diferentes tipos de restricción que se presentan en cuanto a su planteamiento e interpretación se refiere .

2) **El Problema del vendedor viajero** que debe visitar n-ciudades (exactamente una vez cada una), y en donde podemos apreciar que no siempre existe una manera única de definir las variables de decisión, logrando en este caso el planteamiento del mismo problema en dos modelos matemáticos distintos. Tratando de obtener siempre un modelo que involucre un número pequeño de variables y restricciónes.

Adicionalmente presentamos el modelo matemático general cero-uno, sus propiedades básicas y algunas de las formas que existen para encontrar la solución óptima, señalando en algunos casos, aquellas que resultan ser adecuadas, debido a que logran una buena eficiencia ya sea de manera general o para ciertos casos particulares.

EJEMPLO 1.2.1. (Un Problema de Proyectos de Inversión)

Supóngase que tenemos un conjunto de siete proyectos u oportunidades de inversión y cada uno de ellos tiene cierta utilidad estimada a largo plazo así como un costo o cantidad de capital requerido, según se muestra en la tabla .

	Oportunidad de Inversión						
	1	2	3	4	5	6	7
Utilidad Estimada	9	16	18	15	6	12	11
Capital Requerido	25	40	45	35	15	30	25

(millones de pesos)

Suponga que la cantidad total disponible para estas inversiones es de $ 100,000,000.00. Se desea determinar qué proyectos elegir para invertir, a fin de obtener la máxima utilidad total posible.

SOLUCION

Obsérvese que para resolver este problema, necesitamos saber qué proyectos elegir para invertir, es decir para cada proyecto decidir si éste debe o no estar dentro de los que serán elegidos, a fin de optimizar ganancias.

Este tipo de situaciones, en las que sólo se tienen dos elecciones (si-no), es representada por variables cuyos valores son 0 y 1. Así la i-ésima (i=1,...7) variable de decisión es representada por:

$$x_i = \begin{cases} 1 & \text{Si se invierte en el proyecto } i \\ 0 & \text{Si no se decide invertir en el proyecto } i \end{cases}$$

El objetivo es:

$$\textit{Maximizar}\quad Z = \text{total de utilidades obtenidas} = 9X_1 + 16X_2 + 18X_3 + 15X_4 + 6X_5 + 12X_6 + 11X_7$$

Pero respetando la restricción:

$$\text{total de capital invertido} \leq \text{total de capital disponible}$$

o bien

$$25X_1 + 40X_2 + 45X_3 + 35X_4 + 15X_5 + 30X_6 + 25X_7 \leq 100$$

Con lo que el modelo de programación cero-uno es:

$$\begin{cases} \textit{Max}\quad Z = 9X_1 + 16X_2 + 18X_3 + 15X_4 + 6X_5 + 12X_6 + 11X_7 \\ \text{sujeto a} \\ 25X_1 + 40X_2 + 45X_3 + 35X_4 + 15X_5 + 30X_6 + 25X_7 \leq 100 \\ X_i = 0,\ 1 \qquad i = 1,2,\ldots,7 \end{cases}$$

EJEMPLO 1.2.2. (Extensión Problema de Proyectos de Inversión)

Una extensión al problema de proyectos de inversión es presentada a continuación y refleja algo que en la práctica es muy común encontrar, a saber las reglas que rigen de alguna manera las formas de cómo puede proceder el inversionista. Para poder apreciar esto, veamos una extensión al problema anterior y supongamos que se presentan cuatro tipos de situaciónes adicionales:

a.- Se tiene que los proyectos 1 y 2 se desarrollan en dos etapas, teniendo la misma fecha de terminación para su primera etapa.

		Oportunidad de Inversión						
		1	2	3	4	5	6	7
1era. Etapa	Utilidad Estimada	9	16	18	15	6	12	11
	Capital requerido	25	40	45	35	15	30	25
2a. Etapa	Utilidad Estimada	12	15					
	Capital requerido	100	160					

Tanto las utilidades como el capital requerido para las etapas 1 y 2, están evaluadas al momento actual.

b.- Para los proyectos 1 y 2, en lo que corresponde a su primera etapa, sólo podrá elegirse uno de los dos.

c.- Se podrá invertir en la 2a. etapa de los proyectos 1 y 2 sólo si se optó por invertir en la primera etapa del respectivo proyecto.

d.- La fecha de terminación del 3er.y 4o. proyectos, ocurre antes de la fecha de terminación de la primera etapa de los proyectos 1 y 2. Con lo que podrá disponerse del capital resultante de dichas inversiones (esto en caso que se haya optado por invertir en dichos proyectos) para poder ser ocupado en la segunda etapa de inversión.

SOLUCION

Para el replanteamiento del problema bajo estas extensiones, primeramente necesitamos adicionar 2 variables que indicarán las decisiones tomadas sobre la segunda etapa de los proyectos 1 y 2 .

$$X_i^{*} = \begin{cases} 1 & \text{Si se decide invertir en el proyecto } i \text{ para su segunda etapa} \\ 0 & \text{en otro caso} \end{cases} \qquad i=1,2$$

Las variables originales X_i son las mismas, excepto, X_1 y X_2 (que son las decisiones tomadas, sobre los proyectos 1 y 2, en su primera etapa).

El objetivo es obtener la máxima utilidad estimada, es decir:

Maximizar

$$Z = 9X_1 + 12X_1' + 16X_2 + 15X_2' + 18X_3 + 15X_4 + 8X_5 + 12X_6 + 11X_7$$

pero respetando las restricciones:

a.- sobre el capital disponible a inicio de la primera etapa.

$$25X_1 + 40X_2 + 45X_3 + 35X_4 + 15X_5 + 30X_6 + 25X_7 \leq 100$$

b.- Debido a que al momento de inicio de la primera etapa tendrá que optarse por elegir uno y sólo uno de los proyectos 1 y 2, debe entonces suceder que

$$X_1 + X_2 = 1$$

Ya que de esta forma, las únicas posibles soluciones son (X_1=0, X_2=1) o (X_1=1, X_2=0), así qué cuando se elige uno de los dos, el otro queda automáticamente descartado.

En general, a este tipo de restricciones, se les denomina ALTERNATIVAS MUTUAMENTE EXCLUSIVAS, las cuales en el caso general se plantean como sigue:

$$\sum_i X_i = 1 \quad \text{(si exactamente una decisión en el grupo es sí)}$$

ó

$$\sum_i X_i \leq 1 \quad \text{(si cuando mucho una decisión en el grupo es sí)}$$

c.- Dado que se puede invertir en la segunda etapa de los proyectos 1 y 2 sólo cuando se decidió hacerlo en su primera etapa entonces debe suceder que

$$X_1' = X_1$$

y

$$X_2' = X_2$$

Ya que de esta forma, X_1' y X_1 toman siempre los mismos valores, así cuando el proyecto 1 se elige en lo que corresponde a su prmera etapa, también lo será en lo correspondiente a su segunda etapa, de otro modo no pueden elegirse ninguno de los dos. Lo mismo sucede con X_2' y X_2.

En general, a este tipo de restricciones se les conoce como DECISIONES CONTINGENTES, y su planteamiento en el modelo matemático es de la siguiente forma:

$X_k \leq X_j$ (El proyecto k será elegido sólo si el proyecto j también se elige)

$X_k = X_j$ (El proyecto k debe ser elegido si y sólo si también lo es el proyecto j)

En el primer caso, si decidimos elegir el j-ésimo elemento, entonces $X_j=1$ lo que permite elegir libremente a X_k, mientras que $X_j=0$ fuerza a $X_k=0$, es decir, si el j-ésimo elemento no fue elegido, entonces tampoco será elegido el k-ésimo. Normalmente estas restricciones se escriben como:

$$X_k - X_j \leq 0$$

y

$$X_k - X_j = 0$$

d.- Necesitamos también respetar las restricciones sobre el capital necesario para poder invertir, al momento de dar inicio la segunda etapa. Para esto, se observa que el capital disponible al inicio de esta etapa puede provenir básicamente de 3 formas:

Una de éstas proviene de los proyectos tres y cuatro, los cuales finalizan su ejecución antes de iniciar la segunda etapa de los proyectos 1 y 2. Con lo que en caso de invertir en los proyectos 3 y 4, obtendríamos ciertas utilidades producto de dichas inversiones, las que en total son 18 y 15 millones respectivamente, ocupando para ello 45 y 35 millones de capital. Así pues al finalizar estos proyectos, se estaría liberando, según esto, un total de

$$(18 + 45)X_3 + (15 + 35)X_4 \quad \text{millones de pesos}$$

La segunda fuente proviene del capital que se obtenga producto de haber invertido en cualesquiera de los proyectos 1 y 2, en su primera etapa.

$$(9 + 25)X_1 \quad \acute{o} \quad (16 + 40)X_2$$

esto dependiendo si se optó por invertir en el proyecto 1 o en el proyecto 2.

Y la tercer fuente proviene del capital sobrante, producto de no haber invertido completamente los 100 millones al inicio de la primera etapa. Es decir, dado que

$$25X_1 + 40X_2 + 45X_3 + 35X_4 + 15X_5 + 30X_6 + 25X_7 \leq 100$$

entonces existe cierto capital que le llamaremos h, con el que

$$25X_1 + 40X_2 + 45X_3 + 35X_4 + 15X_5 + 30X_6 + 25X_7 + h = 100$$

Sumando todo el de capital disponible al momento de dar inicio la segunda etapa, deberá cumplirse una pero no necesariamente las dos restricciones siguientes.

$$h + (18 + 45)X_3 + (15 + 35)X_4 + (9 + 25)X_1 \geq 100$$

o bien

$$h + (18 + 45)X_3 + (15 + 35)X_4 + (16 + 40)X_2 \geq 160$$

Sin embargo este tipo de restricciones excluyentes, tal cual, no pueden ser planteadas en nuestro modelo; para esto se requiere un replanteamiento, de manera que todas las restricciones especificadas deben cumplirse, dicho replanteamiento se hace de la siguiente forma:

debe cumplirse

$$\begin{cases} h + (18 + 45)X_3 + (15 + 35)X_4 + (9 + 25)X_1 \geq 100 \\ h + (18 + 45)X_3 + (15 + 35)X_4 + (16 + 40)X_2 + M \geq 160 \end{cases}$$

o bien deberá cumplirse

$$\begin{cases} h + (18 + 45)X_3 + (15 + 35)X_4 + (9 + 25)X_1 + M \geq 100 \\ h + (18 + 45)X_3 + (15 + 35)X_4 + (16 + 40)X_2 \geq 160 \end{cases}$$

Ya que sumar M (cualquier número extremadamente grande) al primer miembro de estas restricciones, tiene el efecto de eliminarlas, puesto que automáticamente son satisfechas por cualquier solución que respeta las otras restricciones. M es un valor constante, lo suficientemente grande como para exceder los valores fijos 100 y

160. De hecho M puede ser encontrada fácilmente al considerar

$$M = \text{máximo } \{100,\ 160\} = 160$$

así que deben cumplirse las dos siguientes restricciones :

$$\begin{cases} h + (18 + 45)x_3 + (15 + 35)x_4 + (9 + 25)x_1 + M^*y \geq 100 \\ h + (18 + 45)x_3 + (15 + 35)x_4 + (16 + 40)x_2 + M^*(1-y) \geq 160 \end{cases}$$

$$y = 0 \text{ ó } 1$$

Y como la nueva variable y debe ser 0 ó 1, el planteamiento garantiza que una de las dos restricciones originales debe cumplirse, mientras que la otra, en efecto queda eliminada.

Este enfoque está relacionado directamente con nuestro análisis de las decisiones "si-no" : 1) Tiene que seleccionarse la primera restricción?, 2) Debe seleccionarse la segunda restricción?. Más aún, éstas son alternativas mutuamente excluisvas, como lo asegura en forma automática el hecho de que $y + (1-y) = 1$. Si por el contrario se han usado variables binarias separadas, y_1 y y_2, para representar éstas decisiones sí o no, se necesita la restricción adicional $y_1+y_2=1$, para hacerlas mutuamente excluisvas. Esto lo mencionamos, porque el mismo tipo de ideas puede ser generalizado para el caso en que de un conjunto de N restricciones posibles, deban cumplirse sólo K de ellas (así, el caso que acabamos de considerar, es el caso N=2 y K=1) y su generalización es:

Considere las N restricciones:

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &\leq d_1 \\ f_2(x_1, x_2, \ldots, x_n) &\leq d_2 \\ &\vdots \\ f_N(x_1, x_2, \ldots, x_n) &\leq d_n \end{aligned}$$

Entonces utilizando la misma lógica de nuestro ejemplo, un planteamiento para el caso en que deban cumplirse K de las N restricciones es :

$$f_1(X_1,X_2,\ldots,X_n) \leq d_1 + My_1$$
$$f_2(X_1,X_2,\ldots,X_n) \leq d_2 + My_2$$
$$\vdots \qquad\qquad \vdots$$
$$f_N(X_1,X_2,\ldots,X_n) \leq d_n + My_n$$

$$\sum_{i=1}^{N} y_i = N - K$$

$$y_i = 0 \text{ ó } 1 \qquad i = 1,2,\ldots,n$$

donde se supone que $K < N$, y M es un número extremadamente grande. Así, al imponer la restricción para las y_i's, se garantiza que K de estas variables serán iguales a cero y las restantes serán iguales a uno, equivalentemente, K de las restricciones originales quedarán intactas y el resto, en efecto se eliminarán.

Resumiendo los tres tipos básicos de restricciones que deben agregarse al problema de los proyectos de inversión, obtenemos el siguiente modelo de programación cero-uno que es casi definitivo:

$$\begin{aligned}
&Max\ Z = 9X_1 + 12X_1' + 16X_2 + 15X_2' + 18X_3 + 15X_4 + 6X_5 + 12X_6 + 11X_7 \\
&\text{sujeto a} \\
&25X_1 + 40X_2 + 45X_3 + 35X_4 + 15X_5 + 30X_6 + 25X_7 + h = 100 \\
&X_1 + X_2 = 1 \\
&X_1' = X_1 \\
&X_2' = X_2 \\
&h + (18 + 45)X_3 + (15 + 35)X_4 + (9 + 25)X_1 + M*y \geq 100 \\
&h + (18 + 45)X_3 + (15 + 35)X_4 + (16 + 40)X_2 + M*(1-y) \geq 160 \\
&X_i,\ X_i' = 0 \text{ ó } 1 \text{ para toda } i, \text{ y } h \geq 0 \text{ entero}
\end{aligned}$$

Y es casi definitivo, porque se escapa un último detalle para asegurar que el modelo es estrictamente uno de programación cero-uno, ya que la restricción $h \geq 0$ y entero, no es una restricción del tipo cero-uno. No obstante, esta misma condición tiene un planteamiento equivalente y adecuada al formato de la programación cero-uno, para lo cual es necesario encontrar un valor, digamos U tal que

$$0 \leq h \leq U$$

con

$$2^{n-1} \leq U \leq 2^n$$

y entonces, h puede ser expresado como

$$h = \sum_{i=0}^{n} 2^i y_i \qquad y_i = 0 \text{ ó } 1 \qquad i = 1,2,\ldots,n$$

Con lo que al sustituir h por las n variables cuyos valores son cero o uno el modelo anterior es ya propiamente uno de programación cero-uno. Esto resulta simplemente de poder expresar cualquier número finito especificado en la base usual (base 10) a base 2.

En nuestro caso, dado que es seguro que debe invertirse en los proyectos 1 ó 2 (pero sólo en uno de ellos) y se dispone de 100 millones, entonces el capital sobrante en la primera etapa es a lo más

$$h = \{ 100 - \min\{25, 40\} \} = 75$$

y como

$$2^6 < 75 \leq 2^7 \qquad (\; 2^6 = 64 \;\; y \;\; 2^7 = 128 \;)$$

se tiene que

$$75 = 2^0 \cdot 1 + 2^1 \cdot 1 + 2^2 \cdot 0 + 2^3 \cdot 1 + 2^4 \cdot 0 + 2^5 \cdot 0 + 2^6 \cdot 1 + 2^7 \cdot 0$$

además todo número menor de 75 se expresa en este formato, sólo que puede tener diferentes combinaciones de ceros y unos, de esta forma la variable h que es menor a 75 es expresada como:

$$h = 2^0 \cdot y_0 + 2^1 \cdot y_1 + 2^2 \cdot y_2 + 2^3 \cdot y_3 + 2^4 \cdot y_4 + 2^5 \cdot y_5 + 2^6 \cdot y_6 + 2^7 \cdot y_7$$

$$\text{con} \quad y_i = 0 \text{ ó } 1 \qquad i = 1, \ldots, 7$$

lo que se traduce en agregar 7 variables al modelo, para ajustarlo a uno de programación cero-uno.

$$h = \sum_{i=0}^{7} 2^i y_i \qquad y_i = 0 \text{ ó } 1 \quad i=1,\ldots,7$$

En realidad, esta misma idea puede hacerse más eficiente, siempre que se encuentre una cota U lo más cercana al valor h. En nuestro caso particular, se pudo haber encontrado un valor que acote mejor al capital sobrante del primer periodo y con ello expresar la variable h en este formato, pero utilizando un menor número de variables auxiliares. Sin embargo, esto ya no es realizado, pero lo que si quisieramos remarcar es que, de una u otra forma, el problema de selección de proyectos de inversión ha podido ser planteado en un modelo de programación binaria y así mismo, nos ha permitido mostrar todos los tipos de restricciones y situaciones que pueden ser planteadas en este modelo, sin que con ello perdamos el reflejo de lo que en la práctica realmente sucede.

No obstante, lo que aún falta analizar, es qué tan conveniente resulta el plantear los problemas en este tipo de modelo. Ya que si mediante esta forma no es posible encontrar una solución de manera ágil y segura, nos interesaría buscar otras alternativas de solución mucho más eficientes. El problema que se presenta a continuación, pretende hacernos reflexionar un poco sobre éste aspecto; aunque en realidad todo esto será analizado posteriormente y a lo largo de todo el presente trabajo.

EJEMPLO 1.2.3. (El Problema del Agente Viajero)

Supóngase que un vendedor necesita viajar para visitar n ciudades, exactamente una vez cada una. La distancia entre cada par de ciudades ij, denotado por d_{ij} ($i \neq j$), es conocida, pero ella puede depender de la dirección del viaje (es decir, d_{ij} no necesariamente es igual a d_{ji}) .

El problema es encontrar un recorrido que comience y termine en la ciudad donde se encuentra la casa del vendedor y minimice el total de distancia recorrida .

SOLUCION

Supóngase que etiquetamos la ciudad de partida como ciudad 0 y como ciudad n+1. También introducimos la variable cero-uno X_{ij} ($i=0,1,\ldots,n$, $j=1,\ldots,n+1$), donde $X_{ij} = 1$ si el vendedor viaja de la ciudad i a la ciudad j y $X_{ij} = 0$ en otro caso.

De principio el modelo debe reflejar la consistencia que se tiene cada vez que se hace un recorrido, esto es, garantizar que para llegar a una ciudad intermedia j ($j=1,\ldots,n+1$), debemos partir de una única ciudad i ($i=0,\ldots,n$; $i \neq j$ claro está). Excluimos el caso j=0 porque al crear el nodo n+1 (punto final del recorrido) conlleva la implicación de que ninguna ciudad i llegará a la ciudad origen 0, lo mismo ocurre con el caso i=n+1. Lo anterior se asegura con la restricción:

$$\sum_{i=0}^{n} X_{ij} = 1 \qquad (j=1,\ldots,n+1 \ , \ i \neq j)$$

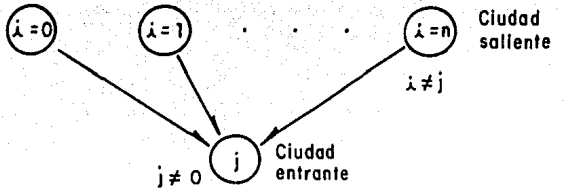


FIGURA 1

Similarmente el modelo debe asegurar que al salir de cualquier ciudad i (i=0,...,n+1) podemos llegar a una única ciudad j (j=1,...,n+1; j≠i). Lo que es planteado como:

$$\sum_{j=1}^{n+1} x_{ij} = 1 \qquad (\ i=0,1,\ldots,n\ ,\ i\neq j\ )$$

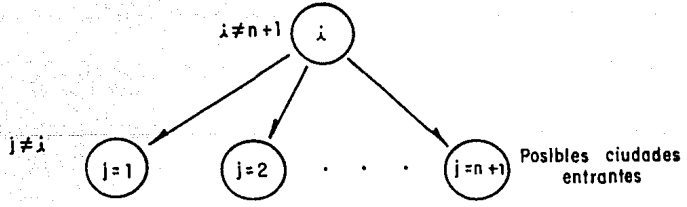


FIGURA 2

Estas restricciones, sin embargo, no eliminan la posibilidad de ciclos como lo indica la figura 3.

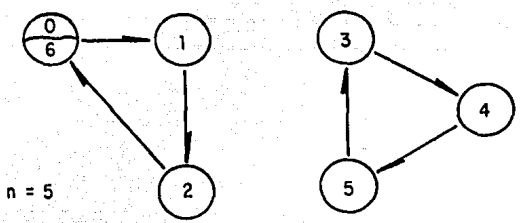


FIGURA 3

Donde la solución con n = 5 es

$$\begin{cases} X_{01} = X_{12} = X_{26} = 1 \\ X_{34} = X_{45} = X_{53} = 1 \\ X_{ij} = 0 \quad \text{en otro caso} \end{cases}$$

Una forma de eliminar la posibilidad de ciclos es agregando las restricciones

$$\alpha_i - \alpha_j + (n+1)X_{ij} \leq n$$

$$(\ i=0,1,\ldots,n\ )$$

$$(\ j=1,\ldots,n+1\ )$$

$$i \neq j$$

donde α_i es un número real asociado con la ciudad i .

Esto demuestra que una solución conteniendo ciclos no puede satisfacer esta restricción, excepto aquel ciclo que contiene a la ciudad de partida. Porque si nosotros sumamos las desigualdades correspondientes para $X_{ij} = 1$ que está dentro de un ciclo, un elemento del término $\alpha_i - \alpha_j$ cancela un elemento correspondiente del siguiente término y tendríamos que

$$(n+1)*N \leq n*N$$

donde N es el número de arcos en el ciclo, lo cual es una contradicción. Relacionando esto con el ejemplo de la figura 3 mostrada anteriormente y el ciclo 3->4->5, las restricciones son:

$$\alpha_3 - \alpha_4 + 6 \leq 5$$
$$\alpha_4 - \alpha_5 + 6 \leq 5$$
$$\alpha_5 - \alpha_3 + 6 \leq 5$$

Y sumando las tres desigualdades obtenemos que $18 \leq 15$, lo cual es una contradicción. ∎

Por otro lado, para verificar que estas restricciones son satisfechas cuando no hay ciclos, mostramos que existen α_{i}'s tales que hacen ciertas dichas restricciones. Definamos para ello $\alpha_0 = 0$, $\alpha_{n+1} = n+1$ y sea $\alpha_i = k$ si la ciudad i es la k-ésima visitada en el recorrido. Entonces, cuando $X_{ij} = 1$, obtenemos la siguiente relación:

$$\alpha_i - \alpha_j + (n+1) = k - (k+1) + n+1 = n$$

También porque definidas así las α_{i}'s tenemos que $1 \leq \alpha_i \leq n+1$ $(i=1,\ldots,n+1)$ y además $\alpha_i - \alpha_j \leq n$, con lo que las restricciones son satisfechas cuando $X_{ij} = 0$.

Para complementar el modelo, debemos minimizar el total de distancia recorrida, es decir:

$$Min \quad Z = \sum_{i=0}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n+1} d_{ij} X_{ij}$$

Por tanto, debemos encontrar los valores de las variables X_{ij} y números reales arbitrarios α_i, tales que:

$$
\begin{cases}
\textit{Minimizar} \quad \sum_{i=0}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n+1} d_{ij} X_{ij} \\
\\
\text{sujeto a} \\
\\
\sum_{\substack{i=0 \\ i \neq j}}^{n} X_{ij} = 1 \qquad (j = 1, \ldots, n+1) \\
\\
\sum_{\substack{j=1 \\ j \neq i}}^{n+1} X_{ij} = 1 \qquad (i = 0, 1, \ldots, n) \\
\\
\alpha_i - \alpha_j + (n+1) X_{ij} \leq n \qquad (i = 0, 1, \ldots, n) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (j = 1, \ldots, n+1) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i \neq j \\
\\
X_{ij} = 0, 1 \qquad (i = 0, 1, \ldots, n, \quad j = 1, \ldots, n+1, \quad i \neq j). \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{donde } X_{0,n+1} = 0
\end{cases}
$$

Observese que el modelo tal como se presenta, no es propiamente de programación cero-uno, pero si lo es de programación entera mixta, ya que la variables X_{ij} sin mayor problema son relajadas a valores enteros, pues no puede haber valores enteros, distintos de 0 y 1, que cumplan con:

$$\sum X_{ij} = 1$$

Además las variables $\alpha_{i's}$ son variables continuas, lo cual implica que el modelo es de programación entera-mixta, modelo que incluso puede ser resuelto por la técnica de ramificación y acotamiento en conjunción con el algoritmo simplex, pero que no es tratado aquí.

Una forma alternativa de resolver el problema del viajero es presentada a continuación y en este caso el problema es planteado en un modelo de programación cero-uno. Con ello podremos apreciar como a menudo el modelo de programación cero-uno es muy grande en comparación con otros modelos asociados con el mismo problema, aún que en realidad la eficiencia no radica únicamente en el tamaño del modelo si no también en el algoritmo mismo.

Para esto redefiniremos nuestras variables como sigue:

$$x_{ijk} = \begin{cases} 1 & \text{si el recorrido de la ciudad } i \text{ a la ciudad } j \text{ se hace en la } K\text{-ésima etapa del viaje .} \\ 0 & \text{en otro caso .} \end{cases}$$

Es decir ahora a cada recorrido de i a j le asociamos n-variables, que están asociadas con el número de viaje hecho de una ciudad a otra, a lo largo del recorrido.

Definidas así las variables, las restricciones del problema son planteadas y clasificadas bajo 4 tipos :

a.- Dado que para cada etapa habrá sólo una ciudad i de partida y otra j de llegada, entonces debe suceder que

$$\sum_i \sum_j x_{ijk} = 1 \qquad k=1,\ldots,n$$

b.- Debe suceder también que para cada ciudad de llegada j hay una sola etapa k del viaje que principia en i

$$\sum_i \sum_k x_{ijk} = 1 \qquad j=1,\ldots,n$$

c.- Análogamente, debe ocurrir que para cada ciudad de partida i hay sólo una etapa k del viaje que llega a j

$$\sum_j \sum_k x_{ijk} = 1 \qquad i=1,\ldots,n$$

d.- Para eliminar la posibilidad de viajes inconexos, cuando en la k-ésima etapa el viaje es realizado de la ciudad i a la ciudad j,

partir de cualquier ciudad y llegar a la ciudad J, entonces la (k+1)-ésima etapa del viaje se realiza comenzando única y exclusivamente de la ciudad J. De este modo, se obtine la ecuación:

$$\sum_{i \neq j} x_{ijk} = \sum_{r \neq j} x_{jr(k+1)} \qquad \text{para todo } J \text{ y } k$$

El modelo se complementa con la función objetivo:

$$\textit{Minimizar } Z = \sum_{i} \sum_{j} \sum_{k} \sum_{i \neq j} d_{ij}\, x_{ijk}$$

Obsérvese que este nuevo modelo se aplica independientemente de cual sea la ciudad de partida. Pues al asegurar que sólo se cruza una única vez cada ciudad, se estará obteniendo con seguridad un circuito de ciudades (entendiendo por esto, que cada ciudad tiene exactamente un arco de salida y un arco de llegada y además todas las ciudades están conectadas), por lo que si partimos de una ciudad cualquiera del circuito, el recorrido nos llevará a esa misma ciudad de partida, concluyendo por tanto que cualquier ciudad puede ser considerada como ciudad origen.

Además, el modelo es aplicable cuando se trate tanto de un vendedor viajero que ya tiene pedidos o entregas fijadas previamente en cada ciudad, como para aquellos casos en que se trate de un vendedor que va a ofrecer y que por ende esté sujeto a la aleatoriedad de las ventas efectuadas, pues en este último caso bastará hacer ciertos estudios de tipo estadístico para saber hasta qué cantidades pueden ser vendidas en cada ciudad y en base a esto y a los costos de trasportación, analizar qué recorridos conviene eliminar de la red debido a que existe pérdida en las ventas para dicha ciudad (puede suceder que se eliminen todas las posibles rutas de acceso a una ciudad, cosa que eliminará toda posibilidad de visitar dicha ciudad). Con lo que a final de cuentas tendríamos que resolver el mismo problema del agente viajero, pero restringido a sólo un grupo

de recorridos o en su caso un grupo de ciudades que vale la pena visitar debido a que prometen dejar buena ganancia de las ventas efectuadas. Incluso si se trata de un vendedor que se dedique sólo a eso, debería analizar a detalle para cada unidad de tiempo (días, semanas, meses) cuál es el grupo de ciudades que periódicamente conviene visitar más y en base a ello y a las posibles vías de acceso a las mismas, formarse un itinerario de las ciudades a visitar en cada lapso de tiempo, aplicando en cada caso el mismo modelo, sólo que para la sub-red restringida.

La modelación del problema del agente viajero, es realizada en este caso, en los modelos de programación entera mixta y de programación cero uno; si analizamos cual de los modelos en este caso es mejor, veamos que al considerar el modelo de programación entera mixta, para el problema del agente viajero con 5 ciudades, tendríamos un total de 36 variables y 30 restricciones y para 10 tendríamos 121 variables y 110 restricciones. Mientras que con el modelo cero-uno, se tienen un total de 125 variables y 40 restricciones para el caso de 5 ciudades y un total de 1000 variables y 130 restricciones para el caso de 10 ciudades.

MODELO DE PROGRAMACION		Número de variables	Número de restricciones
ENTERA MIXTA	n=5	36	30
	n=10	121	110
CERO UNO	n=5	125	40
	n=10	1000	130

Lo anterior sugiere resolver el problema del agente viajero vía programación entera mixta; sin embargo, la eficiencia no sólo

radica en la simplicidad del modelo sino también en la rapidez de la técnica para resolverlo. Pues como veremos más adelante, este problema puede también ser visto como un problema de asignación bi-dimensional. Para estos problemas existen algoritmos que explotando su estructura, obtienen más rápidamente la solución óptima que los de programación entera y cero-uno, algunos de los cuales también están fundados en las mismas ideas que surgen de la técnica de Ramificación y Acotamiento .

1.3. Propiedades Basicas

En general decimos que un problema es de programación cero-uno, si éste se ajusta al modelo referido para seleccionar los valores $x_1, x_2, \ldots, x_n$, con el propósito de:

$$\left\{ \begin{array}{l} \textit{Maximizar} \\ \textit{Minimizar} \end{array} \right\} \quad Z = C_1x_1 + C_2x_2 + \ldots + C_nx_n$$

sujeto a

$$A_{j1}x_1 + A_{j2}x_2 + \ldots + A_{jn}x_n \left\{ \begin{array}{c} \leq \\ \geq \\ = \end{array} \right\} B_j \qquad j = 1,\ldots,m$$

$$x_i \in \{ 0, 1 \} \qquad i = 1,\ldots,n$$

Z será la **función objetivo** y x_i las **variables de decisión**. Ahora bien, veamos algunas características que deben satisfacer los problemas que son planteados en este modelo:

a.- La asignación de cada recurso debe ser hecha de manera completa o bien no debe hacerse.

b.- La medida de efectividad de cada recurso es independiente de las demás actividades.

Para ello supónga que sólo se emprende una de las n actividades, llamémosla la actividad k, de modo que

$$X_j = 0 \text{ para todo } \quad j = 1,2,\ldots,n \quad \text{excepto } j = K$$

entonces, la suposición, en este caso es:

- La medida de efectividad Z es igual a $C_k X_k$ y
- El uso de cada recurso i en la actividad k es igual a $A_{ik} X_k$.

c.- En cuanto a la función objetivo y las funciones de restricción, éstas no deben contemplar términos de actividades "cruzados" (que significaría tener interacciones entre algunas de las actividades, que cambiarían la medida de efectividad o el uso total de algún recurso). Por lo cual se requiere que, dados los niveles cualesquiera de actividad (X_1, X_2, ..., X_n), el uso de cada recurso y la medida total resultante de efectividad sea igual a la suma de las cantidades correspondientes generadas por cada actividad conducidas por sí mismas.

Existe, sin embargo, una alternativa para representar aquellas situaciones en las que los términos "cruzados" deben ser involucrados:

Suponga que el problema binario no-lineal es el siguiente:

$$\textit{Maximizar} \quad Z = X_1^2 - X_1 X_2 + X_2^2$$

sujeto a

$$X_1, X_2 = 0 \text{ ó } 1.$$

por ser x_1 variable cero-uno, el término x_1^2 equivale a tener simplemente x_1. Y si reescribimos el producto no-lineal x_1x_2 como $x_3 = x_1x_2$, entonces el mismo problema es planteado como un problema de programación cero-uno, solo que con 2 restricciones adicionales.

$$\textit{Maximizar} \quad Z = x_1 - x_3 + x_2$$

sujeto a

$$x_1 + x_2 - x_3 \leq 1$$
$$-x_1 - x_2 + 2x_3 \leq 0$$
$$x_1, x_2, x_3 = 0 \text{ ó } 1.$$

ya que de esta forma

(1) Si $x_1 = x_2 = 0$ entonces de las restricciones tenemos que $-1 \leq x_3 \leq 0$ y por tanto $x_3 = 0$.

(2) Si $x_2 = 1$ y $x_1 = 0$ o viceversa, entonces de las restricciones se tiene que $x_3 \geq 0$ y $2x_3 \leq 1$, por lo que $x_3 = 0$.

(3) Por último, si $x_1 = x_2 = 1$, se tiene $x_3 \geq 1$ y $x_3 \leq 1$, y entonces $x_3 = 1$.

En el caso general, siempre que se transforme un término de la forma $x_1^{n_1} \cdot x_2^{n_2} \cdot \ldots \cdot x_m^{n_m}$ debemos hacer la substitución

$$y_m = x_1^{n_1} \cdot x_2^{n_2} \cdot \ldots \cdot x_m^{n_m}$$

y agregar las restricciones

$$\sum_{i=1}^{m} x_i - y_m \leq m-1$$

$$-\sum_{i=1}^{m} x_i + m y_m \leq 0$$

$\therefore$

Así por ejemplo el problema

$$\textit{Maximizar}\quad Z = x_1 - 2x_2 + 8x_1x_2x_3 + x_2x_3 + x_1x_3$$

sujeto a

$$\begin{aligned} x_1 + 2x_2 &\le 2 \\ x_2 + 2x_3 &\le 3 \end{aligned}$$

$$x_1, x_2, x_3 = 0 \text{ ó } 1.$$

es transformado a

$$\textit{Maximizar}\quad Z = x_1 - 2x_2 + 8y_1 + y_2 + y_3$$

sujeto a

$$\left.\begin{aligned} x_1 + 2x_2 &\le 2 \\ x_2 + 2x_3 &\le 3 \end{aligned}\right\} \text{ variables originales}$$

$$\left.\begin{aligned} x_1 + 2x_2 + x_3 - y_1 &\le 2 \\ -x_1 - 2x_2 - x_3 + 3y_1 &\le 0 \end{aligned}\right\} \begin{array}{l}\text{de la} \\ \text{substitución} \\ y_1 = x_1 * x_2 * x_3\end{array}$$

$$\left.\begin{aligned} x_2 + x_3 - y_2 &\le 1 \\ -x_2 - x_3 - 2y_2 &\le 0 \end{aligned}\right\} \begin{array}{l}\text{de la} \\ \text{substitución} \\ y_2 = x_2 * x_3\end{array}$$

$$\left.\begin{aligned} x_1 + x_3 - y_3 &\le 1 \\ -x_1 - x_3 - 2y_3 &\le 0 \end{aligned}\right\} \begin{array}{l}\text{de la} \\ \text{substitución} \\ y_3 = x_1 * x_3\end{array}$$

$$x_1, x_2, y_1, y_2, y_3 = 0 \text{ ó } 1.$$

1.4. METODOS DE SOLUCION

En cuanto a los métodos de solución se refiere, la restricción de que cada variable deba tomar sólo valores 0 ó 1, es difícil de manejar. No obstante se han hecho algunos progresos en el desarrollo de los métodos, aún cuando no se ha obtenido un método que sea tan eficiente como el método simplex en programación lineal.

En realidad, una forma de atacar los problemas de programación cero-uno, es el utilizar el método simplex para lo cual es necesario agregar restricciones de que cada variable sea menor o igual a uno, lo que garantizará que la solución óptima resultante será tal que cada variable X_i cumple con

$$0 \leq X_i \leq 1$$

y posteriormente lo que se puede hacer es redondear al valor cero o al valor uno aquellas variables X_i que hayan resultado tener un valor fraccionario en la solución óptima.

Sin embargo, aún cuando esto facilitase el trabajo para algunos problemas sencillos, existen riesgos con este enfoque. Uno de los cuales es que esta solución no necesariamente es factible, ya que con frecuencia, es difícil ver en qué sentido deben redondearse las variables para mantener la factibilidad. Puede incluso ser necesario alterar los valores de otras variables después de haber hecho el redondeo. Además, no obstante que la solución óptima de programación lineal se redondease de manera satisfactoria, existe todavía otro riesgo ya que no existe garantía de que esta solución sea la óptima para el problema cero-uno.

Por otra parte, existen problemas específicos para los cuales no es

del todo erróneo este enfoque, por ejemplo para problemas que son del tipo

$$\left\{ \begin{array}{c} \textit{Maximizar} \\ \text{o} \\ \textit{Minimizar} \end{array} \right\} Z = \sum_{i} \sum_{j} C_{ij} x_{ij}$$

sujeto a

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

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

$$x_{ij} = 0 \text{ ó } 1$$

los cuales en realidad son problemas de asignación. Problemas que se presentan, por ejemplo, cuando se necesita asignar hombres y máquinas de manera única, pero optimizando los costos de asignación

$$\begin{array}{cc} & \begin{array}{cccccc} & M & A & Q & U & I & N & A & S \\ & 1 & 2 & 3 & . & . & . & n \end{array} \\ \begin{array}{c} H \\ O \\ M \\ B \\ R \\ E \\ S \end{array} & \begin{array}{c} 1 \\ 2 \\ . \\ . \\ n \end{array} \left[\begin{array}{cccccc} C_{11} & C_{12} & C_{13} & \ldots & C_{1m} \\ C_{21} & C_{22} & C_{23} & \ldots & C_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ C_{n1} & C_{n2} & C_{n3} & & C_{nm} \end{array} \right] \end{array}$$

C_{ij} = Costo de asignar el hombre i a la máquina j.

problemas que, resolviéndolos por el método simplex de programación lineal, se asegura que siempre se obtendrán soluciones con valores 0 ó 1 (para esto, agregando la restricción de que cada variable sea menor ó igual a uno).

Lo anterior se justifica por un resultado de programación lineal, que vale la pena enunciar aún cuando no se incluya su demostración.

Resultado.

Dado el problema (en forma matricial)

$$Max \quad Z = C\,\bar{X}$$

sujeto

$$A\bar{X} = B$$
$$\bar{X} \geq 0$$

Si B es un vector en enteros y A es tal que el determinante de cualquier submatriz cuadrada es ±1, entonces, en caso de que exista una solución básica optima $\bar{X}^{*}$, dicha solución será en enteros.

Nótese que Una condición necesaria para que el determinante de cualquier submatriz de A sea ±1, es que A conste sólo de valores -1,0 ó 1.

En nuestro caso particular, no es difícil ver que se satisfacen las condiciones del resultado y sin mayor dificultad podemos afirmar que el problema de asignación puede ser resuelto vía programación lineal, cosa que en realidad es una forma eficiente de resolver este tipo de problemas (incluso este tipo de solución ha sido mejorada, pues se ha desarrollado un algoritmo especializado para estos casos, que sigue la metodología del algoritmo simplex, pero sin la necesidad de acarrear cada tabla simplex).

En el capitulo 3, veremos otra forma también bastante eficiente, para resolver este tipo específico de problemas, que se basa en los algoritmos que surgen de las técnicas de ramificación y acotamiento. Técnicas que, explotando la estructura de este tipo de problemas específicos, logran obtener algoritmos eficientes.

Sin embargo, cuando el problema no es referido a un caso especifico, una alternativa de solución es utilizar los métodos de la programación entera (problemas que se restringen sólo a tomar valores enteros en sus variables solución), entre los que podemos citar:

(1). METODOS DE PLANO Y CORTE.
(2). METODOS DE TEORIA DE GRUPOS.
(3). METODOS DE PROGRAMACION DINAMICA.
(4). METODOS HEURISTICOS.
(5). METODOS DE RAMIFICACION Y ACOTAMIENTO.

Siendo la técnica de ramificación y acotamiento la que interactuando con el método simplex de programación lineal, logra una forma buena y eficiente de resolver problemas con valores enteros, pues los demás métodos resultan muchas veces ineficientes y en pocas ocasiones logran ser eficientes, pero sólo para un grupo pequeño de problemas especificos.

Mas aún, para el caso de querer resolver en general un problema de programación cero-uno, existe otro método propiamente ideado para esta clase de problemas, se trata del método de ENUMERACION IMPLICITA, el que resulta a menudo ser más eficiente para muchos problemas de este tipo y que surge también de las ideas relacionadas con la técnica de ramificación y acotamiento .

Es aquí donde centraremos nuestra atención y donde presentaremos la variedad de algoritmos con los que cuenta la Técnica de Ramificación y Acotamiento, haciendo especial énfasis en aquellos propios para la programación cero-uno y que pueden ser algoritmos que en general permitan resolver cualquier problema de programación cero-uno o que, explotando la estructura especifica de ciertos problemas, logran una mayor eficiencia.

1.5. DISCUSION

Hemos presentado el modelo general de programación cero-uno y sus propiedades básicas. Sin embargo no hay que perder de vista, que al desarrollar un modelo de programación cero-uno, se pretende predecir cuál sería la solución óptima, dadas las condiciones iniciales del problema. Al hacer esta afirmación, se supone se ha podido captar la situación real (esto es, el problema real que queremos resolver) por una definición apropiada de variables y restricciones que conformarán nuestro modelo matemático. En muchos casos, de hecho, nuestra habilidad de retratar la realidad, lo genuino, está expuesto a discusión. Así como en todas las áreas de de la actividad humana, encontraremos que arreglos e ingeniosidades ayudan a obtener una mejor comprensión de los procesos en cuestión y que pretendemos conduzcan a un modelo matemático que produzca una solución que pueda ponerse en práctica.

Al desarrollar la formulación de un modelo de programación cero-uno debemos cuidarnos de que no se nos acuse de tener una herramienta para hacer el trabajo y, si la herramienta no se ajusta, cambiar el trabajo para que éste se ajuste a la herramienta. Esta advertencia no regula, ni debe hacerlo, nuestra libertad de hacer las suposiciónes de simplificación apropiadas en vista del deseo de desarrollar modelos que capten los aspectos esenciales del problema. Así nuestros modelos matemáticos deben producir respuestas que sean comprensibles a las personas responsables del proceso que se está estudiando, o en algunos casos ser simplemente útiles para los tomadores de decisiones.

2. LA TECNICA DE RAMIFICACION Y ACOTAMIENTO

2.1. INTRODUCCION

Debido a que cualquier problema de programación cero-uno acotado tiene sólo un número finito de soluciones factibles, es natural pensar en usar algún tipo de procedimiento de enumeración para encontrar una solución óptima. Sin embargo, cada vez que analicemos problemas con decisión múltiple, la enumeración de todas estas posibles alternativas puede a menudo ser tediosa y larga. Por ejemplo, si se tienen 10 variables cada una pudiendo asumir los valores 0 ó 1, entonces habra $2^{10} = 1024$ soluciones factibles y aún cuando hoy en dia algunas computadoras pueden realizar varios millones de operaciones aritméticas por segundo, la enumeración exhaustiva significará un enorme consumo en tiempo. Es imperativo por lo tanto, que cualquier procedimiento de enumeración se estructure en forma hábil para que sólo una pequeña parte de las soluciones factibles tengan que examinarse.

Este enfoque es parecido al proporcionado por la Técnica de la Ramificación y el Acotamiento (RyA), la que se caracteriza por jerarquizar las distintas alternativas de decisión, considerando sólo una parte del total posible, pues muchas de ellas son eliminadas por criterios que establecerán que tales alternativas no pueden ser óptimas.

Así la técnica de RyA consiste en la aplicación recursiva de dos operaciones :

1.) **Operación de Ramificación.** Consistente en dividir un conjunto de alternativas de decisión en subconjuntos ajenos y cuya unión es el conjunto original.

2.) **Operación de Acotamiento.** Consistente en asignar una cota a la función objetivo en un conjunto de alternativas de decisión especificadas.

El proceso es repetitivo y se utilizan ambas operaciónes de manera alternada para ir eliminando aquellos conjuntos de decisiones que no contienen la solución óptima, continuando así hasta determinar el que la contiene, si es que existe dicha solución.

La técnica proporciona en general una variedad de estrategias para la ramificación y el acotamiento de cada subconjunto de soluciones factibles, de las cuales se han desprendido algoritmos bastante eficientes como los que se presentan en el capitulo 3. Sin embargo antes de mostrar estos algoritmos específicos, es necesario entender el funcionamiento general de la técnica y para ello en el presente capítulo presentamos algunos problemas sencillos que pueden resolverse siguiendo las ideas básicas del método (sección 2.2.), lo que sirve de motivación para ver cómo es que surge esta técnica y posteriormente son presentadas las reglas más o menos generales utilizadas por la misma (sección 2.3.), con el propósito de dar un panorama sobre las diferentes estrategias, a fin de adecuar alguna para el problema específico de que se trate.

2.2. ASPECTOS BASICOS

En esta sección veremos tres problemas sencillos (del tipo discreto) y que son resueltos mediante ideas muy sencillas, ideas que después se generalizan para ser utilizadas en la mayoría de los procesos seguidos en la técnica de RyA.

EJEMPLO 2.2.1.

Se busca el máximo de $f(x) = -3x^2 + \pi x$ tal que x sea entero.

SOLUCION

La ecuación no es del tipo DIOFANTINA, por lo que no es posible utilizar herramienta algebraica de este tipo para obtener el óptimo entero de esta función. Pero lo que si sabemos, es utilizar herramienta del cálculo para optimizar funciones cuyos valores se encuentren en algún intervalo continuo de los reales. Asi que relajando los valores de $x \in Z$ por $x \in R$, obtenemos fácilmente el valor óptimo de f, a saber $x_o = \frac{\pi}{6}$.

Pero debemos preguntarnos ¿COMO UTILIZAR LA INFORMACION DISPONIBLE PARA ENCONTRAR EL OPTIMO VALOR ENTERO?.

Con la relajación hecha se deduce que el problema P fue relajado al problema P_o

$$P_o : \max_{x \in R} \pi x - 3x^2$$

$$P : \max_{x \in Z} \pi x - 3x^2$$

por lo que el máximo de P_0 es una cota superior para el máximo de P, pero además el punto donde se alcanza el máximo de P debe estar alrededor del valor $X_0^* = \pi/6 \cong 0.52$. Esto sugiere ramificar el conjunto Z en dos conjuntos $S_1 = Z \cap \{X \leq 0\}$ y $S_2 = Z \cap \{X \geq 1\}$ (esto es particularizar cada vez más el conjunto de soluciones) y analizar cual de estos dos subconjuntos tiene el valor mayor, pero por otro lado si advertimos que la función f es cóncava podemos asegurar que S_1 y S_2 alcanzan su máximo valor precisamente en el punto frontera más cercano a X_0 y como $f(1) = \pi - 3 > 0 = f(0)$ concluimos que $X^* = 1$ es el máximo para el problema original P.

El proceso de optimización para P, lo podemos seguir en el árbol siguiente:

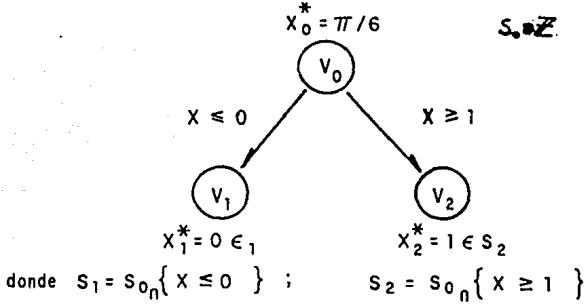


FIGURA 4

Antes de proseguir con el siguiente ejemplo es importante hacer notar dos hechos en el problema anterior:

a.- Al conjunto relajado se le impusieron dos restricciones: $X \leq 0$ y $X \geq 1$ (en forma separada), lo cual sin eliminar soluciones enteras, redujo a encontrar el óptimo en los valores frontera (pues la función es concava). Este hecho es muy útil ya que se generaliza para funciones lineales n-dimensionales, las cuales se comportan monótonamente.

b.- La solución al problema planteado se pudo representar mediante el concepto de "árbol de ramificación" el que permite hacer la enumeración sistematica de un número finito de posibles conjuntos solución.

EJEMPLO 2.2.2

¿ De cuantas formas se puede seleccionar enteros positivos para que su suma sea un cierto número entero I ?.

SOLUCION

Es obvio que los enteros positivos que formarán parte de la suma, deben ser menores al valor I. Es decir, deben ser elementos del conjunto $\{1,2,\ldots,I\}$. Entonces, el problema ahora se traduce primeramente a decidir para cada uno de los números del conjunto anterior, si dicho elemento debe o no estar dentro de los que conformarán la suma. Es decir, estaremos definiendo variables x_j que tomarán los valores 0 ó 1, como sigue:

$$x_j = \begin{cases} 1 & \text{Si se selecciona el entero } j \\ 0 & \text{en otro caso} \end{cases}$$

$$1 \leq j \leq I$$

Entonces el problema se traduce en encontrar las soluciones de la ecuación:

$$\sum_{j=1}^{I} j\, x_j = I$$

utilizando un árbol de enumeración, podemos ir examinando las

diferentes alternativas y para que el análisis se realize sistemáticamente, lo haremos empezando por el valor de X_1 y continuando de manera ascendente hasta llegar al valor de I.

Para el caso I = 5, las soluciones vienen dadas por los caminos únicos del vértice Vo a cada vértice terminal, marcado con un asterisco en el árbol de la enumeración y los vértices subrayados indican qué enumeración posterior es inútil, ya que ninguna variable posterior puesta a nivel de uno, podrá dar una solución a la ecuación (ver figura 5).

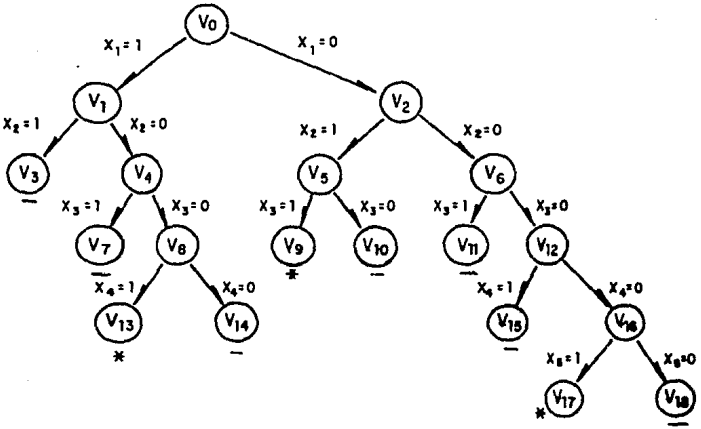


FIGURA 5

En general, dado un conjunto finito S, si se desea encontrar cada $X \in S$, entonces, cada vértice V_j del árbol que enumera los elementos

de S, restringe $\mathcal{X}$ a estar en un conjunto $S_j \subseteq S$. S_j es la intersección de S con el conjunto de aquellas $\mathcal{X}$ que satisfacen las restricciones marcadas por los arcos del camino que unen a V_0 con V_j .

EJEMPLO 2.2.3.

Las preferencias expresadas en un experimento de comparación apareada sobre el conjunto {a,b,c,d} son dadas en la siguiente matriz. El problema es encontrar un ordenamiento de los elementos del conjunto, que minimice el número de discrepancias, esto es, el número de preferencias "violadas" (el elemento x es ordenado arriba de y, siendo que y es preferente a x).

	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

SOLUCION 1

El ordenamiento se irá construyendo conforme se desarrollan las decisiones.

Partiendo de que 'a' y 'c' tienen una mejor puntuación sobre los el resto de los elementos y que además 'a' es preferido sobre 'c', entonces la primera decisión es hecha al responder la pregunta: ¿ podría 'a' ser ordenada arriba de 'c' o no ?. Esta es una

pregunta razonable que se plantea, donde al parecer, un ordenamiento donde 'c' es preferido sobre 'a' no necesita mayor consideración.

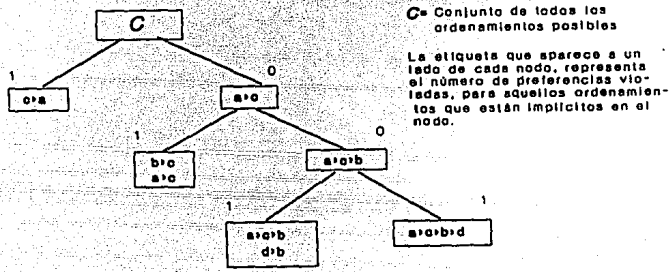


FIGURA 6

De los dos nodos del primer nivel del árbol, el primero con 'a>b' (esto es 'a' ordenada arriba de 'b') es el más prometedor, donde 'a>b' no implica -inmediatamente- discrepancias. De la matriz de preferencias se observa que 'c' también es notablemente mejor que 'b', y la segunda ramificación corresponde de este modo a 'c>b'.

Después de tres ramificaciones, el árbol de búsqueda desarrollado como se muestra en la figura 6, donde el ordenamiento 'a>c>b>d', involucra sólo una discrepancia y una solución correspondiente al subconjunto de soluciones. Cualquier otro nodo implica al menos una discrepancia, por lo que 'a>c>b>d' debe ser un ordenamiento óptimo. Por supuesto, no se ha demostrado en esta etapa que dicha solución es única, ya que para establecer esto o encontrar otras soluciones óptimas, es necesario desarrollar todos los nodos que implican sólo una discrepancia.

El método aplicado para el ejemplo anterior, resulta ser bastante eficiente, ya que de manera ágil encuentra una solución al problema propuesto, sin embargo en otros casos y debido a la manera juiciosa de escoger la ramificación, esto no hubiera sido posible, pues en general ninguna (al menos como estrategia) tiende a ser favorable.

SOLUCION 2

Sea $X_1 > X_2 > X_3 > X_4$ un ordenamiento, donde $X_i \in \{a, b, c, d\}$, entonces existen $4^4 = 256$ diferentes cuartetas (X_1, X_2, X_3, X_4) de las que sólo $4 \cdot 3 \cdot 2 = 24$ corresponden a ordenamientos válidos, es decir, de los 256 candidatos a solución, 232 no son factibles y deben ser eliminados.

La regla de ramificación que adoptaremos ahora es: La primera ramificación corresponde a la selección de X_1 (el elemento de la ordenación más alto) la segunda y tercera ramificaciones corresponden a la selección de X_2 y X_3 respectivamente (si X_1, X_2 y X_3 son especificados, entonces X_4 queda automáticamente determinada).

El árbol de búsqueda se desarrolla en forma similar que el primer método, como se muestra en la figura 7.

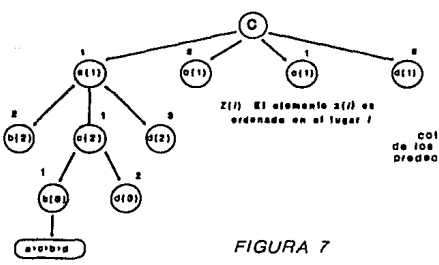


FIGURA 7

El segundo método aplicado proporciona nuevamente la ordenación a>c>b>d como una solución óptima. Mientras un nodo a(2) es sucesor potencial del nodo a(1), éste no se incluye en el árbol, ya que sólo contiene soluciones no factibles y en general conjuntos como éste son también rechazados.

En el ejemplo, ambos métodos de solución nos conducen directamente a un ordenamiento óptimo, pero en general éste no es el caso y a menudo cierto proceso de "retroceso" es necesario.

Además es justo mencionar que cada vez que es resuelto un problema discreto mediante la técnica de RyA, a menudo es conveniente relajar el conjunto de soluciones factibles a uno en el que es más fácil encontrar una solución, que si bien no será una solución para el problema original, sí nos dará un buen indicio de dónde se encuentra la solución buscada. Esta idea es muy utilizada en combinación con la técnica de RyA para resolver problemas de programación entera y en particular la utilizamos para resolver el ejemplo 2.1.1.. Así que, dada la importancia de este concepto, es conveniente formalizar dicho término de la manera siguiente.

DEFINICION:

Sean P_1 y P_2 los problemas :

P_1 : Min $f(X)$ sujeto a $X \in S_1$
P_2 : Min $f(X)$ sujeto a $X \in S_2$

$S_2 \subseteq S_1$ llamaremos a P_1 relajamiento de P_2
(Análogamente P_2 es una restricción de P_1) .

RESULTADO 2.2.1

Sea X_o la solución óptima de P_1, el cual es un relajamiento de P_2, entonces tenemos que:

a.- $f(X_0) \leq f(X)$ para todo $X \in S_2$

b.- Si $X_0 \in S_2$ entonces X_0 es la solución óptima de P_2

DEMOSTRACION

a.- Como X_0 es óptimo de $P_1 \Rightarrow f(X_0) \leq f(X)$ para todo $X \in S_1$, pero además como $X \in S_2 \Rightarrow X \in S_1$, por ser P_1 relajamiento de P_2, por tanto $f(X_0) \leq f(X)$ para todo $X \in S_2$.

b.- El inciso (a) implica que $f(X_0) \geq f(X)$ para todo $X \in S_2$, por lo que si $X_0 \in S_2$ tenemos que X_0 es el óptimo de S_2.

El resultado anterior es fácil de aplicar a problemas como los de programación entera (las incógnitas en este caso son vectores $\bar{X}$ con valores enteros). Para ello primero pedimos que $f(\bar{X})$ sea lineal, posteriormente relajamos el conjunto de soluciones factibles a S_1 y paulatinamente se restringe el conjunto S_1 a tomar valores enteros, como lo demuestra el conjunto S_2.

$S_1 = \{\bar{X} \in R^n \mid A\bar{X} = b,\ \bar{X} \geq 0\}$ y

$S_2 = \{\bar{X} \in R^n \mid A\bar{X} = b,\ \bar{X} \geq 0,\ X_j$ entero para algún $j\}$.

En estas condiciones, el inciso (a) implica que la solución al problema lineal continuo es una cota superior al problema en enteros (o mixto). El inciso (b) garantiza que si la solución del problema continuo es entera, entonces es una solución óptima al problema original.

En el caso que la solución óptima al problema relajado es tal que $X_0 \notin S_2$, es necesario definir un proceso que encuentre la solución óptima al problema original (o bien, que indique que ésta no existe). Una idea es "Reducir" S_1 hacia S_2, o más precisamente determinar conjuntos S_j tales que $S_1 \supset (S_1 \cap S_j) \supseteq S_2$ y aprovechar la información obtenida del problema relajado para obtener eficientemente la solución óptima de $S_1 \cap S_j$.

2.3. METODOS DE RAMIFICACION Y ACOTAMIENTO

El procedimiento general que sigue la técnica de RyA es caracterizado por un proceso que comienza particionando el conjunto $\mathcal{X}$ de soluciones factibles en conjuntos ajenos $\mathcal{X}_1,\ldots,\mathcal{X}_n$ (proceso de Ramificación), posteriormente debemos asignar a cada nodo una cota inferior a su función objetivo (caso de minimizar) e identificar el $\mathcal{X}_k$ que tiene la menor cota inferior, la que será una cota superior para el valor óptimo mínimo (proceso de ramificación). La Ramificación continua al desarrollar $\mathcal{X}_k$, esto es, particionandolo en subconjuntos $\mathcal{X}_{k1},\ldots\mathcal{X}_{ks}$ y nuevamente debemos determinar un nodo $\mathcal{X}_{k1}$ que prometa contener una mejor solución (puede incluso considerarse un nodo anteriormente creado), si tal nodo existe entonces el problema nuevamente ha sido reducido a optimizar sobre $\mathcal{X}_{k1}$ y así sucesivamente. Esta es la idea que se encuentra detrás de las técnicas de Ramificación y Acotamiento, esto es particionar un conjunto mayor en subconjuntos cada vez más pequeños y mediante un proceso de acotamiento ir descartando algunos de estos subconjuntos, conservando los subconjuntos más prometedores y continuando este proceso hasta analizar todas las posibilidades.

2.3.1. Reglas de Ramificación.

Para presentar las reglas de Ramificación más comúnmente usadas es necesario explicarlas mediante un problema muy particular, "el problema del ordenamiento", el cual consiste en ordenar un conjunto dado $A=\{a_1,\ldots,a_n\}$ tal que dicho ordenamiento cumpla con alguna propiedad jerárquica.

El problema de ordenamiento puede resolverse asignando un orden r_i a cada uno de los elementos del conjunto A. Así el conjunto de

soluciones factibles puede tomarse como el conjunto de vectores n_dimensionales:

$$F = \{\bar{x} \mid x_i \in A \text{ y } x_i \neq x_j \text{ si } i \neq j\}$$

donde x_i es el elemento con orden i.

Las posibles formas de ramificar el conjunto F para obtener un ordenamiento deseado, se presentan a continuación.

1. RAMIFICACION ORDENADA.

Sea $R = \{r_1, \ldots, r_p\}$ un subconjunto de $\{1, 2, \ldots, n\}$ que contiene p elementos distintos. Un nodo del árbol de búsqueda se determina al imponer las restricciones:

$$x_{r_1} = a_{j_1},\ x_{r_2} = a_{j_2}, \ldots,\ x_{r_p} = a_{j_p}$$

Así el nodo actual especifica al conjunto:

$$F' = \{\bar{x} \mid \bar{x} \in F \text{ y } x_{r_1} = a_{j_1},\ x_{r_2} = a_{j_2}, \ldots,\ x_{r_p} = a_{j_p}\}$$

El cual puede ser particionado en n-p subconjuntos ajenos F_k, al fijar el elemento que tenga orden $r(x)$ para algún $r(x) \notin R$, es decir:

$$F_k = \{\bar{x} \mid \bar{x} \in F' \text{ y } x_{r(x)} = a_k,\ k \neq j_1, \ldots, j_p\}$$

Esta regla de ramificación recibe el nombre de RAMIFICACION ORDENADA y en general la forma dada requiere de reglas explícitas de selección, para determinar $r(x)$ cuando x es conocida.

Así en la práctica $r(x)$ se toma como una función que depende sólo del nivel de x (esto es p). Por ejemplo $r(x)=p+1$ fue usada en la solución 2 del problema 2.2.3. de la sección anterior y $r(x)=n-p$ a veces es usada.

La figura 8 muestra un árbol de enumeración completa para n=4 y A={a,b,c,d} generado por ramificación ordenada.

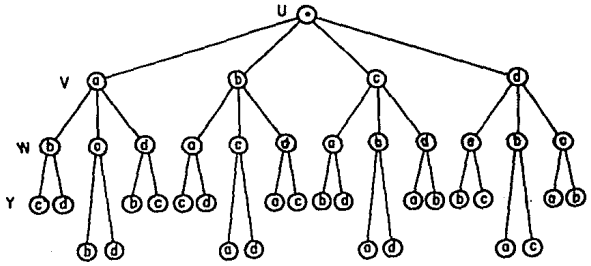


FIGURA 8

Un subconjunto de soluciones correspondientes a un nodo particular es ubicado en el árbol, dependiendo de la regla de selección usada. Por ejemplo considérese los nodos marcados U,V,W y Y por los niveles 0,1,2 y 3 respectivamente, entonces Y corresponde a la solución:

(b,a,d,c) si r(U)=2 r(V)=1 y r(W)=4.
(a,b,c,d) si r(α)= (nivel de α) + 1 para toda α.
(d,c,b,a) si r(α)= n-(nivel de α) para toda α.

2. RAMIFICACION INMEDIATA DEL SUCESOR.

Un Conjunto F' es especificado por p relaciones de la forma $r_j=r_i+1$ (que puede interpretarse como el elemento a_j es ordenado inmediatamente después del elemento a_i). F' entonces puede particionarse en dos conjuntos:

$$F'(kl)=\{\bar{\alpha}|\bar{\alpha}\in F' \text{ y } r_i=r_k+1\}$$

$$\overline{F'(kl)}= F'-F'(kl) = \{\bar{\alpha}|\bar{\alpha}\in F' \text{ y } r_i\neq r_k+1\}$$

Y como podemos ver, se requiere una regla de selección para elegir el par de índices k y l. La figura 9 da un ejemplo de un árbol de enumeración completa para n=4 y A={a,b,c,d}, la ramificación comienza al considerar si el elemento a es ordenado inmediatamente encima del elemento b o no, esto es ab o $\overline{ab}$.

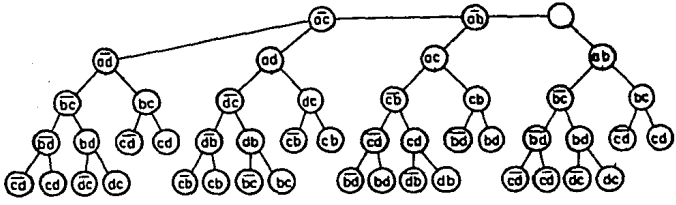


FIGURA 9

3. RAMIFICACION DEL SUCESOR.

Un conjunto F' se define como un conjunto de p relaciones $r_j>r_i$ (que se interpreta como un elemento a_j es ordenado posteriormente a un elemento a_i). F' puede entonces particionarse en dos conjuntos:

$$F'(kl)=\{\bar{\alpha}|\bar{\alpha}\in F' \text{ y } \alpha_l>\alpha_k\}$$

$$F'(lk)= F'-F'(kl) = \{\bar{\alpha}|\bar{\alpha}\in F' \text{ y } \alpha_k>\alpha_l\}$$

Nuevamente se requiere de una regla de selección para escoger la pareja (k,l), k≠l. La regla de ramificación del sucesor se usó en la solución del problema 2.2.3., la figura 10 da un ejemplo del árbol de la enumeración completa, en este caso se comienza por considerar que el elemento a se encuentre en un orden mayor que el elemento b (nodo ab) o no se encuentre en un orden mayor que b (nodo ba), si no a no está ordenado encima de b entonces se prueba con el elemento c ordenado encima del elemento a y así sucesivamente hasta descartar todas las combinaciones y encontrar un ordenamiento único.

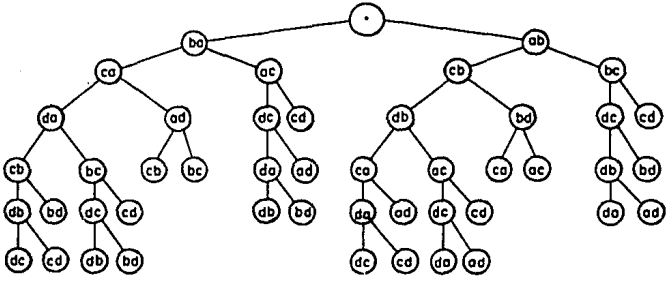


FIGURA 10

2.3.2. Reglas de Acotamiento.

Siempre que se quiera resolver un problema discreto mediante la técnica de RyA, es necesario desarrollar un árbol de búsqueda. Sin embargo para desarrollar un "buen" árbol (en el sentido de que proporcione una solución óptima de manera ágil y eficiente) se necesita adicionalmente escoger una regla adecuada de acotamiento, dicha regla a menudo es más fácil elegirla cuando en lugar de

trabajar con el conjunto F de soluciones factibles, se trabaja con un conjunto C que es una relajación de F (por ejemplo en programación entera, en lugar de trabajar con el conjunto Z se trabaja con el conjunto relajado R, lo que facilita el cálculo de las cotas ya que es más fácil encontrar óptimos de funciones reales, que de funciones con valores enteros).

Una función de Acotamiento es una función

$$g : 2^C \longrightarrow \mathbb{R}$$

C=Conjunto relajado de soluciones factibles.

2^C=Conjunto de todos los posibles subconjuntos de C.

tal que asigna un número real $g(\mathcal{X})$ a cada $\mathcal{X} \subseteq C$, cumpliendose que

$$g(\mathcal{X}) \leq f(\alpha) \qquad \text{para todo } \alpha \in \mathcal{X} \cap F$$

F=Conjunto de todas las soluciones factibles.

(nótese que todas son cotas inferiores, ya que estamos trabajando con un problema de minimización). Además esta función de acotamiento debe satisfacer que

$$g(\{\alpha\}) = f(\alpha) \qquad \text{para toda } \alpha \in F \qquad \ldots(2.2.1)$$

EJEMPLO (Algunas formas de acotar la función objetivo en un problema de asignación)

Cuatro hombres están disponibles para realizar cuatro tareas. Supongamos que el hombre i tiene una clasificación U_{ij} hacia la indeseabilidad de realizar la tarea j. Se requiere encontrar una asignación que minimize las calificaciones totales.

HOMBRES	TAREAS				Mínimo por renglón
	1	2	3	4	
1	15	5	9	5	5
2	17	14	4	7	4
3	9	13	13	7	7
4	13	10	6	15	6

SOLUCION.

Sea $F=\{\bar{\alpha}|\bar{\alpha} = (1,j,k,l)\ 1,j,k,l\in\{1,2,3,4\}\}$ el conjunto de todas las posibles combinaciones de tareas asignadas a los hombres (1,2,3,4) respectivamente. El problema de asignación lineal es:

Minimizar $U = U_{1i} + U_{2j} + U_{3k} + U_{4l}$
donde (i,j,k,l) son las permutaciones de (1,2,3,4)

Un problema relajado C es aquél en las que las restricciones son relajadas a $i,j,k,l\in\{1,2,3,4\}$. Claramente F=C, en este caso.

Ejemplos de cotas son:

$$g(C) = \min_{1\le\alpha\le4} U_{1\alpha} + \min_{1\le\beta\le4} U_{2\beta} + \min_{1\le\gamma\le4} U_{3\gamma} + \min_{1\le\delta\le4} U_{4\delta}$$

$$= 5 + 4 + 7 + 6 = 22$$

$$g(C[i=3]) = U_{i3} + \min_{1\le\beta\le4} U_{2\beta} + \min_{1\le\gamma\le4} U_{3\gamma} + \min_{1\le\delta\le4} U_{4\delta}$$

$$= 9 + 4 + 7 + 6 = 26$$

$$g(C[i=3],[j=2]) = U_{13} + U_{22} + \min_{1\le\gamma\le4} U_{3\gamma} + \min_{1\le\delta\le4} U_{4\delta}$$

$$= 9 + 14 + 7 + 6 = 36$$

donde C[i=3] denota el subconjunto de C con i fija en 3.

Pueden obtenerse mejores cotas para las asignaciones cuyo primer elemento es 3 (la tercera tarea es asignada al primer trabajador), ya que si i=3 entonces podemos insistir en que $\beta,\gamma,\delta\neq3$. De este modo se obtienen los valores:

$$g'(C) = g(C) = 22$$

$$g'(C[i=3]) = U_{13} + \min_{\beta=1,2,4} U_{2\beta} + \min_{\kappa=1,2,4} U_{3\gamma} + \min_{\delta=1,2,4} U_{4\delta}$$

$$= 9 + \min\{17,14,7\} + \min\{9,13,7\} + \min\{13,10,15\}$$

$$= 9 + 7 + 7 + 10 = 33$$

$$g'(C[i=3],[j=2]) = U_{13} + U_{22} + \min_{\gamma=1,4} U_{3\gamma} + \min_{\delta=1,4} U_{4\delta}$$

$$= 9 + 14 + \min\{9,7\} + \min\{13,15\}$$

$$= 43$$

obsérvese que g' satisface la ecuación 2.2.1. y además $g'(\mathcal{X})\ge g(\mathcal{X})$ para todo $\mathcal{X}\subset C$ (se dirá entonces que g' es más fuerte que g).

El proceso es continuado, al ramificar el nodo que tenga la menor cota (esta es la forma de desarrollar el árbol, pero no es la única) y cada vez que se encuentre una mejor solución (la solución actual), debemos checar para cada nodo $\mathcal{X}$ (esto es $\mathcal{X}\subset C$) si $g(\xi)\le g(\mathcal{X})$, en cuyo caso el nodo $\mathcal{X}$ se declara explorado por no merecer mayor consideración ya que $g(\xi)\le g(\mathcal{X})\le f(\alpha)$ para toda $\alpha\in\mathcal{X}\cap C$. Sin embargo, la solución completa del ejemplo no es presentada,

ya que en este capítulo, únicamente se presentan algunas técnicas generales para ramificar y acotar los nodos de un árbol, sin mostrar el detalle del proceso completo. Si el lector está interesado en verificar dicho proceso, se le sugiere revisar la sección 3.4. (algoritmo de asignación).

2.2.3. Reglas de Desarrollo (formas de elegir el nodo a ramificar).

1. BUSQUEDA POR PRIMERA PROFUNDIDAD.

Una regla común de desarrollo es la estrategia de ramificación de primera profundidad del nodo activo recientemente creado (un nodo $\mathcal{Y}$ se dice que es activo si $g(\xi) \geq g(\mathcal{Y})$ donde ξ es la solución actual al problema de minimización). Esta regla fuerza al árbol a un nodo terminal de manera rápida, aunque esto podría requerir más cálculos computacionales, sin embargo, reduciría el tamaño de memoria suficiente para acumular problemas correspondientes a cada nodo en una ruta de la raíz al nodo terminal del árbol.

2. PRIMER ALCANCE.

Es la regla de desarrollo aplicada en la programación dinámica y es la opuesta a la de primera profundidad. La estrategia consiste en la ramificación del mínimo nodo activo y que más recientemente fue creado.

3. COSTO UNIFORME.

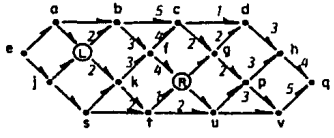
Esta regla consiste en escoger un nodo $\mathcal{Y}$ del árbol parcialmente expandido, para el cual

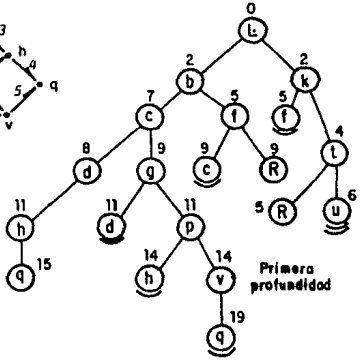
$$g(\mathcal{Y}) \leq g(\mathcal{X}) \quad \text{para todo } \mathcal{X} \text{ nodo del árbol.}$$

y los empates se resuelven por alguna regla subordinaria.

La estrategia de costo uniforme, en ocasiones llamada ramificación de la menor cota, es la más eficiente, en el sentido de que en general requiere un mínimo número de nodos a ser desarrollados. Aunque a menudo produce árboles pequeños y requiere de poco tiempo de cómputo, sufre el defecto de producir una solución fáctible cerca del fin de los cálculos.

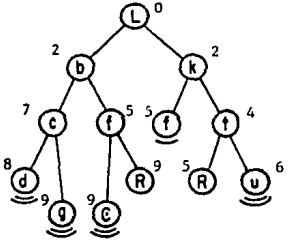
Las 3 estrategias se describen en los árboles de búsqueda de la figura 11, desarrollados para encontrar la ruta más corta de L a R.





Inramificable por dominacia
(Una ruta corta al nodo ya fué encontrada)

Inramificable por acotamiento



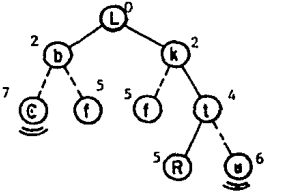


FIGURA 11

En algunas aplicaciones de Ramificación y Acotamiento puede presentarse el caso en que aprovechando la estructura del problema particular se permita comparar un par de conjuntos $\mathcal{X}, \mathcal{Y} \subset C$ y decir que la mejor solución en $\mathcal{X}$ es al menos tan buena como la mejor solución en $\mathcal{Y}$ (ver algoritmo de secuenciación, sección 3.5.). En tal caso se dice que $\mathcal{X}$ domina a $\mathcal{Y}$ y entonces el nodo $\mathcal{Y}$ en el árbol no necesita desarrollarse (si $\mathcal{X}$ domina a $\mathcal{Y}$ y $\mathcal{Y}$ domina a $\mathcal{X}$, uno de ellos puede ser inactivo -caso de empates-).

Finalmente es necesario hacer notar que, aún cuando las técnicas de ramificación y acotamiento son exactas en el sentido que garantizan soluciónes óptimas, tienen sin embargo la desventaja de involucrar una cantidad considerable de cálculos. Pero siempre que se tenga la flexibilidad suficiente para aceptar una solución subóptima, es mejor usar ramificación y acotamiento "heurísticamente", ya que con este tipo de algoritmos se logra reducir el tiempo de proceso, sólo que obteniendo en algunos casos soluciones infactibles o soluciones factibles pero no óptimas (cuando la estratégia es aceptar la mejor solución fáctible, generada en un tiempo dado).

2.4. Discusion

En el presente capítulo se mostro de manera general, el funcionamiento de la técnica de RyA, así como las reglas de ramificación y acotamiento más comúnmente utilizadas cada vez que se requiere resolver un problema del tipo discreto. Y aunque no existe un proceso bien especifico de cómo resolver un problema discreto mediante la técnica de RyA, podemos decir que los pasos siguientes engloban su funcionamiento general.

1.- Siempre que sea conveniente, debemos tratar de relajar el conjunto de soluciones factibles a uno donde sea más fácil identificar una solución y de esta forma sondear el lugar donde se encuentra una solución al problema original.

2.- Separar (ramificar) el conjunto de soluciones en dos o más conjuntos ajenos (con el propósito de eliminar subconjuntos en los que aseguraremos que no pueden contener una solución óptima) mediante alguna regla que permita llegar de manera más rápida a la especificación de una posible solución. aunque esto es realmente subjetivo, puesto que la rápidez con que encontremos una solución óptima no radicará sólo en la regla de ramificación si no también en saber sobre qué nodo hacer la separación y ádemas en el obtener una buena regla de acotación para cada nodo explorado.

3.- Acotar inferiormente (superiormente) cada nodo ramificado si el problema es del tipo minimizar (maximizar) con la finalidad de eliminar aquellos nodos que no pueden contener una solución óptima, ya sea debido a que es infactible o bien porque no tiene una mejor cota que la actual solución fáctible (si es que ya se encontró). Entre más nodos declaremos inactivos, más pronto obtendremos una solución óptima.

4.- Escoger el nodo sobre el cual se realizará la siguiente ramificación y aunque se sugiere considerar aquel nodo que tiene la menor cota inferior, a menudo conviene considerar otras formas de elección como la de "búsqueda por primera profundidad" la que permite cada vez declarar mayor número de nodos inactivos o bien utilizar "primer alcance" o "costo uniforme". Aún cuando en realidad ninguna (al menos como estrategia) tiende a ser favorable.

5.- Repetir pasos dos a cuatro, hasta terminar el análisis con todos los nodos creados.

3. ALGORITMOS ESPECIFICOS

3.1. INTRODUCCION

En el presente capitulo veremos tres algoritmos basados en la técnica de RyA y que permiten resolver problemas de optimización en los qué el conjunto de soluciones factibles es del tipo discreto. Uno de estos algoritmos es el **algoritmo de la enumeración implicita**, el cual permite resolver el problema de programación lineal binaria (cero-uno) que en su forma general viene dada por:

$$\text{Minimizar} \quad Z = \sum_{j=1}^{n} C_j X_j$$

$$\text{sujeto a} \quad \sum_{j=1}^{n} a_{ij} X_j \leq b_i \qquad i=1,2,\ldots,m$$

$$X_j = 0 \text{ ó } 1 \qquad j=1,2,\ldots,n.$$

Entre los problemas de optimización que se ajustan a este modelo se encuentran: El problema de la mochila, el problema de la selección de proyectos de inversión, el problema de nivelar una línea de ensamblado, el problema del cubrimiento y el de balance de líneas de producción, entre otros. Y aún cuando es eficiente resolver un problema binario general, mediante la enumeración implícita, esto no es comparable con la eficiencia que se logra obtener, cuando algunos problemas específicos son resueltos con algoritmos específicos, como son los algoritmos de asignación y de secuenciación. Algoritmos basados, todos, en la técnica de RyA y que a continuación veremos, claro, analizando también la eficiencia de los mismos (ver sección 3.6. -complejidad computacional-).

Adicionalmente se incluye una parte de análisis de postoptimalidad para el algoritmo de enumeración implícita, que es muy útil siempre que se necesite resolver un mismo problema, salvo por pequeños cambios en los valores de algunos parámetros y sin la necesidad de reformular y resolver todo el problema desde el principio.

3.2. ALGORITMO DE ENUMERACION IMPLICITA

IDEAS BASICAS:

Considere el problema lineal cero-uno en su forma general:

$$\begin{aligned} Min \quad & Z = C^{*}\overline{X} \\ \text{sujeto a} \quad & \\ & A^{*}\overline{X} \leq b \\ & \overline{X} = 0,1. \end{aligned}$$

Es conveniente suponer que el vector de costos C es no-negativo, pues en caso que asuma valores no-negativos, bastará substituir a X_j por $1-X_j'$, cada vez que $C_j<0$, ya que en estas condiciones, si X_j toma los valores 0 ó 1, entonces X_j' toma los valores 1 ó 0 y la

función objetivo queda expresada por:

$$z = \sum c_j x_j = \sum c_j^- x_j + \sum c_j^+ x_j$$

$$= \sum c_j^-(1-x_j') + \sum c_j^+ x_j$$

$$= \underbrace{-\sum c_j^- x_j' + \sum c_j^+ x_j}_{\text{costos} \geq 0} + \underbrace{\sum c_j^-}_{\text{constante}} \qquad c_j^- < 0 \text{ y } c_j^+ \geq 0$$

en donde, la constante $\sum c_j^-$ puede ser eliminada de la función objetivo, ya que no influye en el proceso de optimización.

Una vez hecha esta transformación, el proceso de optimización se basa en un método de búsqueda, el cual trabaja con la enumeración implícita de los 2^n posibles vectores $\bar{X}$, con entradas cero-uno. El procedimiento enumerativo es representado por un árbol de búsqueda, compuesto de nodos y ramas. Un nodo corresponde a una solución candidato cero o uno y dos nodos conectados por una rama, difieren en el estado de una variable. Cada variable puede tener uno de los tres estados siguientes: fija en 1, fija en 0 ó libre. Un nuevo nodo es definido, cuando una variable es fijada en 1 (forward) y un nodo es revisado si una variable es fijada en 0 (backward).

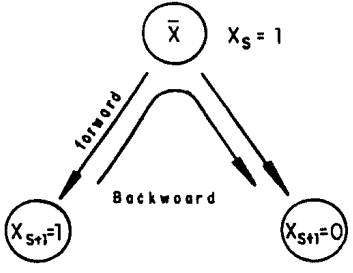


FIGURA 12

Así el problema en un nodo intermedio, es de la forma :

$$\textit{Minimizar}\quad Z \sum_{j\in J} C_j X_j + \sum_{j\in J^1} C_j$$

sujeto a

$$\sum_{j\in J} a_{ij} X_j \le Y_i$$

$$X_j \in \{0,1\}\ j\in J$$

donde $Y_i = b_i - \sum_{j\in J^1} a_{ij}$

$J=\{j \mid X_j$ es libre respecto al nodo actual$\}$

$J^1=\{j \mid X_j$ es fijada en 1 hasta el nodo actual$\}$

Nótese que no hubo necesidad de invertir matrices o almacenar coeficientes. Tanto las a_{ij} como las b_i son los coeficientes originales. Esto por sí sólo reduce el tiempo de cálculo y evita errores numéricos, además de almacenar menor cantidad de información.

Cada nodo entonces será ramificado cuando se fije una variable en 1, tratando de hacer que los términos Y_i tiendan a ser positivos (pues si todos fueran positivos, la enumeración hasta el nodo actual, sería fáctible y cualquier enumeración posterior a ésta, arroja un mayor valor de la función objetivo, siendo inútil cualquier enumeración posterior) se continúa este proceso de ramificación hasta encontrar un nodo en el que la enumeración sea factible (cuando $Y_i \ge 0$ para $i=1,2,\ldots,m$) o bién cuando sea infactible (debido a que ninguna enumeración posterior podrá hacer que el término Y_i sea positivo); en tal caso es necesario regresar a un nodo anterior (aquél que fue creado más recientemente).

CRITERIOS DE LA RAMIFICACION Y ACOTAMIENTO

1. Sobre la función Objetivo

El propósito de esta prueba, es poder identificar y desechar aquellos nodos, que al tratar de hacerlos factibles ($Y_i<0$ en el actual nodo) y siendo necesaria una enumeración posterior con aquellas variables en las que $a_{ij}<0$ (con el propósito de reducir la infactibilidad), no podamos llegar a una solución en la cual obtengamos un mejor valor para la función objetivo. Para esto, creamos el conjunto T de variables tales que:

$$T = \{j \mid X_j \text{ es libre, } Z+C_j<\hat{Z}\ ,\ a_{ij}<0,\ Y_i<0\}$$

donde:

$J^1 = \{j \mid X_j$ es fijada en 1 en la solución actual$\}$

$Z=\sum_{j\in J^1} C_j$

$\hat{Z}$ = mejor valor de la función objetivo hasta ahora encontrada.

Cuando T=ø entonces ninguna enumeración posterior, que reduzca la infactibilidad, podrá mejorar el actual valor de la función objetivo $\hat{Z}$ y por lo tanto es necesario regresar a un nodo anterior (que será el más recientemente creado).

2. Prueba de Infactibilidad

a.- Si T≠ø, es posible identificar cuando ninguna enumeración posterior, al nodo actual, conducirá a una solución factible, pues en el caso que exista un índice i tal que:

$$Y_i < 0 \quad y \quad Y_i - \sum_{j\in T} \min\ (0, a_{ij}) < 0$$

indudablemente, en el nodo actual se tendría que, la i-ésima restricción permanecería infactible, aún cuando todas las variables de T tuvieran valor 1. Así que, cuando el nodo satisface la prueba de infactibilidad, debe declararse como inactivo y la búsqueda debe ser continuada sobre el nodo más recientemente creado (realizar el backward).

b.- Verificar si

$$\sum_{j \in J} \min (o, a_{ij}) + |a_{ip}| > Y_i$$

si es el caso, continuar la búsqueda en el nodo actual, con la rama $\mathcal{X}_p=0$ ($\mathcal{X}_p=1$) si $a_{ip}>0$ ($a_{ip}<0$). Ya que, si la prueba es cumplida en el nodo actual (llamémosle S) y $a_{ip}>0$, la nueva rama asignada con $\mathcal{X}_p=1$, sería infactible, ya que:

$$\sum_{j \in J^{s} \cup \{\mathcal{X}_p=1\}} \min (0, a_{ij}) = \sum_{j \in J^{s}} \min (0, a_{ij}) > Y_i - a_{ip} = Y_i^{s \cup \{\mathcal{X}_p=1\}}$$

donde

$J^{s \cup \{\mathcal{X}_p=1\}}$ = $\{\mathcal{X}_k | \mathcal{X}_k=0$ ó 1, según el trayecto hasta S$\}$
$\cup$ $\{\mathcal{X}_p$ que en el momento se le asigna el valor 1$\}$

s = nodo actual.

Por lo que el nodo $\mathcal{X}_p=1$, a partir de S, no puede dar una solución factible y por lo tanto, es necesario considerar $\mathcal{X}_p=0$ en cualquier sucesor de S (lo mismo ocurre cuando la prueba $a_{ip}<0$ y la prueba de infactibilidad es satisfecha).

3. Prueba de Ramificación de Balas

El seleccionar las variables libres que serán fijadas en 1, puede

influir significativamente en la eficiencia de algoritmo. Esto particularmente es cierto en el principio, donde una selección "pobre" de alguna variable libre, puede resultar en la enumeración innecesaria de un gran número de nodos y ramas que al final del proceso, diferirán en mucho de la solución óptima. Una buena regla, diseñada para ir en la búsqueda directa de una solución, ha sido propuesta por Balas.

A cada variable libre x_j se le asocia el conjunto M_j

$$M_j = \{i \mid y_i - a_{ij} < 0\}$$

y calculamos

$$V_j = \sum_{i \in M_j} (y_i - a_{ij})$$

Si $M_j = \emptyset$, se define $V_j = 0$

Y entonces determinamos aquella variable libre, que al asignarle el valor 1, reduce en "total" las infactibilidades. Por "total" se entiende como la suma de todas las cantidades por las que cada una de las restricciones son violadas. Entonces debe seleccionarse aquella variable x_j que maximiza V_j (es decir la que más reduce en "total" las infactibilidades).

En caso de que $M_j = \emptyset$ para todo j, entonces ninguna enumeración posterior es necesaria, ya que la mejor de ellas se obtiene cuando a todas las variables libres se les asigna el valor cero (solución que además resulta ser factible); por lo que debemos regresar a un nodo anterior y continuar el proceso.

ALGORITMO ADITIVO CERO-UNO DE BALAS

Para registrar la trayectoria de la enumeración progresiva utilizaremos los vectores $\bar{U} = (u_1, u_2, \ldots, u_n)$ y $\bar{X} = (x_1, x_2, \ldots, x_n)$.

Así cuando las variables $x_{j_1}, x_{j_2}, \ldots, x_{j_s}$ son asignadas en ese orden, las componentes de $\bar{X}$ correspondiente a las variables asignadas son 0 ó 1 y las restantes variables libres tendrán valor -1. Las componentes de $\bar{U}$ son:

$$U_k = \begin{cases} J_k \text{ si } x_{j_k} = 1 \text{ y su complemento aún no ha sido considerado.} \\ -J_k \text{ si } x_{j_k} = 1 \text{ ó } 0 \text{ y su complemento ya fue considerado.} \\ 0 \text{ si } K > S. \end{cases}$$

y $\hat{Z}$ será el mejor valor de la función objetivo hasta el momento encontrado.

Paso Inicial.

PASO 1. Verificar si $b \geq 0$. Si es el caso, entonces la solución óptima es $\hat{X} = (0,0,\ldots,0)$. En otro caso, inicializar $\bar{U}=(0,0,\ldots,0)$, $\bar{X} = (-1,-1,\ldots,-1)$ y $\hat{Z}=\infty$ (un número suficientemente grande).

Pasos Iterativos.

Paso 2. Calcular

$$Y_i = b_i - \sum_{j \in J^1} a_{ij} \qquad i=1,2,\ldots,m$$

$$\hat{Y} = \min Y_i$$

$$Z_0 = \sum_{j \in J^1} C_j$$

Donde $J^1 = \{j \mid x_j$ es fijada en 1 hasta el nodo actual$\}$

Paso 3. Si $\hat{Y} \geq 0$ y $Z_0 < \hat{Z}$ entonces, asignar $\hat{Z}=Z_0$ y $\hat{X}=\bar{X}$. De este modo,

el vector $\hat{X}$ es la nueva solución encontrada. Ir al paso 7 (*backward*).

- Si $Z > \hat{Z}$ ir al paso 7.
- En otro caso continuar.

Paso 4. Crear el conjunto T de variables libres $\mathcal{X}_j$ definido como sigue:

$$T = \{j \mid Z+C_j<\hat{Z},\ a_{ij}<0,\ Y_i<0\}$$

- Si T=ø entonces ir al paso 7.
- En otro caso continuar.

Paso 5. PRUEBA DE INFACTIBILIDAD.

(a). Prueba si hay un índice i tal que $Y_i<0$ y $Y_i-\sum_{j\in T} \min(0,a_{ij}) < 0$

si es el caso, ir al paso 7. En otro caso continuar.

(b). Prueba si hay un índice i tal que

$$\sum_{j\in J} \min(0,a_{ij}) + |a_{ij}| > Y_i,$$

Si es el caso, declarar infactible el nodo $\mathcal{X}_j=0$ ($\mathcal{X}_j=1$) si $a_{ij}<0$ ($a_{ij}>0$) y continuar la ramificación de S con el nodo $\mathcal{X}_j=0$ ($\mathcal{X}_j=1$) si $a_{ij}>0$ ($a_{ij}<0$). Modificar el vector $\bar{U}$, asignando $U_k = -J_k$ a la componente 0 que está más a la derecha.

Paso 6. PRUEBA DE RAMIFICACION-BALAS.

Para cada variable libre $\mathcal{X}_j$ crear el conjunto Mj como:

$$M_j=\{i \mid Y_i-a_{ij}<0\}$$

Si Mj=ø para todo j∈J; ir al paso 7.

En otro caso, calcular el valor V_j para cada variable $\mathcal{X}_j$

$$V_j = \sum_{i \in M_j} (Y_i - a_{ij})$$

Donde $V_j=0$ si $M_j=\emptyset$. Ramificar el nodo actual, adicionando la variable $\mathcal{X}_j$ que maximiza V_j a la solución parcial actual. Regresar al paso 3.

Paso 7. BACKWARD.

(a). Regresar al nodo más recientemente creado.

(b). Modificar el vector $\bar{U}$. Cambiarle el signo a la componente positiva, que está más a la derecha. Asignar en cero todos los elementos a la derecha de dicha componente. Esta es la nueva enumeración parcial. Regresar al paso 3.

(c). Si no hay elementos positivos en $\bar{U}$, la enumeración implicita está completada. La solución óptima es el actual vector $\hat{X}$ y el correspondiente valor de la función es $\hat{Z}$. Si $\hat{Z}=\infty$, entonces no hay solución factible.

EJEMPLO 1

Minimizar $Z = 3\mathcal{X}_1+2\mathcal{X}_2+\mathcal{X}_3+\mathcal{X}_4$

sujeto a

$$-\mathcal{X}_1+\mathcal{X}_2+6\mathcal{X}_3+\mathcal{X}_4 \leq 5$$
$$-\mathcal{X}_1-2\mathcal{X}_2+3\mathcal{X}_3-\mathcal{X}_4 \leq -3$$
$$2\mathcal{X}_1+2\mathcal{X}_2-\mathcal{X}_3-8\mathcal{X}_4 \leq -6$$
$$\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3, \mathcal{X}_4 = 1 \text{ ó } 0.$$

Inicialmente $\bar{U}=(0,0,0,0)$ $\bar{X}=(-1,-1,\ldots,-1)$, $\hat{Z}=999$

Iteración 1

La Prueba de infactibilidad (b) es satisfecha para la restricción 2 y 3.

$$\sum_{j\in J} \min(0, a_{2j}) + |a_{22}| > Y_2$$

es decir: $\quad -4 + |2| > -3$

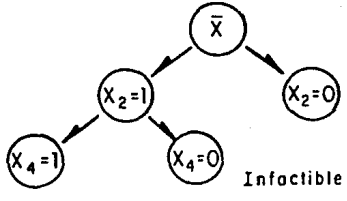
y $$\sum_{j\in J} \min(0, a_{3j}) + |a_{34}| > Y_3$$

es decir: $\quad -9 + |-8| > -6$

por lo que deben considerarse solamente las ramas $X_2=1$ y $X_4=1$ a partir del nodo actual.

Iteración 2

$\bar{U}=(-2,-4,0,0)$
$\hat{Z}=3$



S = Nodo actual

FIGURA 13

El problema en el nodo S es

$$\textit{Minimizar } Z=3X_1+X_3+3$$

sujeto a

$$-X_1+6X_3 \leq 3$$
$$-X_1+3X_3 \leq 0$$
$$2X_1- X_3 \leq 0$$
$$X_1,\ X_3 = 1 \text{ ó } 0.$$

Debido a que ninguna enumeración posterior a S puede arrojar una mejor solución, es necesario hacer el *backward* a un nodo anterior. Y como todos los nodos anteriores arrojan una solución infactible, se concluye que la solución óptima es:

$$X_1=0,\ X_2=1,\ X_3=0,\ X_4=1 \quad y \quad \hat{Z}=3.$$

EJEMPLO 2 (Solución al problema de los proyectos de inversión)

Para el problema del capítulo 1 referente a la selección de 7 proyectos u oportunidades de inversión, se aplicó el algoritmo de Balas, pero debido a la cantidad enorme de posibles nodos a ramificar (2^{17}=131,072 posibles combinaciones) no se presenta el desarrollo detallado del árbol de búsqueda y en lugar de ello se muestra la solución obtenida en un programa computacional basado en el algoritmo de Balas y que consume aproximadamente 20 minutos de proceso para una computadora Acer/PC con procesador 80286.

Los resultados de la aplicación del programa se presentan en la hoja siguiente y de ellos se observa que la solución óptima es:

$$\hat{X}=(\underset{X1}{1},\ \underset{X2}{0},\ \underset{X3}{1},\ \underset{X4}{0},\ \underset{X5}{0},\ \underset{X6}{0},\ \underset{X7}{1},\ \underset{X8}{1},\ \underset{X9}{0},\ \underset{X10}{1},\ \underset{X11}{0},\ \underset{X12}{1},\ \underset{X13}{0},\ \underset{X14}{0},\ \underset{X15}{0},\ \underset{X16}{0},\ \underset{X17}{0}),\ \hat{Z}=50$$

Por lo cual se decide invertir en los proyectos 1, 3 y 7 para la

primera etapa y en el proyecto 1 para su segunda etapa, gastando en la primera etapa 25+45+25=95 millones de pesos y sobrando 5 millones sin invertir ($h = 1*X_{10} + 4*X_{12}$, ya que $X_{10}=X_{12}=1$). Para la segunda etapa se dispone de 25+9=34 millones que se liberan del proyecto 1 en su primera etapa, más 18+45=63 millones que se liberan del proyecto 3, más 5 millones que sobraron de los 100 millones disponibles para la primera etapa; así que al inicio de la segunda etapa se dispone de 34+63+5=102 millones; cantidad que sí permite invertir en el proyecto 1 para su segunda etapa (se necesitan 100 millones para invertir en este proyecto), pero no para la segunda etapa del proyecto 2, ya que para ello se necesitarían 160 millones.

```
PROGRAM: Zero One Programming

##### INPUT DATA ENTERED #####

Max  Z =  9 x 1 + 16 x 2 + 18 x 3 + 15 x 4 + 6 x 5 + 12 x 6 + 11 x 7 + 12 x 8 +
15 x 9

Subject to:

C 1   25 x 1 + 40 x 2 + 45 x 3 + 35 x 4 + 15 x 5 + 30 x 6 + 25 x 7 + 1 x 10 + 2
x 11 + 4 x 12 + 8 x 13 + 16 x 14 + 32 x 15 + 64 x 16 =  100
C 2   1 x 1 + 1 x 2 =  1
C 3  - 1 x 1 + 1 x 8 =  0
C 4  - 1 x 2 + 1 x 9 =  0
C 5   34 x 1 + 63 x 3 + 50 x 4 + 1 x 10 + 2 x 11 + 4 x 12 + 8 x 13 + 16 x 14 + 3
2 x 15 + 64 x 16 + 200 x 17 >=  100
C 6   56 x 2 + 63 x 3 + 50 x 4 + 1 x 10 + 2 x 11 + 4 x 12 + 8 x 13 + 16 x 14 + 3
2 x 15 + 64 x 16 - 200 x 17 >= -40

#####  PROGRAM OUTPUT  #####

Iteration :  44296
Assigned Variables:  1 -1  1 -1  1 -1 -1  1 -1  1  1  1  1  0  0  0  0
The value of this combination is  45

Iteration :  47415
Assigned Variables:  1 -1  1 -1 -1 -1  1  1 -1  1 -1  1  0  0  0  0  0
The value of this combination is  50

Iteration :  48473
Assigned Variables:  1 -1  1 -1 -1 -1 -1  1 -1 -1  1  1  1  1  0  0  0
The value of this combination is  39

Iteration :  52530
Assigned Variables:  1 -1 -1  1  1 -1 -1  1 -1  1 -1 -1  1  1  0  0  0
The value of this combination is  42

Iteration :  56753
Assigned Variables:  1 -1 -1  1 -1 -1 -1  1 -1 -1 -1 -1  1 -1  1  0  0
The value of this combination is  36

Iteration :  64755
Assigned Variables:  1 -1 -1 -1 -1 -1 -1 -1  1 -1  1  1 -1  1 -1 -1  1  0
The value of this combination is  21

The optimal solution is 50

The solution values of decision variables are as follows:

 1  0  1  0  0  0  1  1  0  1  0  1  0  0  0  0  0
```

3.3. ANALISIS DE POSTOPTIMALIDAD

Para poder hacer más fácil el análisis de postoptimalidad, para un problema de programación binaria, dos modificaciones básicas al algoritmo de enumeración implícita son necesarias

(a) Eliminación del paso 4.
(b) Para la prueba de infactibilidad (inciso 5°.) cambiar el término:

$$Y_i - \sum_{j \in T} \min(0, a_{ij}) < 0$$

Por:

$$Y_i - \sum_{j \in J} \min(0, a_{ij}) < 0$$

De este modo, la inactividad de un nodo, es única y exclusivamente por violación a una restricción o bien porque $Z > \hat{Z}$, pero no ambas.

Esto por un lado facilita el análisis de Postoptimalidad, pero por otro, hace menos eficiente el algoritmo de enumeración implícita, en el sentido que tarda más tiempo en encontrar la solución óptima al problema binario (aunque esto puede no ser tan importante, si deseamos analizar de manera rápida el cambio de la solución óptima, bajo diversas alteraciones en el valor de los parámetros, sin necesidad de replantear completamente el problema).

IDEAS BASICAS

La simplicidad de las pruebas para declarar un nodo inactivo (paso 3 y 4) hace posible que, cada vez que un nodo es infactible, a éste se le asocie el número de alguna restricción, la cual no es satisfecha en el nodo analizado.

Así por ejemplo, si al sondear una solución parcial S, existe infactibilidad causada por la restricción K (debido a que $Y_K - \sum \min(0, a_{kj}) < 0$), entonces la solución parcial se declara infactible inactiva y se le atribuye la restricción K (la cual origina la infactibilidad del nodo). En el caso que la solución parcial S se declara inactiva, debido a la prueba del paso 3, esta inactividad es causada por la función objetivo, ya que el valor de la función objetivo de la mejor complementación de S es mayor que $\hat{Z}$.

Sea Γ el conjunto de soluciones sondeadas al resolver el problema cero-uno. Tan grande como pueda ser la enumeración, ésta es exahustiva y no redundante, así cada solución al problema será la complementación de algún miembro preciso de Γ. Esto se sigue de que cualquier partición de Γ induce una partición de las 2^n soluciones. En este marco, el proceso de Postoptimalidad hace uso de este hecho, procediendo en dos fases. En la primera, una partición de las 2^n soluciones es conseguida cuando se genera Γ, al resolver un problema cero-uno especialmente construido (el de mayor "interés"). En la segunda fase, la partición Γ es utilizada para realizar el análisis de postoptimalidad.

ANALISIS DE POSTOPTIMALIDAD

I. FASE DE SOLUCION

a).- Formular y resolver (usando el algoritmo de enumeración implícita modificado) el problema:

$$\textit{Minimizar } Z = C^{+}\bar{X}$$

Sujeto a

$$A^{+}\bar{X} \leq b^{+}$$

$$X_j = 0 \text{ ó } 1.$$

donde C_j^+ y a_{ij}^+ son los mayores C_j y a_{ij} de interés respectivamente y b_i^+ es el más pequeño de los b_i de interés.

De esta forma, el reducir valores de C_j^+ y a_{ij}^+ o incrementar los de b_i^+ producirán restricciones relajadas o reducción de costos, implicando potencialmente mejores soluciones.

b).- Cada vez que una solución parcial es sondeada, debe guardarse la solución parcial y atribuirla a la restricción apropiada. Al final de la fase de solución, las soluciones parciales sondeadas de (I^+) han sido particionadas en m+1 conjuntos disjuntos. Denotemos por Γ_k al cojunto de soluciones parciales atribuidas a la restricción K, K=0,1,...,m (K=0 es asignada a la función objetivo).

$$\bigcup_{i=0}^{m} \Gamma_i = \Gamma \qquad \Gamma_i \cap \Gamma_k = \emptyset \quad i \neq k$$

c).- En caso de que el lado derecho de la restricción K (b_k) sea incrementado (involucrando una relajación en la restricción) y los demás parámetros no cambian entonces:

- La solución óptima es factible (pero no necesariamente óptima) en el problema revisado.

- Una mejor solución que la óptima previa (si ésta existe) puede ser una completación de algún $S \in \Gamma_k$. Ya que en la complementación de cualquier otro nodo terminal S, tal que $S \notin \Gamma_k$ seguirá permaneciendo infactible debido que no satisface alguna otra restricción o tendrá una cota $\bar{z}$ mayor al actual óptimo; por lo que sólo aquellos nodos en Γ_k pueden arrojar una mejor solución.

II ANALISIS DE RANGO (AR)

(a) Sobre b_k

Como se mencionó en el punto (c) de la fase de solución, es fácil comprender que si el parámetro b_k fuese incrementado y los demás parámetros no cambian; entonces la solución óptima es factible, pero posiblemente no sea el nuevo óptimo. Sin embargo en caso de que exista una mejor solución, ésta puede ser una complementación de $S \in \Gamma_k$. No obstante, aún cuando se altere el valor de b_k, existe un rango de valores alrededor de los cuales la solución óptima permanece inalterable, esto puede verse mediante la siguiente proposición.

PROPOSICION

La solución óptima S^* al problema (I^+) permanece óptima para valores de b_k en el intervalo

$$b_k^+ - S_k \leq b_k < b_k^+ + U_k$$

Donde:

S_k=la holgura de la restricción K en el actual óptimo S^*.

$$U_k = \min_{S \in \Gamma_k} \{ \rho_k^s - Y_k^s \}$$

$$\rho_k^s = \sum_{j \in J} \min (0, a_{ij}) \quad \text{según } S$$

$$Y_k^s = Y_k \quad \text{según } S$$

DEMOSTRACION

La idea básica de la demostración es tratar de probar que todos los nodos terminales $S \in \Gamma$, salvo S^*, permanecen infactibles para cambios de b_k en el intervalo ya mencionado.

Así por un lado es claro que todos los nodos $S \in \Gamma_i$, $i=0,1,2,\ldots,K-1, k+1, \ldots, m$ (salvo Γ_k) pemanecen infactibles, ya que, sólo hicimos modificaciones en la restricción K, lo que nos lleva a analizar sólo aquellos $S \in \Gamma_k$.

Sea S un nodo término y $S \in \Gamma_k$, Sabemos que al cambiar b_k^+ en la restricción K sucede que

$$Y_k^{s\,\prime} = b_k - \sum_{j \in J^1} a_{kj} < b_k^+ + (\rho_k^s - Y_k^s) - \sum_{j \in J^1} a_{kj}^+$$

(b_k es el valor revisado de b_k^+)

$$= b_k^+ - \sum_{j \in J^1} a_{kj} + \rho_k^s - Y_k^s$$

$$= Y_k^s + \rho_k^s - Y_k^s$$

$$= \rho_k^s$$

$$= \rho_k^{s\,\prime}$$

$\therefore Y_k^{s\,\prime} < \rho_k^{s\,\prime}$ y por tanto el nodo sigue siendo infactible.

Adicionalmente, para asegurar que todo $S \in \Gamma_k$ permanece infactible, bajo estas condiciones; debe considerarse como cota única, aquella que resulta ser la mínima cantidad que hace infactible todos y cada uno de los nodos $S \in \Gamma_k$. Así que considerando:

$$U_k = \min_{S \in \Gamma_k} \{\rho_k^* - Y_k^*\}$$

es seguro que todos los nodos $S \in \Gamma_k$ siguen siendo infactibles. Por otro lado para asegurar que S^* siga siendo óptima, es necesario que continúe satisfaciendose la restriccción K de la actual solución S^* cuando se presentan cambios en el parámetro b_k^+.

Así por un lado, si b_k^+ es incrementado (es decir la restricción K en S^* es relajada) entonces la restricción K continúa satisfaciendose, además ningún $S \in \Gamma_0$ podrá lograr que $Z \leq Z^*$ y S^* permanece óptima en este caso. Por otro lado, también es fácil ver que b_k^+ no puede ser decrementado en más de una cantidad igual a la holgura S_k en la restricción K (según S^*), donde:

$$S_k = b_k - \sum_{j \in J^1} a_{kj} \qquad J^1 \text{ según } S^*$$

de otra manera, dicha restricción no sería satisfecha.

(b) Sobre C_r.

Es claro que si la variable $X_r=0$ (según S^*) y su coeficiente C_r^+ fuese incrementado, entonces el actual óptimo permanece óptimo, puesto que ningun $S \in \Gamma_0$ puede dar una mejor solución y además todas las cotas Z_S (para cada $S \in \Gamma_0$) se incrementaran al incrementar C_r^+ y como Z_S excedia a la cota Z_S^*; entonces seguirá cumpliendose que $Z_S > Z_S^*$ y por tanto S^* permanece óptima.

Por otro lado si la variable $X_r=0$ (en el actual óptimo) y su coeficiente C_r^+ es reducido, siendo que todos los demás parámetros no cambian, entonces:

1.- La solución óptima puede no ser óptima para el problema revisado, pero

2.- Una mejor solución, si ésta existe, se obtiene de la complementación para algún $S \in \Gamma_0$ para el que $\mathcal{X}_r^s$ es libre o $\mathcal{X}_r^s = 1$, pues de esta forma todos los nodos $S \in \Gamma_0$, en donde $\mathcal{X}_r^s = 0$, seguirán teniendo una cota mayor que la asignada a S^*).

Sin embargo, existe un intervalo en el cual el coeficiente C_r^+ puede ser decrementado sin que cambie la actual solución óptima, y esto podemos analizarlo con la siguiente proposición.

PROPOSICION

Si $\mathcal{X}_r = 0$ en el actual óptimo S^*; entonces S^* permanece óptima para nuevos valores de C_r en el intervalo

$$C_r^+ - W_r < C_r < \infty$$

donde $W_r = \min \{ \phi^s \}$

$$\phi^s = \begin{cases} Z - Z^* + C_r^+ & \text{si } \mathcal{X}_r \text{ es libre en } S \\ Z - Z^* & \text{si } \mathcal{X}_r = 1 \text{ en } S \end{cases}$$

DEMOSTRACION

Basta probar que para decrementos de C_r^+ por arriba del valor $C_r^+ - W_r$ sigue cumpliendose que

$$Z_s \geq Z_{s^*} \quad \text{para toda} \quad S \in \Gamma_k$$

1.- Si $\mathcal{X}_r = 0$ en $S \in \Gamma_0$, se tiene que la cota Z_s no se altera al cambiar C_r^+. Y por tanto sigue cumpliendose que:

$$Z_s' = Z_s > Z_s^*$$

2.- Si $\mathcal{X}_r = 1$ en $S \epsilon \Gamma_0$, y si el nuevo C_r satisface que $C_r > C_r^* - W_r$ entonces:

$$\begin{aligned} Z_s' &= Z_s - C_r^* + C_r \\ &> Z_s - C_r^* + (C_r^* - \phi_s) \\ &= Z_s - C_r^* + C_r^* - (Z_s - Z^*) \qquad \text{pues } \mathcal{X}_r = 1 \text{ en } S \epsilon \Gamma_0 \\ &= Z^* \end{aligned}$$

$\therefore Z_s' > Z^*$

y entonces S no puede arrojar una mejor solución que S^*

3.- Si $\mathcal{X}_r$ es libre en $S \epsilon \Gamma_0$. La solución S tampoco puede arrojar una mejor solución que S^*. Ya que al ramificar S, considerando la rama $\mathcal{X}_r = 0$, el nuevo nodo seguirá teniendo una cota

$$Z'_{s \cup \{\mathcal{X}_r = 0\}} = Z_s > Z^*$$

y por tanto el nodo S es desechado. En caso de que $\mathcal{X}_r = 1$, a partir de S, tenemos que:

$$\begin{aligned} Z'_{s \cup \{\mathcal{X}_r = 1\}} &= Z_s' + C_r \\ &= Z_s + C_r \qquad \text{pues } \mathcal{X}_r \text{ es libre según S} \\ &> Z_S + C_r^* - \phi_s \\ &= Z_S + C_r^* - (Z_S - Z^* + C_r^*) \\ &= Z^* \end{aligned}$$

$$\therefore \quad Z'_{s\cup\{\mathcal{X}_r=1\}} = Z^*_s$$

con lo que las ramas $\{\mathcal{X}_r=0\}$ y $\{\mathcal{X}_r=1\}$, desde el nodo S, no pueden arrojar una mejor solución óptima.

Por tanto ningún $S\in\Gamma_0$ proporciona una mejor solución que S^* para decrementos de C^+_r que cumplan con:

$$C^+_r > C_r - W_r$$

(c) Sobre a_{kr}

El procedimiento para analizar los coeficientes de las restricciones es muy parecido al proceso seguido para los coeficientes de costo. Con respecto al análisis de rango, nosotros podemos definirlo de la siguiente forma:

PROPOSICION

La solución óptima S^* al problema (I^+) permanece óptima para los valores de a_{kr} en el intervalo

$$a^+_{kr} - V_{kr} < a_{kr} < a^+_{kr} + S_k$$

donde S_k = La cantidad de holgura en la restricción K, en el actual óptimo.

$$V_{kr} = \min\{\phi^*\}$$

$$\phi^* = \begin{cases} \rho^*_k - Y^*_k + a^+_{kr} & \text{si } \mathcal{X}_r \text{ es libre y } a_{kr} \geq 0 \\ \rho^*_k - Y^*_k & \text{en otro caso} \end{cases}$$

y el mínimo es tomado sobre todo $S\in\Gamma_k$ en los que $\mathcal{X}_r$ es libre o $\mathcal{X}^*_r=1$

DEMOSTRACION

Basta probar que para cambios de a_{kr} en el intervalo

$$a^{+}_{kr} - V_{kr} < a_{kr} < a^{+}_{kr} + S_k$$

todas las soluciones parciales $S \epsilon \Gamma_k$ permanecen infactibles y además la restricción K de S^{*}, continúa siendo satisfecha.

1.- Si $\mathcal{X}_r = 0$ en $S \epsilon \Gamma_k$, tenemos que cuando se modifica a^{+}_{kr}

$$\rho^{s}_{k}{}' = \rho^{s}_{k} \quad \text{(pues } \mathcal{X}_r \text{ es fijo según S)}$$

y

$$Y^{s}_{k}{}' = b_k - \sum_{j \epsilon J^1} a_{kj} = Y^{s}_{k} \quad (J^1 \text{ según S})$$

por lo que

$$\rho^{s}_{k}{}' = \rho^{s}_{k} > Y^{s}_{k} = \rho^{s}_{k}$$

y entonces S sigue siendo infactible.

2.- Si $\mathcal{X}_r = 1$ en $S \epsilon \Gamma_k$

$$\rho^{s}_{k}{}' = \rho^{s}_{k}$$

y

$$Y^{s}_{k}{}' = b_k - \sum_{j \epsilon J^1 - \{r\}} a^{+}_{kj} - a_{kr}$$

$$= b_k - \sum_{j \epsilon J^1 - \{r\}} a^{+}_{kj} - a^{+}_{kr} + a^{+}_{kr} - a_{kr}$$

$$= Y^{s}_{k} + a^{+}_{kr} - a_{kr}$$

$$< Y^{s}_{k} + a^{+}_{kr} - a^{+}_{kr} + \phi^{s} \quad \text{(debido a que } a_{kr} > a^{+}_{kr} - \phi^{s} \text{ en S)}$$

$= Y^*_k + \phi^*$

$= Y^*_k + \rho^*_k - Y^*_k$

$= \rho^*_k$

por lo que

$\rho^{*\prime}_k = \rho^*_k > Y^{*\prime}_k$

y entonces S sigue siendo infactible.

3.- En el caso que $\mathcal{X}_r$ sea libre según $S \in \Gamma_k$ entonces:

(i).- Si $a_{kr} < 0$, se tiene:

$$\rho^{*\prime}_k = \sum_{j \in J^1 - \{r\}} \min(0, a^+_{kj}) + a_{kr}$$

$$= \sum_{j \in J^1 - \{r\}} \min(0, a^+_{kr}) + a^+_{kr} - a^+_{kr} + a_{kr}$$

$$= \sum_{j \in J} \min(0, a^+_{kj}) - a^+_{kr} + a_{kr}$$

$$= \rho^*_k - a^+_{kr} + a_{kr}$$

$$> \rho^*_k - a^+_{kr} + a^+_{kr} - \phi^* \qquad \text{(ya que } a^+_{kr} - \phi^* < a_{kr}\text{)}$$

$$= \rho^*_k - \phi^*$$

$$= \rho^*_k - (\rho^*_k - Y^*_k) \qquad \text{(ya que } \mathcal{X}_r \text{ es libre pero } a_{kr} < 0\text{)}$$

$$= Y^*_k$$

esto es

$$\rho_k^{\pi\,\prime} > Y_k^{\pi}$$

pero como $\mathcal{I}_r$ es libre según S, entonces

$$Y_k^{\pi\,\prime} = Y_k^{\pi}$$

y por lo tanto

$$\rho_k^{\pi\,\prime} > Y_k^{\pi} = Y_k^{\pi\,\prime} \quad \text{y S sigue siendo infactible}$$

(ii).- Si $a_{kr} \geq 0$

se tiene:

$$\rho_k^{\pi\,\prime} = \rho_k^{\pi}$$

y como $\mathcal{I}_r$ es libre según S; se tiene que

$$Y_k^{\pi\,\prime} = Y_k^{\pi}$$

por lo que en este caso se concluye que

$$\rho_k^{\pi\,\prime} = \rho_k^{\pi} > Y_k^{\pi} = Y_k^{\pi\,\prime} \quad \text{y S sigue siendo infactible}$$

En cualquier caso, si a_{kr}^{+} es decrementado en a lo más la cantidad ϕ^{π}, esto es $a_{kr}^{+} - \phi^{\pi} < a_{kr}$, entonces la restricción K en $S \in \Gamma_k$ sigue siendo infactible. Y para asegurar que toda $S \in \Gamma_k$ permanece infactible, basta tomar el mínimo de los decrementos, esto es $V_{kr} =$ mínimo $\{\phi^{\pi}\}$, ya que de esta forma toda solución $S \in \Gamma_k$ permanece infactible para cambios del valor de a_{kr}^{+} arriba del valor $a_{kr}^{+} - V_{kr}$, es decir $a_{kr} > a_{kr}^{+} - V_{kr}$.

Finalmente debemos tener claro que si incrementamos el valor a^{+}_{kr}, siempre tendremos que para toda $S \in \Gamma_k$, S permanece infactible. Sin embargo para asegurar que la solución S^{*} permanezca invariante, es necesario pedir que la restricción K continúe satisfaciendose, según S^{*}. Así que cuando a^{+}_{kr} es decrementado; entonces dicha condición no cambia, ya que $\rho^{*}_{k} ' < \rho^{*}_{k} < Y^{*}_{k} = Y^{*}_{k}$, y lo mismo sucedera si a^{+}_{kr} es incrementado, en una cantidad de a lo más S_k unidades, donde:

$S_k = Y^{*}_{k} - \sum_{j \in J^1} a_{kj}$ (La cantidad de holgura de la restricción K en S^{*})

Por tanto la solución S^{*} permanece óptima para cambios de a^{+}_{kr} en el intervalo

$$a^{+}_{kr} - V_{kr} < a_{kr} \leq a^{+}_{kr} + S_k$$

III. CAMBIOS SIMPLES DE PARAMETROS (CSP)

Con cualquier solución parcial S, nosotros podemos asociar un problema parcial (I^{*}) de la forma

(I^{S}) Minimizar $Z = \sum_{j \in J} C^{+}_{j} X_j + Z^{*}$

sujeto a

$$\sum_{j \in J^1} a^{+}_{ij} X_j \leq b^{*}_{i}$$

$$X_j = 0 \text{ ó } 1 \quad j \in J$$

donde

$$Z^{*} = \sum_{j \in J^1} C_j \quad y \quad b^{*}_{i} = b^{+}_{i} - \sum_{j \in J^1} a^{+}_{ij}$$

$$J = \{j \mid x_j \text{ es libre respecto al nodo actual}\}$$

y

$$J^1 = \{j \mid x_j = 1 \text{ hasta el nodo actual S}\}.$$

Cada solución para (I^n), junto con el mismo S, forman una complementación de S para (I^*). La mejor solución para (I^n) producirá la mejor complementación de S para (I^*) y al mismo tiempo cualquier solución factible (infactible) de (I^n) producirá una complementación factible (infactible) de S en (I^*).

Para localizar un nuevo óptimo donde un sólo parámetro es cambiado, se necesita simplemente encontrar una secuencia apropiada de problemas parciales.

(a) Revisión de b_k

Cada vez que modifiquemos el parámetro b_k^* y el nuevo valor sea una cantidad mayor que cualquier valor del intervalo

$$b_k \in (b_k - S_k , b_k^* + U_k]$$

entonces no es posible asegurar que la solución S^* permanezca óptima. Por lo que es necesario formular el problema (I^n) para cada $S \in \Gamma_k$ usando el valor revisado de b_k y un valor inicial de Z determinado por la mejor solución hasta el momento encontrada (ya sea en la fase de solución o en cualquier problema parcial previamente resuelto). Posteriormente es necesario resolver completamente este problema usando el algoritmo de enumeración implícita. Si cada (I^n), $S \in \Gamma_k$, ha sido examinado, una nueva solución óptima, si ésta existe, habrá sido encontrada.

(c) Revisión de C_r

Suponiendo que $x_r = 0$ en el actual óptimo S^* , si el parámetro C_r^* es

decrementado abajo del intervalo $(C_r^*-W_r , \infty)$ entonces no es posible asegurar que la solución S^* permanezca óptima. Sin embargo para encontar una mejor solución (si ésta existe) es necesario formular el problema (I^s) para cada $S\in\Gamma_0$ en el que X_r no está fijada en cero, usando para ello el valor revisado de C_r y un valor inicial de Z determinado por la mejor solución hasta el momento encontrada. Resolver este problema completamente usando el algoritmo aditivo y cuando cada (I^s) ha sido examinado, una nueva solución óptima, si ésta existe, habrá sido encontrada.

(c) Revisión de a_{kr}

Suponiendo que $X_r=0$ en el actual óptimo S^*, si el parámetro a_{kr}^* es decrementado abajo del intervalo $(a_{kr}^*-V_{kr} , a_{kr}^*+S_k)$ entonces la actual solución S^*, puede no permanecer óptima, sin embargo para encontrar una mejor solución óptima, en caso de que ésta exista, es necesario formular y resolver el problema (I^s) $S\in\Gamma_k$, usando el valor revisado de a_{kr}^* y un valor inicial de Z, que esta determinado por la mejor solución hasta el momento encontrada. Cuando cada (I^s) ha sido examinado, un nuevo óptimo (si éste éxiste) habrá sido encontrado.

IV CAMBIOS MULTIPLES DE PARAMETROS (CMP)

Para el caso en que se trate de hacer cambios en un conjunto de parámetros a la vez, el análisis en algunos casos, pueden ser reducido al análisis CSP (cambios simples de parámetros), sin embargo una mayor generalización de esto, se verá en la parte referida a Cambios Permanentes de Parámetros (CPP).

(a) Revisión de $b_i, b_k, \ldots$ y b_l

Cada vez que se alteren conjuntamente los valores de $b_i^*, b_k^*, \ldots, b_l^*$ y cada nuevo valor sea mayor que cualquier cantidad que esta en el

intervalo $[b_1^+-S_1 , b_1^+ + U_1)$, $[b_k^+-S_k , b_k^+ + U_k)$,...,$[b_l^+-S_l , b_l^+ + U_l)$ respectivamente, es necesario que para cada $S \in \Gamma_l U \Gamma_k U \ldots U \Gamma_l$ formular (I^*) usando el valor revisado de $b_1, b_k, \ldots, b_l$ y un valor inicial de $\hat{Z}$ determinado por la mejor solución factible ya encontrada. Cuando cada (I^*), $S \in \Gamma_l U \Gamma_k U \ldots U \Gamma_l$ ha sido examinado, un nuevo óptimo (si éste existe) habrá sido encontrado.

Sin embargo es necesario aclarar que para las restricciones del tipo igualdad no son analizadas en este proceso, debido a que este tipo de restricciones, que son de la forma

$$\sum_{j=1}^{n} a_j X_j = b$$

necesitan ser tratadas como 2 restricciones

$$b_1 \leq \sum_{j=1}^{n} a_j X_j \leq b_2 \quad \text{donde } b_1 = b_2 = b$$

y entonces cuando se quiera "relajar" la restricción, decrementando para ello b_1 o incrementando b_2, no es posible hacer su análisis de postoptimalidad, debido a que $b_1 = b_2 = b$ y un decremento de b_1 implica forzosamente un incremento de b_2, caso que no contempla el proceso ya presentado, pues sólo se analizan incrementos conjuntos de las $b_{i'}^+$.

(b) Revisión de $C_q, C_r, \ldots$ y C_s

Cada vez que requiera analizarse conjuntamente cambios en los parámetros $C_q^+, C_r^+, \ldots C_s^+$ siempre es necesario revisar cada $S \in \Gamma_0$ en los que $X_q, X_r, \ldots$ y/o X_s no esta fijado en cero y reformular el problema (I^*), usando el valor revisado de $C_q, C_r, \ldots C_s$ y un valor inicial de $\hat{Z}$ determinado por la mejor solución hasta el momento encontrada. Cuando cada (I^*) ha sido examinada, un nuevo óptimo (si es que existe) habrá sido encontrado.

(c) Revisión de $a_{iq}, a_{kr}, \ldots$ y a_{ls}

Para el caso del análisis conjunto de los coeficientes de restricción es necesario mencionar que si los coeficientes $a_{iq}, a_{kr}, \ldots$ y a_{ls} todos pertenecen a restricciones distintas y además cada uno de ellos cae en el intervalo dado en el Análisis de Rango (AR) entonces la solución S^* no cambia. Si éste no fuera el caso, entonces es necesario que para cada $S \in \Gamma_i \cup \Gamma_k \cup \ldots \cup \Gamma_l$ en que $x_q, x_r, \ldots$ y/o x_s no están fijados en cero, formular (I^s) usando los valores revisados de $a_{iq}, a_{kr}, \ldots a_{ls}$ y un valor inicial de $\hat{z}$ determinado por la mejor solución hasta el momento encontrada. Cuando cada (I^s) ha sido examinado, un nuevo óptimo (si existe) habrá sido encontrado.

V ELIMINACION DE RESTRICCIONES

El análisis correspondiente para cuando se eliminen las restricciones i,k,...y l, es realizado mediante el procedimiento CMP(a), para ello basta asignar arbitrariamente valores grandes para cada $b_i, b_k, \ldots$ y b_l, ya que esto equivale a borrar las restricciones i,k,...yl para cada (I^s) como está formulado.

VI ADICION DE VARIABLES

La adición de variables q,r,...y s es más fácilmente tratada al incluir en el problema (I^*) cada variable de interés, pero inicialmente asignando arbitrariamente valores grandes para $C_q^+, C_r^+, \ldots$ y C_s^+. Posteriormente para contemplar estas variables, reducimos los valores de $C_q, C_r, \ldots$ y C_s y usamos el procedimiento CMP(b).

Alternativamente, si no deseamos incluir estas variables, es necesario examinar todo $S \in \Gamma_k$ para los que $a_{kq} < 0$ y/o $a_{kr} < 0 \ldots$ y/o $a_{ks} < 0$, pues para aquellas restricciones K (de las soluciones $S \in \Gamma_k$)

en las que $a_{kq}>0$ y/o $a_{kr}>0$ ... y/o $a_{ks}>0$, los valores de ρ_k^* y Y_k^* no se alteran al incluir las variables q,r,...y s, por lo que la solución S seguirá siendo infactible debido a la restricción K.

VII CAMBIOS PERMANENTES DE PARAMETROS

Los parámetros de (I^*), junto con el algoritmo de enumeración implícita, determinan la partición de soluciones determinada en la fase de solución. En la fase de postoptimalidad, cambios en estos parámetros pueden ser hechos permanente o temporalmente, dependiendo sobre si la partición inicial es o no apropiada para reflejar esto. Para el caso en que se quiera cambiar de manera permanente algún parámetro (con el propósito de buscar una nueva solución para el caso en que se quiera analizar simultáneamente cambios en los párametros C_q^*'s, b_k^*'s, a_{ij}^*'s, eliminación de restricciones y adición de variables, pues esto aún no pueden ser análizados con el proceso anteriormente presentado), es necesario que mientras resolvemos la secuencia requerida de problemas parciales, simultaneamente llevemos a cabo dos operaciones básicas:

(i). Eliminar del conjunto original Γ, aquellas soluciones parciales que necesitan ser revisadas, debido a los respectivos cambios realizados en los parámetros de (I^*).

(ii). Resolver los problemas parciales asociados, con el algoritmo aditivo como se modificó en la fase de solución y agregar al conjunto Γ los nuevos nodos terminales, obtenidos de esta revisión.

Si esto no es hecho, entonces cada cambio nuevo de algún parámetro es, en efecto, hecho en el mismo problema (I^*) y ésto significa que el cambio es temporal.

Una vez que el problema (I^*) es resuelto y una solución óptima inicial es encontrada, podemos usar los diversos elementos del análisis de postoptimalidad en una variedad de formas. Así por ejemplo, podemos seguir la solución óptima sobre una secuencia de cambios permanentes en los parámetros y probar la sensibilidad de cada uno de ellos, usando para ello el Análisis de Rango. Alternativamente, se puede simplemente querer comparar el óptimo generado sobre un conjunto de cambios temporales o realizar cambios de manera conjunta, en cualquiera de los parámetros, mediante los cambios permanentes de parámetros.

Finalmente es necesario mencionar, que al aplicar el análisis de postoptimalidad, los experimentos realizados en muchos problemas de prueba para cambios muy drásticos en los valores de los parámetros, nos indica que los resultados no son tan malos como aquellos problemas que han sido reformulados y resueltos desde el principio. Claro que esta experiencia tiende a ser acumulada.

EJEMPLO

Considere el problema

$$\textit{Minimizar} \quad Z = 2X_1 + 5X_2 + 5X_3 + 6X_4 + 4X_5 + X_6 + 8X_7 + X_8$$

sujeto a

$$-2X_1 + 2X_4 - 6X_5 + X_6 - X_7 + 2X_8 \leq -5$$

$$-4X_1 + 11X_2 - 11X_3 - 7X_4 + 4X_5 + 3X_6 - 5X_7 + X_8 \leq -6$$

$$X_2 + X_3 + X_4 - X_5 - 2X_6 + X_8 \leq 0$$

$$X_j = 0 \text{ ó } 1 \qquad j=1,\ldots,8.$$

La solución de este problema es dada para mostrar los procedimientos previamente presentados. La solución parcial S, es mantenida como un vector en que "j" ("-j") que es un elemento de la secuencia, si la variable X_j es fijada en 1 (o fijada en 0). Los elementos "J" de la solución parcial, indican que la variable X_j ha sido fijada en 1, debido a que su complemento (X_j=0) arroja una solución infactible, así mismo, los elementos "-j" indican que la variable Xj es fijada en 0, debido a la infactibilidad de la combinación con X_j=1, en el nodo actual.

I.- FASE DE SOLUCION

La secuencia de soluciones parciales examinadas en el problema resuelto aparecen en la primera columna de la Tabla I. En la segunda columna aparece el vector asociado ρ^s= (ρ_1^s , ρ_2^s , ρ_3^s) y el vector de términos independientes para cada S, a saber Y^s=(Y_1^s , Y_2^s , Y_j^s).

Cada vez que una solución parcial es enumerada y analizada, ésta es asignada al número que aparece en la última columna de la Tabla I y en la Tabla II siguiente. Inicialmente ρ =(-9, -27, -3) y Y =(-5, -6, 0). La ramificación es hecha usando la propuesta por Balas, esto es, ramificar sobre aquella variable que maximiza

$$V_J = \sum_{i \in M_j} (Y_i - a_{ij})$$

donde

$$M_J = \{ i \mid Y_i - a_{ij} < 0 \}$$

la solución óptima al problema es X_3= X_5= 1 y todas las otras variables iguales a cero, Z^* = 9.

TABLA I

SOLUCION PARCIAL	VECTORES DE PRUEBA	COMENTARIOS
{5}	ρ=(-3,-27,-2) Z=4 Y=(1,-10,1)	Variable 5 fijada en 1 debido a la restricción 1
{5,3}	ρ=(-3,-16,-2) Z=9 Y=(1,1,0)	Mejor solución encontrada hasta el momento
{5,-3}	ρ=(-3,-16,-2) Z=4 Y=(1,-10,1)	Backtrack
{5,-3,-2}	ρ=(-3,-16,-2) Z=4 Y=(1,-10,1)	Variable 2 fijada en 0 debido a la restricción 2
{5,-3,-2,4}	ρ=(-3,-9,-2) Z=10 Y=(-1,-3,0)	Variable 4 fijada en 1 debido a la restricción 2
{5,-3,-2,4}	ρ=(-3,-9,-2) Z=10 Y=(-1,-3,0)	Infactible debido a la función objetivo

b.- FASE DE POSTOPTIMALIDAD

En la tabla II cada solución parcial ha sido atribuida a la restricción apropiada ubicandola en una de las 4 celdas. A continuación de la solución parcial en la celda i (i=1,2,3) están las cantidades $(\rho_i^*-Y_i^*)$ y Z, la primera de éstas es la cantidad por la que aún la mejor complementación de S (con el propósito de que la restricción i sea satisfecha, según S, lo más que se podría hacer es complementar S al asignar X_{ij}=1 si $a_{ij} \leq 0$) no satisface

la restricción 1, la segunda es el valor de la función objetivo asociado a la mejor complementación de S. Solamente la segunda de estas cantidades aparece en la celda de la función objetivo.

TABLA II

Función objetivo	Restricción 1
{5,3} [9] {5,-3,-2,4} [10]	{-5} [2,0]
Restricción 2	**Restricción 3**
{5,-3,2} [5,9] {5,-3,-2,-4} [1,4]	

El Análisis de Rango sobre los coeficientes de costo requiere la examinación de lo registrado en la celda de la función objetivo. Porque $\mathfrak{X}_1$, $\mathfrak{X}_2$, $\mathfrak{X}_4$, $\mathfrak{X}_6$, $\mathfrak{X}_7$ y $\mathfrak{X}_8$ están fijados en 0 en el actual óptimo, se tiene que para alteraciones en los coeficientes C_1^* , C_2^* , C_4^*, C_6^* , C_7^* y C_8^* que estén en los siguientes intervalos, la solución óptima no cambia.

$C_1^*-W_1 < C_1 < \infty$ $\quad C_1^*= 2,\ W_1= \min\{(9-9+2),\ (10-9+2)\} =2$

$C_2^*-W_2 < C_2 < \infty$ $\quad C_2^*= 5,\ W_2= \min\{(9-9+5)\} = 5$

$C_4^*-W_4 < C_4 < \infty$ $\quad C_4^*= 6,\ W_4= \min\{(9-9+6),\ (10-9)\} = 1$

$C_6^*-W_6 < C_6 < \infty$ $\quad C_6^*= 1,\ W_6= \min\{(9-9+1),\ (10-9+6)\} =1$

$C_7^*-W_7 < C_7 < \infty$ $\quad C_7^*= 8,\ W_7= \min\{(9-9+8),\ (10-9+8)\} =8$

$C_8^*-W_8 < C_8 < \infty$ $\quad C_8^*= 1,\ W_8= \min\{(9-9+1),\ (10-9+1)\} =1$

Por que $\mathcal{X}_3$ y $\mathcal{X}_5$ son fijados en 1 en el actual óptimo, es claro que si C_3^+ y C_5^+ se incrementan; entonces el actual óptimo cambia, sin embargo, cuando C_3^+ y C_5^+ son decrementados, el óptimo permanece invariante, salvo por modificaciones de Z^*. Por tanto el rango para C_3^+ y C_5^+ es:

$$-\infty < C_3 < C_3^+$$

y

$$-\infty < C_5 < C_5^+$$

Para búscar una cota para el lado derecho de la segunda restricción, nosotros examinamos las soluciones parciales en la celda correspondiente a la restricción 2, una cota $Z \leq Z^*$. Así pues, el rango de valores sobre los que puede alterarse el parámetro b_2^+ sin que la solución óptima cambie es :

$$b_2^+ - S_2 \leq b_2 < b_2^+ + U_2$$

donde $b_2^+ = -6$, $U_2 = \min\{5, 1\} = 1$, $S_2 = 1$

Análogamente

$$b_1^+ - S_1 \leq b_1 < b_1^+ + U_1$$

con $b_1^+ = -5$, $U_1 = \min\{2\} = 2$, $S_1 = 1$

y por que ninguna solución parcial es atribuida a la restricción 3, nosotros podemos tomar $U_3 = \infty$, por lo que el análisis de rango para b_3^+ es:

$$b_3^+ - S_3 \leq b_3 < b_3^+ + U_3 \qquad b_3^+ = 0,\ S_3 = 0$$

Para el caso del análisis de rango para los coeficientes de restricción a_{kr}^+, éste es muy similar al hecho para los coeficientes de costo C_r^+, sin embargo esto no representa mayor dificultad, que el hacer simples cálculos y por ello no es presentado.

Adicionalmente es posible realizar cambios, conjuntamente en varios parámetros (dentro de los rangos establecidos), sin que la solución óptima cambie, para lo que es necesario cuidar de no alterar dos o más parámetros de una misma restricción, ni alterar a la vez dos o más coeficientes de costo.

Para ello, suponga que C_2 y C_4 son ambos reducidos a 2. Las entradas de la celda cero puede ser otra vez consideradas. Y la única solución parcial que no tiene fijada a X_2 y/o X_4 y cuyo valor Z es menor a Z^*, es la solución $\{5,-3,-2,4\}$; por lo que en caso de existir una mejor solución, es necesario resolver el problema parcial:

$$\textit{Minimizar} \quad Z = 2X_1 + X_6 + 8X_7 + X_8 + 6$$

sujeto a

$$-2X_1 + X_6 - X_7 + 2X_8 \leq -5$$

$$-4X_1 + 3X_6 - 5X_7 + X_8 \leq -6$$

$$-2X_6 + X_8 \leq 0$$

$$X_1,\ X_6,\ X_7,\ X_8 = 0 \text{ ó } 1.$$

Realizando una iteración del algoritmo de enumeración implícita, encontramos que al instanciar $X_1 = 1$, el problema es factible y por tanto una mejor solución es:

$$X_1=X_4=X_5=1 \qquad X_2=X_3=X_6=X_7=X_8=0 \qquad Z^*=8$$

En cambio, si b_1 fuera incrementado a -2, el problema parcial $\{-5\}$ requiere formulación y solución, es decir, el siguiente problema debe ser resuelto:

$$\textit{Minimizar}\quad Z = 2x_1 + 5x_2 + 5x_3 + 6x_4 + x_6 + 8x_7 + x_8$$

sujeto a

$$-2x_1 + 2x_4 + x_6 - x_7 + 2x_8 \leq -5$$

$$-4x_1 + 11x_2 - 11x_3 - 7x_4 + 3x_6 - 5x_7 + x_8 \leq -6$$

$$x_2 + x_3 + x_4 - 2x_6 + x_8 \leq 0$$

Sin embargo uno puede verificar que este problema no proporciona una solución factible menor que el óptimo original.

Finalmente, supongase que la restricción 2 fuera eliminada de el problema. Las soluciones parciales $\{5,-3,2\}$ y $\{5,-3,-2,-4\}$ requerirán consideración. Así con la modificación hecha, el primer problema es factible y arroja una $Z=9$, sin embargo el segundo de éstos es también factible y la cota es $Z=4$. Por tanto la mejor solución al problema revisado es $x_1=x_2=x_3=x_4=x_6=x_7=x_8=0$, $x_5=1$ y $Z=4$.

3.4. EL PROBLEMA DE ASIGNACION

El problema de asignación puede ser planteado en un modelo de programación cero-uno y resuelto vía el algoritmo de enumeración implícita. Lo cual es una buena forma de resolver el problema, pero que no es comparable con el siguiente algoritmo específicos de RyA, que resuelven esta clase particular de problemas.

3.4.1. ASIGNACION DE PERSONAL

El problema es definido por la matriz C(i,j) de preferencias:

$$[\,C(i,j)\,] = \begin{array}{c} \\ \text{Aspirante 1} \\ \text{Aspirante 2} \\ \vdots \\ \text{Aspirante n} \end{array} \begin{array}{c} \begin{array}{cccc} \text{Empleo 1} & \text{Empleo 2} & \cdots & \text{Empleo n} \end{array} \\ \begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1n} \\ C_{21} & C_{22} & \cdots & C_{2n} \\ \vdots & \vdots & & \vdots \\ C_{n1} & C_{n2} & \cdots & C_{nn} \end{bmatrix} \end{array}$$

matriz que es construida con la calificación C_{ij} de cada uno de los aspirantes, en cada uno de los empleos ofrecidos. Deberán calificarse todos los aspirantes en todos los empleos. Y deben existir tantos aspirantes como empleos, pues en caso que existan más aspirantes que empleos, basta agregar un número necesario de empleos ficticios con un costo de asignación todos iguales al menor (mayor) de los valores de la matriz de preferencias, en caso de tratarse de un problema del tipo maximizar (minimizar), de otro modo, si hubiese más empleos que aspirantes, basta considerar a los empleos como aspirantes y viceversa.

3.4.2. ALGORITMO DE LITTLE

Ejemplo

Suponga que necesitamos asignar, minimizando el problema, dado por la siguiente matriz:

$$[C(i,j)] = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 5 & 4 & 6 & 4 \\ 7 & 4 & 9 & 8 \\ 9 & 4 & 7 & 10 \\ 4 & 5 & 4 & 6 \end{bmatrix} \end{array}$$

Tabla 1

Solución

Empezaremos por observar que si T es el costo total de cualquier asignación factible, entonces, el costo total de esa misma asignación con la matriz que resulta de restar un escalar α al renglón i es dado por $T-\alpha$. Lo mismo sucede si restamos un escalar α de la columna j (j=A,B,C,D). Una cota inferior a la asignación óptima es sencilla de obtener, si restamos de cada renglón de la matriz de costos el mínimo elemento correspondiente y posteriormente, a la matriz resultante, efectuarle la misma operación sólo que hacerlo ahora por columnas: En la tabla 1 restamos el escalar 4 del primer renglón y los escalares 4, 4 y 4 de los respectivos renglones restantes, así obtenemos una nueva matriz de costos:

	A	B	C	D
A	1	0	2	0
B	3	0	5	4
C	5	0	3	6
D	0	1	0	2

Tabla 2 (costos modif.)

como no es posible reducirla por columnas, entonces, una asignación de costo total T asociada a la tabla 1, tiene un costo T-16 cuando el costo es calculado en base a la tabla 2.

Un aspecto importante de la discusión anterior, es que cualquier asignación asociada a la tabla 2 tiene un costo no-negativo y que difiere del costo de asignación original en 16 unidades. Equivalentemente una cota inferior de la asignación óptima es 16.

Método de Ramificación y Acotamiento

El proceso para llegar a la identificación de una asignación óptima, es comenzado al particionar el conjunto de todas las asignaciones posibles, como sigue:

(a1.) Asignaciones que incluyen la opción XY (X,Y=A,B,C,D)
(a2.) Asignaciones que no incluyen la opción XY

Para identificar la opción XY sobre la qué es hecha la ramificación, debemos analizar para cada opción que tiene costo de asignación igual a cero en la tabla 3, la correspondiente penalización por no usarla en la asignación (suma del mínimo en su renglón, más el mínimo en su columna) y seleccionar, entonces, la opción de mayor penalización.

$$\begin{bmatrix} & {}^{0} & & {}^{2} \\ 1 & 0_{3} & 2 & 0 \\ 3 & 0_{3} & 5 & 4 \\ 5_{1} & 0 & 3_{2} & 6 \\ 0 & 1 & 0 & 2 \end{bmatrix}$$ Tabla 4

la penalización se especifica en el extremo superior derecho de cada entrada con valor igual a cero.

La ramificación es hecha al realizar la asignación BB o no realizarla.

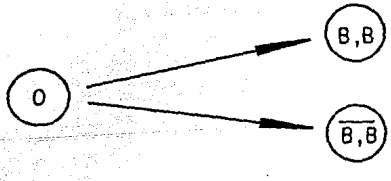


FIGURA 14

En el primer caso, la matriz de costos entre localidades se reduce a una matriz en donde se elimina el renglón 2 y la columna 2,

obteniendose así la tabla 4:

	A	B	C	D
A	1	•	2	0
B	•	•	•	•
C	5	•	3	6
D	0	•	0	2

Tabla 4

Una cota inferior al costo de las asignaciones asociadas con la Tabla 4, se obtienen al reducir por renglones y columnas los costos de dicha tabla y agregarle el costo acumulado hasta la tabla 3, que en este caso es: 16 unidades obtenidas hasta la tabla 3, más 3 unidades obtenidas al reducir la tabla 4. Equivalentemente, 19 unidades es una cota inferior al costo de las asignaciones que usan la opción BB.

	A	B	C	D
A	1	•	2	0
B	•	•	•	•
C	2	•	0	3
D	0	•	0	2

Tabla 5

Una cota inferior al costo de asignaciones que no usan la opción BB es sencilla de obtener, si asignamos el valor M (cantidad que excede cualquier cota superior) al costo de la opción BB, que equivale a no utilizar dicha opción, de este modo se obtiene la tabla 6:

	A	B	C	D
A	1	0	2	0
B	3	M	5	4
C	5	0	3	6
D	0	1	0	2

Tabla 6

a la cual pueden reducirsele los costos, obteniendo así la tabla 7:

	A	B	C	D
A	1	0	2	0
B	0	M	2	1
C	5	0	3	6
D	0	1	0	2

Tabla 7

Al evitar usar la opción BB es necesario pagar un costo adicional de 3 unidades. Esto significa que 19 unidades es una cota inferior de aquellas asignaciones que no usan la opción BB.

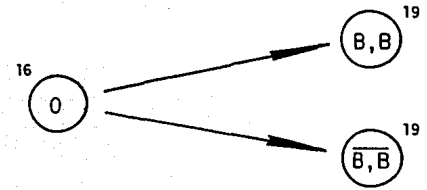


FIGURA 15

Como siguiente paso y aún cuando hay empate en los nodos terminales, ramificaremos el nodo $\overline{BB}$ (el de mínima cota inferior). Para ello partimos de la tabla 7 y calculamos, para cada opción, la correspondiente penalización por no utilizar la asignación BB.

		0		1
1	0	2	0	
0	M	2	1	
5	0	3	6	
0	1	0	2	

Tabla 8

se elige la opción CB para ramificar como sigue:

(a1.) Asignaciones que usan la opción CB.
(a2.) Asignaciones que no usan la opción CB.

En el primer caso partimos de la tabla 7 y eliminamos el renglón 3 y columna 2.

	A	B	C	D
A	1	•	2	0
B	0	•	2	1
C	•	•	•	•
D	0	•	0	2

Tabla 9

tabla a la cual ya no pueden reducirsele los costos y entonces una cota inferior para las asignaciones que usan la opción CB en el nodo BB es 19 unidades.

Una cota inferior del costo de las asignaciones que no usan la opción CB es obtenida al considerar la tabla 7 con un costo igual a M en la opción CB.

	A	B	C	D
A	1	0	2	0
B	0	M	2	1
C	5	M	3	6
D	0	1	0	2

Tabla 10

La cual es reducida en 3 unidades del tercer renglón:

	A	B	C	D
A	1	0	2	0
B	0	M	2	1
C	2	M	0	3
D	0	1	0	2

Tabla 11

por lo que el valor de 22 unidades es una cota inferior para las asignaciones del actual nodo y que no usan la opción CB

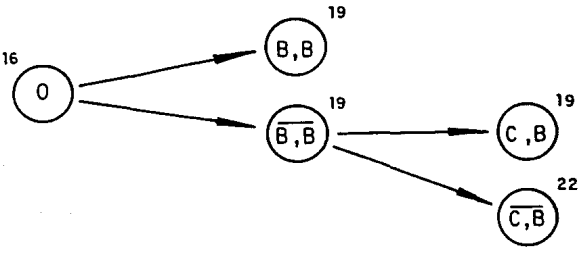


FIGURA 16

Ramificamos ahora la rama CB, la matriz asociada es:

	A	B	C	D
A	1_1	·	2	0^2
B	0	·	2	1
C	$\cdot_0$	·	$\cdot_2$	·
D	0	·	0	2

Tabla .12

la ramificación es hecha sobre la asignación AD.

Si asignamos AD la matriz asociada es:

	A	B	C	D
A	·	·	·	·
B	0	·	2	·
C	·	·	·	·
D	0	·	0	·

Tabla 13

tabla a la cual ya no pueden reducirsele los costos y entonces 19 unidades es una cota inferior para las asignaciones en el actual nodo y que usan la opción AD.

Si no asignamos AD la matriz asociada es:

	A	B	C	D
A	1	•	2	0
B	0	•	2	1
C	•	•	•	•
D	0	•	0	2

Tabla 14

la cual se reduce al restar una unidad del primer renglón y después una unidad de la cuarta columna.

	A	B	C	D
A	0	•	1	M
B	0	•	2	0
C	•	•	•	•
D	0	•	0	1

Tabla 15

así que, 19+1+1=21 unidades es una cota inferior para el nodo $\overline{AD}$.

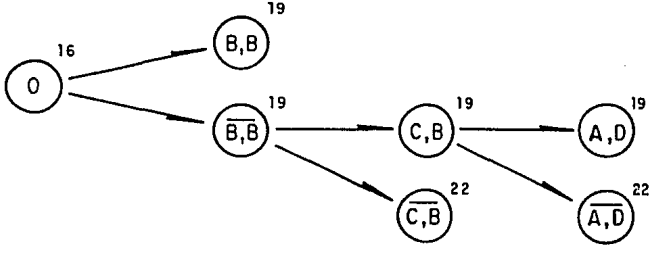


FIGURA 17

Ramificamos ahora el nodo terminal AD. La matriz asociada es:

	A	B	C	D
A	•	•	•	•
B	0	•	2	•
C	•	•	•	•
D	0	•	0	•

Tabla 16

escogemos la rama BA para la ramificación. Asignado BA, se obtiene la matriz:

	A	B	C	D
A	•	•	•	•
B	•	•	•	•
C	•	•	•	•
D	•	•	0	•

Tabla 17

tabla a la cual no pueden reducirsele los costos y por lo tanto 19 unidades es una cota inferior para la rama BA.

Si no asignamos la opción BA, la matriz asociada es:

	A	B	C	D
A	•	•	•	•
B	M	•	2	•
C	•	•	•	•
D	0	•	0	•

Tabla 18

a la cual se le pueden reducir dos unidades del segundo renglón. Por lo que 19+2=21 unidades es una cota inferior para dicho nodo.

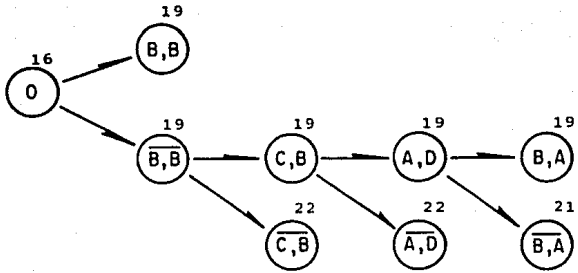


FIGURA 18

Para ramificar ahora la rama BA, partimos de la tabla 17 y vemos que la asignación DC es obligada, ya que si no asignamos DC, tenemos una solución infactible. Además como el costo de la opción DC, en la tabla 17, es cero; entonces 19 es el costo de las asignaciones que se completan con la opción BA.

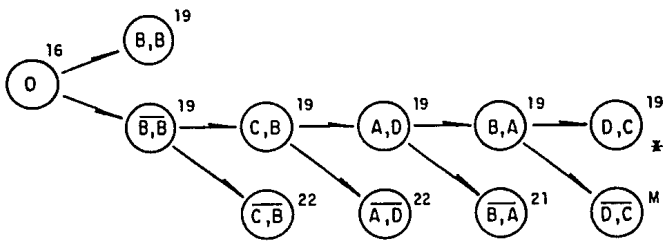


FIGURA 19

Hemos encontrado una solución óptima factible y que es: asignar CB, AD, BA y DC. Sin embargo si continuamos el proceso de ramificación y acotamiento obtenemos el siguiente árbol de búsqueda, donde la asignación de BB, AD, DA y CC es otra solución óptima factible.

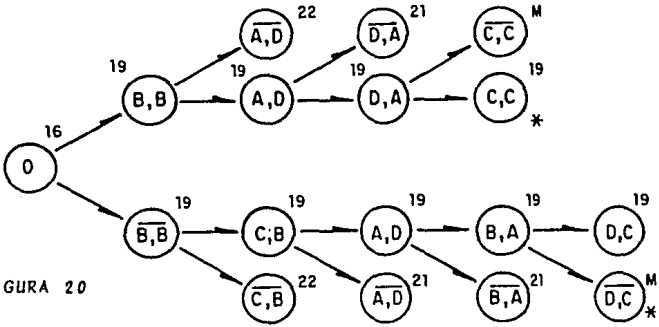


FIGURA 20

Tres observaciones son hechas para un mejor aprovechamiento de este algoritmo:

(a). Si durante el procesos de ramificación ocurre que en un nodo intermedio existe un único cero en cada renglón y cada columna, entonces no hay duda de que la mejor completación del nodo es aquella en la que se asignan las opciones con costo igual a cero. Si éste es el caso, entonces ramificar con aquellas asignaciones cuyo costo es cero y continuar el proceso en la forma acostumbrada, ramificando para ello el nodo de menor cota inferior.

(b). El algoritmo de Little es empleado en problemas de asignación con función objetivo minima y los problemas de función objetivo máxima son resueltos al minimizar los complementos respecto a una constante K (método "húngaro"). Por ejemplo el mayor elemento de la matriz es 12.

$$A = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \left[\begin{array}{cccc} 3 & 3 & 6 & 3 \\ 6 & 9 & 3 & 0 \\ 0 & 6 & 12 & 0 \\ 3 & 0 & 3 & 6 \end{array}\right] \end{array}$$ asignar maximizando

restandole a todos los valores el valor 12, el problema equivale a:

$$A_1 = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \left[\begin{array}{cccc} -9 & -9 & -6 & -9 \\ -6 & -3 & -9 & -12 \\ -12 & -6 & 0 & -12 \\ -9 & -12 & -9 & -6 \end{array}\right] \end{array}$$ asignar maximizando

sólo que Máximo(A) = Máximo(A_1) - 4(12). Y cambiandole el signo a todas las entradas de A_1 tenemos que:

$$A_2 = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \left[\begin{array}{cccc} 9 & 9 & 6 & 9 \\ 6 & 3 & 9 & 12 \\ 12 & 6 & 0 & 12 \\ 9 & 12 & 9 & 6 \end{array}\right] \end{array}$$ **asignar minimizando**

De este modo, las asignaciones en el óptimo para(A), son las mismas que la realizadas en el óptimo para(A_2), pero difieren en valor, ya que:

$$-\text{mínimo}(A_2) = \text{máximo}(A_1) = \text{máximo}(A) + 4(12)$$

(c). El algoritmo de Little es también utilizado para resolver el problema del agente viajero, sólo que con el fin de evitar utilizar los bucles (nodos AA, BB, CC, ...), es necesario asignarles costos igual a M en dichas opciones y también con el propósito de evitar un posible regreso a una ciudad de partida en alguna etapa intermedia, esto es, si por ejemplo se llevan asignados los pares de ciudades (2,3), (3,5) y (5,4)

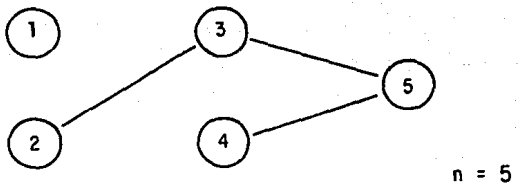


FIGURA 21

deberán haberse hecho iguales a M, de manera paulatina, los costos (3,2), (5,2) y (4,2).

3.5. El Problema de Secuenciacion Optima

El problema general es presentado cuando se cuentan con m máquinas y n trabajos. Así cada trabajo debe ser procesado en todas las máquinas, pero respetando las siguientes situaciones.

(a) Una máquina no puede elaborar dos trabajos simultáneamente.
(b) Un trabajo no puede pasar a la máquina j, si no ha sido terminado ya en la máquina j-1.
(c) Los trabajos son elaborados en la misma secuencia, en cada máquina.

Además se conocen las duraciones d(i,j), que es el tiempo de proceso del trabajo i en la máquina j.

Por ejemplo, supóngase que se tienen 4 trabajos y 3 máquinas, con la siguiente matriz de duraciones:

	Máquinas		
Trabajos	M1	M2	M3
T1	3	5	8
T2	7	6	4
T3	6	3	2
T4	8	9	9

Es fácil ver que el tiempo total desde que el primer trabajo comienza en la primera máquina hasta que el último trabajo sale de la última, varía según sea el orden en que los trabajos son procesados en cada una de las máquinas. Así el problema es encontrar la secuencia de trabajos que minimize el tiempo total de proceso.

Si la secuencia de trabajos fijada para cada una de las tres máquinas es: W = (W1, W2, W3, W4) = (3, 2, 1, 4), el proceso puede representarse gráficamente por:

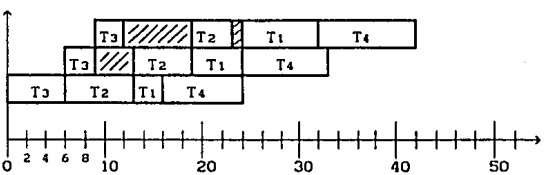


FIGURA 22

De la figura 22 se ve que el tiempo total es de 42 unidades. Este mismo resultado puede obtenerse, sabiendo que:

$f(wi,j) = \max [f(W_{i-1}, j), f(W_i, j-1)] + d_{ij}$ . . . (*)

donde

$f(W_i,j)$ = Instante medido desde el inicio del proceso hasta que el trabajo i es acabado en la máquina j.

$f(W_{i-1},j)$ = Instante en que la máquina j queda en condiciones de poder efectuar el trabajo i.

$f(W_i,j-1)$ = Tiempo medido desde el inicio del proceso hasta que el trabajo i es terminado en la máquina j-1 o sea instante en que el trabajo i podría empezar a efectuarse en la máquina j.

Para el ejemplo particular, si aplicamos la ecuación (*) en forma recursiva para calcular los valores f(wi,j)

$f(W_1,1) = f(3,1) = d\ 3,1)=6$ no hay trabajo, ni máquina anterior

$f(w_1,2) = f(3,2) = f\ 3,1) + d(3,2) = 6 + 3 = 9$

$f(w_1,3) = f(3,3) = f\ 3,2) + d(3,3) = 9 + 2 = 11$

$f(w_2,1) = f(2,1) = f\ 3,1) + d(2,1) = 6 + 7 = 13$

$f(w_2,2) = f(2,2) = \text{máx}\{f(3,2), f(2,1)\} + d(2,2)$
$= \text{máx}\{9,13\} + 6 = 19$

$f(w_2,3) = f(2,3) = \text{máx}\{f(3,3), f(2,2)\} + d(2,3)$
$= \text{máx}\{11,19\} + 4 = 23$

$f(w_3,1) = f(1,1) = f\ 2,1) + d\ \ 1,1) = 13 + 3 = 16$

$f(w_3,2) = f(1,2) = \text{máx}\{f(2,2), f(1,1)\} + d(1,2)$
$= \text{máx}\{19,16\} + 5 = 24$

$f(w_3,3) = f(1,3) = \text{máx}\{f(2,3), f(1,2)\} + d(1,3)$
$= \text{máx}\{23,24\} + 8 = 32$

$f(w_4,1) = f(4,1) = f\ 1,1) + d\ (4,1) = 16 + 8 = 24$

$f(w_4,2) = f(4,2) = \text{máx}\{f(1,2), f(4,1)\} + d(4,2)$
$= \text{máx}\{24,24\} + 9 = 33$

$f(w_4,3) = f(4,3) = \text{máx}\{f(1,3), f(4,2)\} + d(4,3)$
$= \text{máx}\{32,33\} + 9 = 42$

$f(w_2,2) = f(2,2) = \text{máx}\{f(3,2), f(2,1)\} + d(2,2)$
$= \text{máx}\{9,13\} + 6 = 19$

Así obtenemos la matriz de tiempos en que es terminado cada trabajo, en cada máquina:

$$[f(W_1,J)] = \begin{matrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{matrix} \begin{bmatrix} 16 & 24 & 32 \\ 13 & 19 & 23 \\ 6 & 9 & 11 \\ 24 & 33 & 42 \end{bmatrix}$$

Observación: La misma matriz puede ser obtenida también calculando recursivamente f(W1,j), es decir calcular f(W1,1), f(W2,1), f(W3,1), f(W4,1), f(W1,2), etc., en ese orden.

Con la sucesión elegida $W=(W_1,W_2,W_3,W_4)=(3,2,1,4)$, el último trabajo que pasa por M3 es el 4, esto indica que el total de tiempo de proceso para realizar todos los trabajos en todas las máquinas es $f(4,3) = 42$.

En general cuando se tienen n trabajos, el número total de secuencias diferentes que pueden pasar en todas las máquinas es n. En caso que se permita que la secuenciación sea diferente en cada máquina (suponiendo que se tienen m máquinas), es decir que cualquier trabajo puede ser empezado en cualquier máquina, entonces el número de alternativas diferentes es $(n)^m$ y en este caso, el tiempo total de proceso, varía dependiendo de la secuencia que se asigne como inicial para cada máquina.

Sin embargo, este último caso no será tratado y nuestra atención sólo estará centrada en el caso de una secuenciación donde el orden de proceso de los trabajos para cada máquina es el mismo.

ALGORITMO DE LOMNICKI

El algoritmo es aplicable al problema de encontrar el orden en que los n trabajos deberán ser procesados en cada una de las máquinas (se analizará el caso de 3 máquinas, pero la generalización para m máquinas se intuye directamente), respetando la restricción en cuanto al orden de procesamiento de los trabajos, el cual debe ser el mismo para todas las máquinas.

El problema, es un problema de decisión si-no (debe o no asignarse el trabajo W_i en la posición j de la secuencia) e independientemente de que el problema pueda plantearse en un modelo de programación binaria, resulta que el utilizar un proceso de ramificación y acotamiento, es la mejor forma de resolverlo. Y es aquí, donde recurriendo al método de Lomnicki, se logra obtener un algoritmo que resuelve de manera eficiente el problema de la secuenciación óptima, el que a continuación se muestra.

El proceso de ramificación comienza dividiendo a todas las n secuencias posibles en n clases, según sea el elemento inicial de la secuencia. Cada una de estas n clases se dividirán a su vez en n-1 clases según sea el segundo elemento de la secuencia y así sucesivamente. Al final del árbol se tendrán las secuencias perfectamente definidas.

El proceso de acotación para un nodo intermedio, es efectuado al calcular las duraciones de 3 tipos de trayectorias que completan la secuencia dada en dicho nodo y de acuerdo a esto, la cota para el nodo referido, es la mayor de estas 3 duraciones.

Supóngase que estamos analizando un nodo intermedio, donde ya se tiene fija la secuencia $W_1, W_2, \ldots, W_k$ y falta definir quienes son $W_{k+1}, \ldots, W_n$. En estas condiciones, el valor de cada trayectoria asociada a dicho nodo es:

Primera Trayectoria

Cuando definimos particularmente los k primeros trabajos a realizar en la secuencia de n trabajos, el tiempo total requerido para realizar todos los n trabajos, será al menos el tiempo que consume el realizar los primeros k trabajos en las 3 máquinas (es decir $f_k(W_k,3)$), más lo que duraría realizar los trabajos k+1,...,n en la máquina 3. Ya que independientemente del ordón en que sean procesados los n-k trabajos restantes, en el mejor de los casos, la máquina 3 no presentaría tiempos muertos.

Equivalentemente una cota inferior para el conjunto de todas las posibles secuencias de n trabajos, cuando se definen los k primeros trabajos es:

$$g' = f_k(W_k,3) + d(W_{k+1},3) + \ldots + d(W_n,3)$$

Gráficamente:

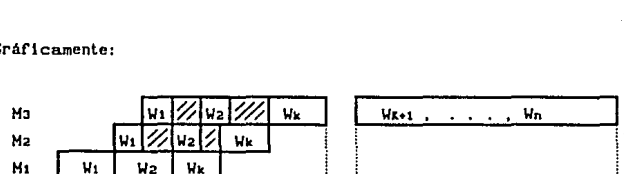


FIGURA 23

Segunda Trayectoria

Para este caso se supondrá que en la máquina 2 no hay tiempos muertos al momento de ser procesados los trabajos $W_{k+1},\ldots,W_n$ y que en el mejor de los casos, el último trabajo que es procesado en la máquina 2 (W_n) puede inmediatamente ser comenzado en la máquina 3. De tal suerte que la máquina 3 terminaría todo el proceso en un

tiempo casi igual al de la máquina 2, salvo por un tiempo igual a la duración mínima de los trabajos $W_{k+1},\ldots,W_n$ (se adopta la de mínima duración, por que se trata de un problema de minimización y conviene conservar aquellas soluciones con mínima duración).

Así que el tiempo total de proceso está acotado inferiormente por:

$$g'' = f_k(W_k,2) + d(W_{k+1},2) + \ldots + d(W_n,2) + \min_{i=k+1,\ldots,n} \{d(W_i,3)\}$$

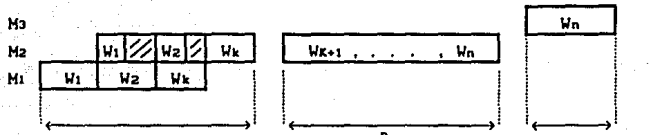


FIGURA 24

Tercera Trayectoria

En la tercera trayectoria supóngase que el último de entre los trabajos $k+1,\ldots,n$, que están por realizarse, puede pasar sin demora por las tres máquinas. De esta forma el tiempo de terminación más próximo del proceso completo es igual al tiempo que se lleva invertido en los trabajos $W_1,\ldots,W_k$ para la máquina 1, más lo mínimo que se necesitaría para completar el trabajo W_n, cuando éste pudiera ser procesado ininterrumpidamente en las máquinas dos y tres.

De esta forma se obtiene otra cota para el tiempo total de proceso, a saber:

$$g''' = \sum_{i=1}^{n} d(W_i,1) + \min_{i = k+1,\ldots,n} \{d(W_i,2) + d(W_i,3)\}$$

Gráficamente:

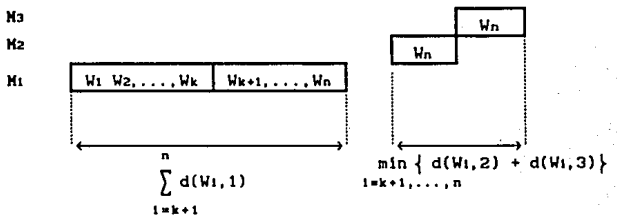


FIGURA 25

Cuando han sido calculados los valores de g', g'' y g''', una cota inferior asociada al nodo cuya secuencia es $W_1,\ldots,W_k$, habrá sido encontrada. Ya que una vez fijada la trayectoria $W_1,\ldots,W_k$, no importa cuales sean los trabajos $W_{k+1},\ldots,W_n$ que complementan dicha trayectoria, el tiempo total de proceso en el mejor de los casos es igual a G_k, donde:

$$G_k = \text{máx } (g', g'', g''')$$

Se considera el máximo por que en caso de tomar el mínimo, el nodo asociado no proporciona un menor tiempo para complementar todos los trabajos, ya que existe una trayectoria con una duración mayor y cuyo valor ya no puede ser reducido aún por la mejor complementación de la secuencia $W_1,\ldots,W_k$, cuando se han fijado los k primeros trabajos.

Con el propósito de checar simplemente que las cotas g', g'' y g''', no dependen unas de otras, se exponen a continuación dos casos particulares.

Supóngase un proceso en el que necesitan ser secuenciados 4 trabajos a lo largo de 3 máquinas. Además suponga que el desarrollo de la solución es iniciado y que se ha optado por fijar al trabajo C como el primer elemento de la secuencia.

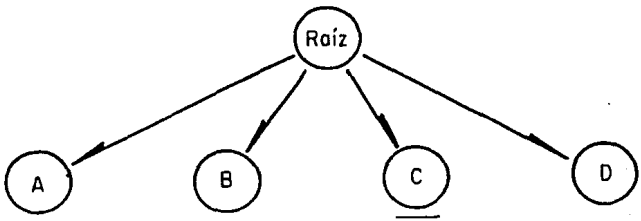
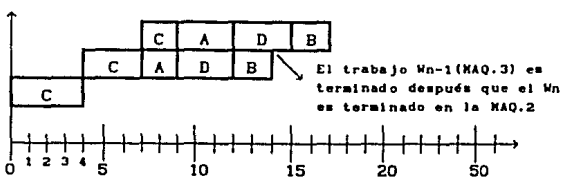


FIGURA 26

CASO 1

Suponga que los tiempos de realización de los trabajos A,B,C Y D para cada máquina, son iguales a los que se muestran en la figura 27, ocasionando por lo tanto que los trabajos sean secuenciados en

acorde con esta gráfica:



el trabajo C es el único trabajo fijo, en el nodo actual del árbol de solución.

FIGURA 27

Así en este caso, dado que no hay tiempos muertos en las máquinas 2 y 3, cuando son procesados los trabajos A, D y B, se obtienen las siguientes cotas para el nodo en donde el trabajo C es fijo:

$$g'_C = f_1(c,3) + [d(A,3) + d(D,3) + d(B,3)]$$
$$= 4 + 3 + 2 + [3 + 3 + 2]$$
$$= 17$$

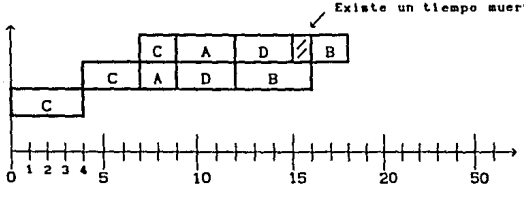
$$g''_C = f_1(c,2) + [d(A,2) + d(D,2) + d(B,2)] + \min_{i=A,D,B} \{d(i,3)\}$$
$$= 4 + 3 + [2 + 3 + 2] + d(B,3)$$
$$= 4 + 3 + [2 + 3 + 2] + 2$$
$$= 16$$

Así que en este caso, la relación obtenida es:

$$g'_C > g''_C$$

CASO 2

El caso $g'_C < g''_C$ puede ser obtenido al considerar alternativamente otros tiempos de realización para cada trabajo, de acuerdo con la figura 28:



el trabajo C es fijo

FIGURA 28

Al presentarse un tiempo muerto, después de que los trabajos C,A y D son procesados en la máquina 3, el valor de las cotas g'_c y g''_c se altera considerablemente, obteniendo en este caso:

$$g'_c = f_1(c,3) + [\, d(A,2) + d(D,3) + d(B,3)] = 4 + 3 + 2 + [3+3+2] = 17$$

y

$$g''_c = f_1(c,2) + [d(A,2), + d(D,2) + d(B,2)] + \min_{i=A,D,B} \{d(1,3)\} = 4 + 3 + [\, 2 + 3 + 4\,] + 2 = 18$$

por lo que en este caso:

$$g'_c < g''_c$$

De la misma forma existen casos en los que la cota g''' puede estar por arriba o por abajo de las cotas g' y g''. Así que podemos concluir que no existe dependencia de orden entre estas cotas.

REGLA DE ACOTAMIENTO

Para cada secuencia (asociada a cada nodo del árbol de solución) $W_1,\ldots,W_k$, deben calcularse los valores de g', g'' y g''', como ya fue indicado anteriormente. De este modo, no importando cuál sea el orden en que los trabajos $W_{k+1},\ldots,W_n$ serán procesados, tenemos que el tiempo total del proceso L(W), una vez que son fijados los primeros k-trabajos, siempre es mayor o igual a g', g'' y g'''. Por lo que:

si G = máx {g', g'', g'''}

$\longrightarrow$ $L(W) \geq G$

DOMINACION

Siempre que en dos nodos figuren exactamente los mismos trabajos (aunque en diferente secuencia) debe declarase inactivo uno de los dos, aquel de mayor cota. Ya que la secuencia de duración minima que los complementa, será la misma para ambos y por lo tanto no podrá obtenerse una mejor solución en el nodo de mayor cota.

REGLA DE RAMIFICACION

El árbol de busqueda de la solución es ramificado al identificar aquellos nodos activos que tengan la menor cota G y generando en dicho nodo todas las posibles combinaciones en que puede ser complementada la trayectoria $W_1, W_2, \ldots, W_k$, representada por el nodo en cuestión. Posteriormente debemos acotar cada uno de los nodos recientemente creados, de acuerdo a la regla de acotamiento ya mencionada. Cuando se tenga perfectamente definida una secuencia $W_1, \ldots, W_k, W_{k+1}, \ldots, W_n$ y su cota sea menor o igual que las demás en el árbol, se considera resuelto el problema.

SECUENCIACION DE n TRABAJOS EN m MAQUINAS (caso más general)

El algoritmo anteriormente presentado, resuelve el problema de secuenciación para el caso en que se cuenta con tan sólo 3 máquinas de proceso, sin embargo el mismo algoritmo puede generalizarse para el caso de m máquinas con n trabajos, aplicando, en este caso, las mismas reglas para el desarrollo del árbol de búsqueda solución, excepto por que la regla de acotamiento debe generalizarse a este caso. Ya que cuando se cuentan con m máquinas de proceso, en cada nodo del árbol de soluciones debe calcularse m distintas cotas, de acuerdo a la siguiente formula:

$$g^{(j)}_{j=1,\ldots,m} = f_k(W_k, j) + \sum_{i=k+1}^{n} d(W_i, j) + \min_{i=k+1,\ldots,n} \left\{ \sum_{h=j+1}^{m} d(W_i, h) \right\}$$

La figura 29 nos muestra la manera de obtener cada cota $g^{(j)}$, al considerar que la secuencia $W_1,\ldots,W_k$ consumirá como mínimo un tiempo de proceso igual a lo que se lleva acumulado hasta la máquina M_j, esto es $f_k(W_k, j)$, más el mínimo tiempo que se llevaría al procesar todos los demás trabajos en las demás máquinas cuando cada uno de estos trabajos es procesado ininterrumpidamente en el resto de las máquinas.

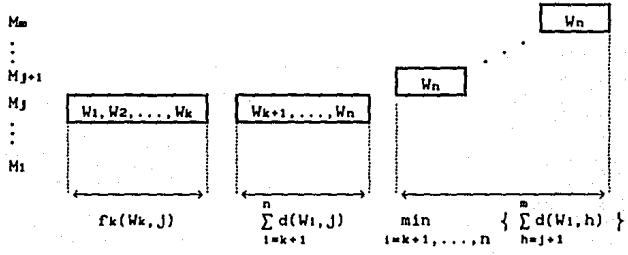
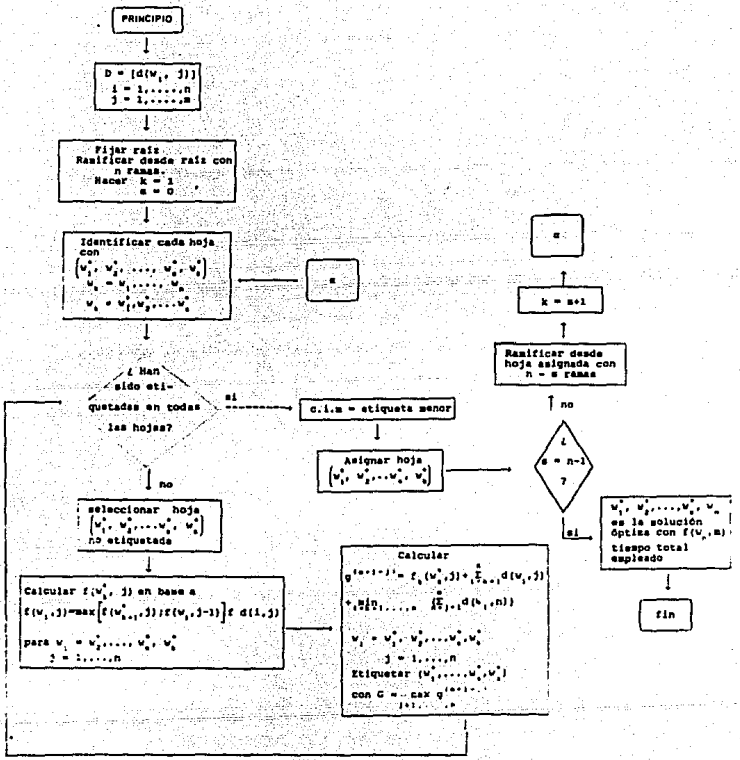


FIGURA 29

A continuación se muestra el algoritmo de ramificación y acotamiento que resuelve el problema de secuenciación para el caso que se tienen n trabajos a procesar en m máquinas.


PRINCIPIO
D = [d(v_i, j)]
i = 1,.....n
j = 1,.....m
Fijar raiz
Ramificar desde raiz con
n ramas.
Hacer k = 1
s = 0
Identificar cada hoja
con
Ramificar desde
hoja asignada con
n - s ramas
k = s+1
¿ Han
sido eti-
quetadas en todas
las hojas?
si
no
c.i.m = etiqueta menor
Asignar hoja
seleccionar hoja
no etiquetada
¿
s = n-2
?
si
no
es la solución
óptima con f(v_i,m)
tiempo total
empleado
fin
Calcular f(v_i, j) en base a
para
Calcular
Etiquetar
con G =

Ejemplo

Consideremos la siguiente matriz D, que muestra el tiempo en que el trabajo i es realizado por la máquina j.

$$D = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{ccccc} M_1 & M_2 & M_3 & M_4 & M_5 \end{array} \\ \begin{bmatrix} 3 & 5 & 8 & 4 & 2 \\ 7 & 3 & 6 & 9 & 1 \\ 5 & 4 & 3 & 8 & 4 \\ 9 & 3 & 4 & 7 & 3 \end{bmatrix} \end{array}$$

Se desea encontrar el orden en que los trabajos deben ser ejecutados, de manera que cada máquina realize los trabajos en el mismo orden y que además se minimice el tiempo total, cuando se llevan a cabo todos los 4 trabajos, en las 5 máquinas.

SOLUCION

El proceso de solución comienza con la ramificación del nodo raiz (conjunto de todas las secuenciaciones posibles de 4 trabajos), del cual surgen cuatro nuevas ramas, según sea fijado el primer elemento en la secuencia. De este modo, para cada una de las nuevas ramas, debe calcularse las cotas inferiores, de acuerdo a la función de acotamiento generalizada. Los resultados se muestran en la siguiente tabla:

W_1 / J		1	2	3	4	5	$g^{(5)}$	$g^{(4)}$	$g^{(3)}$	$g^{(2)}$	$g^{(1)}$	G
A	$f_1(A,J)$	3	8	16	20	22	30	45	39	32	41	45
B	$f_1(B,J)$	7	10	16	25	26	35	46	37	36	41	46
C	$f_1(C,J)$	5	9	12	20	24	30	41	36	34	41	41
D	$f_1(D,J)$	9	12	16	23	26	33	45	39	38	43	45

así que las cotas inferiores para cada nodo son:

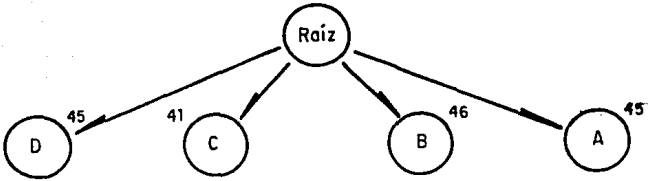


FIGURA 30

El proceso es continuado al ramificar el nodo que tiene la menor cota y calcular las cotas para los nuevos nodos. Estos resultados se muestran a continuación:

C	J	1	2	3	4	5	$g^{(5)}$	$g^{(4)}$	$g^{(3)}$	$g^{(2)}$	$g^{(1)}$	g
	$f_1(C,J)$	5	9	12	20	24						
CA	$f_2(A,J)$	3	14	22	26	28	32	43	42	34	41	43
CB	$f_2(B,J)$	12	15	21	30	31	36	43	39	37	41	43
CD	$f_2(D,J)$	14	17	21	28	31	34	42	41	39	43	43

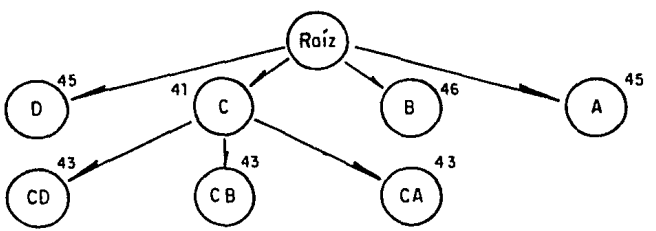


FIGURA 31

Del último árbol de búsqueda de la solución, se observa que existen 3 nodos terminales con cota inferior igual a 43 y que es el menor de los valores que aparecen en el árbol. De este modo, optamos por ramificar el nodo CA. Los resultados se muestran en la tabla y en la gráfica siguiente:

CA	J	1	2	3	4	5	$g^{(5)}$	$g^{(4)}$	$g^{(3)}$	$g^{(2)}$	$g^{(1)}$	g
	$f_2(A,J)$	8	14	22	20	24						
CAB	$f_3(B,J)$	15	18	28	37	38	41	47	42	35	41	47
CAD	$f_3(D,J)$	17	20	26	33	36	37	43	42	39	43	43

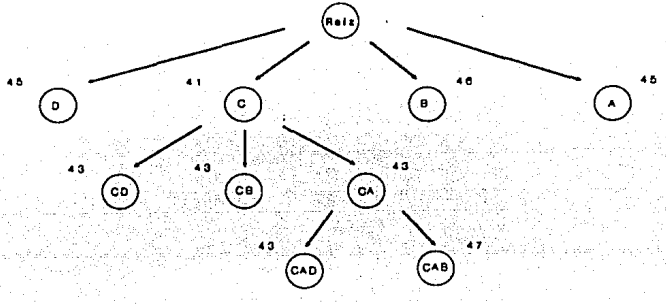


FIGURA 32

Puesto que no existe ninguna cota inferior a 43, CADB es una solución óptima. Y el tiempo total de proceso es:

CAD	$f_3(D,J)$	17	20	26	33	36
CADB	$f_4(B,J)$	24	27	33	42	43

Gráficamente, se observa:

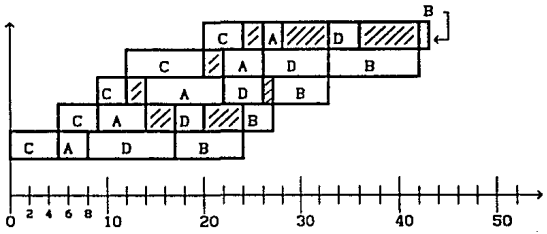


FIGURA 33

3.6. COMPLEJIDAD COMPUTACIONAL

3.6.1. Conceptos básicos

El concepto de complejidad computacional surge a raíz de la búsqueda de alguna herramienta para medir la eficiencia de los algoritmos. El problema de medir esta eficiencia no se había estudiado hasta antes de 1970, ya que desde el surgimiento de la investigación de operaciones, el interés se centraba en la búsqueda de algoritmos que resolvieran los problemas, más que en la forma de como lo resolvieran. Sin embargo, fue en 1971 Cook y en 1972 Karp, quienes introdujeron a la literatura este concepto, el cual busca clasificar los algoritmos en términos del nivel de requerimientos computacionales de los algoritmos (orden matemático del tiempo que necesita el algoritmo para resolver un problema dado).

El orden computacional es el concepto más utilizado cada vez que se quiere medir la eficiencia de un algoritmo y se define como la expresión matemática de la cantidad de cálculos requeridos, en función del tamaño de una instancia del problema (valores particulares de un problema). Por ejemplo, si un algoritmo requiere un número de cálculos que puede ser expresado como un polinomio de cuarto grado, entonces decimos que el orden es n^4 o simplemente $O(n^4)$, otros algoritmos pueden ser $O(n^2)$, $O(n\log n)$, $O(2^n)$, etc..

Con frecuencia, el orden de un algoritmo puede decirnos cuando conviene ser utilizado. Para ello supongamos que se cuenta con 3 algoritmos con los siguientes grados de complejidad:

$1^0.-$ $O(\log n) = 4 \log(n)$ *horas*
$2^0.-$ $O(n^5) = 10^{-5}\, n^5$ *minutos*
$3^0.-$ $O(2^n) = 10^{-6}\, 2^n$ *segundos*

Si comparamos el tiempo que consumen cada uno, para un problema de n=10 y para otro de n=100, obtenemos que:

Orden	$O(\log n)$	$O(n^5)$	$O(2^n)$
Función / unidades	4 log n *horas*	$10^{-5}\, n^5$ *minutos*	$10^{-6}\, 2^n$ *segundos*
n = 10	4 *horas*	1 *minuto*	0.001 seg. = 1 *milesi. de seg.*
n = 50	6.8 *horas*	3125 *min.* = 52 *horas*	1,125,899,907 seg. = 35.7 *años*
n = 100	8 *horas*	100,000 *min* = 2.3 *meses*	valor extremadamente grande

Así que para n suficientemente grande, el algoritmo logaritmico es preferido sobre el polinomial y éste a su vez sobre el exponencial.

Además, el tiempo de proceso para el algoritmo polinomial es razonable, aún para n grande (un problema de tamaño n=100 es bastante grande y difícil que se nos presente). De este modo se dirá que un algoritmo es eficiente si su complejidad computacional es del orden a lo más polinomial.

3.6.2. Ventajas y limitaciones de los métodos de ramificación y acotamiento

De los métodos más populares para resolver problemas de optimización discreta se encuentran los métodos de ramificación y acotamiento. En muchas aplicaciones prácticas de programación entera y optimización combinatoria, estos métodos han proporcionado resultados satisfactorios. Sin embargo estos métodos en muchos de los casos no son eficientes (su complejidad computacional es del orden exponencial), pero podemos decir que de los algoritmos existentes para la optimización discreta, los métodos de ramificación y acotamiento son de los más aceptables que existen (o son, de todos los malos, los menos malos; según se prefiera).

La ventaja de utilizar los métodos de ramificación y acotamiento, radica en lo flexible que son estos algoritmos, ya que al utilizar la técnica enumerativa de soluciones, es posible resolver cualquier problema de optimización discreta. Además la técnica puede interactuar con algunas otras técnicas de optimización, produciéndose algoritmos bastante aceptables; como el caso de la solución de problemas de programación entera, los cuales pueden ser resueltos mediante la técnica de ramificación y acotamiento conjuntamente con la utilización de una estrategia basada en la relajación a problemas de programación lineal y simultáneamente con el empleo de los métodos de planos de corte.

También es importante señalar que debido a la técnica enumerativa

de soluciones, los métodos de ramificación y acotamiento pueden ser programados en máquinas de proceso compartido (máquinas que pueden realizar varias tareas a la vez) o bien, resolver un mismo problema, al generar varias soluciones parciales del problema original (claro, de manera exahustiva) y optimizar cada solución en una máquina por separado; de este modo estaríamos reduciendo el tiempo que nos llevaría resolver todo el problema en una sola máquina.

3.6.3. Complejidad y experiencia computacional del Algoritmo de Balas

Lo conceptos básicos de complejidad computacional nos llevan a concluir que un algoritmo es eficiente si para cada instancia dada (problema con valores particulares), el tiempo de proceso para resolverlo es a lo más de orden polinomial. En el caso del algoritmo del Balas, es fácil verificar que el tiempo de proceso para resolver un problema con n-variables es del orden $O(2^n)$ (en el "peor" de los casos, para resolver el problema, es necesario analizar todas y cada una de las posibles soluciones), esto significa que el algoritmo de Balas es ineficiente. Por otro lado, en caso que fuera posible identificar la trayectoria que conduce a la solución óptima (serie de nodos que conducen a la solución óptima), el problema binario sería resuelto en tiempo polinomial. Esta característica, en teoría de la complejidad computacional, se conoce como: el problema es NP (polinomial no-determinista). Más aún dentro de la clase NP existen otra clase de problemas que son difíciles de resolver y que son aquellos problemas NP que pueden ser transformados, en tiempo polinomial, a otro problema NP; a esta clase se le conoce como NP-completa (ejemplo: el problema de programación binaria, el problema de satisfacibilidad, el problema del máximo ciclo, el problema del agente viajero, el problema del máximo conjunto independiente, el problema de la mochila ajustado

al modelo 0-1, el problema de la mochila ajustado el modelo de programación entera, el problema del cubrimiento con conjuntos de cardinalidad 3, etc.; revisar [7] y [8]). Los problemas NP-completos son problemas para los que no se conoce y probablemente no existan algoritmos polinomiales para resolverlos; cuando se halle un algoritmo polinomial, la teoria de la complejidad se habrá terminado.

Por otro lado, cuando el algoritmo de Balas es comparado con los algoritmos existentes para programación lineal, concluimos que el de Balas es un buen algoritmo, pero no tan bueno como el algoritmo que se obtiene de combinar la técnica de ramificación y acotamiento con una estrategia basada en el óptimo del problema de programación lineal relajado o con el método de planos de corte [4]. Sin embargo cuando el algoritmo de Balas es comparado con todos los algoritmos de enumeración implícita, Balas es considerado uno de los más importantes. Para confirmar esto, veamos el estudio reportado por Narula y Kindorf en 1979 [13].

En el estudio realizado por Narula y Kindorf ellos comparan el algoritmo de Balas con Hammer y Rudeanu (1968), Peterson (1967), Zion† (1972), Geoffrion (1969) y Zionts (1974) (todos ellos basados en la enumeración implícita de las soluciones), considerando problemas donde el número de variables es de 30 a 100 y cuya matriz A tiene una densidad igual a 0.2, 0.4, 0.6 y 0.8 (densidad(A)=$Pr[a_{ij}>0]$), así problemas como el de selección de proyectos de inversión de multi-periodos tiene una densidad cerrada de uno, mientras que el problema del cubrimiento puede tener densidades tan pequeñas como de 0.02 a 0.11). En estas condiciones, ellos concluyen que de los 6 algoritmos comparados, el algoritmo de Balas y el Algoritmo de Zionts son de los mejores, más aún, se recomienda usar el método de Balas para problemas en los que la densidad $d \leq 0.6$ y $n \geq 50$ y usar el método de Zionts para problemas con $d=0.8$ y/o $n \leq 30$.

Todo lo anterior nos lleva a concluir que el algoritmo de Balas es ineficiente si consideramos el tiempo que se lleva para resolver cualquier problema binario. Sin embargo, de los algoritmos que existen para programación binaria, Balas es uno de los mejores.

3.6.4. Complejidad y experiencia computacional del algoritmo de Little (El problema del agente viajero)

Comparando el algoritmo de Little (de ramificación y acotamiento) con los algoritmos existentes para asignación en general, concluimos que el de Little es ineficiente, ya que existen algoritmos que lo superan en eficiencia (asignación mediante al algoritmo simplex y el método hungaro). Sin embargo cuando Little es referido al problema del agente viajero, Little es de los más eficientes (o el menos malo de los algoritmos exactos; según se prefiera).

El problema del agente viajero por si mismo es un problema difícil de resolver, ya que cuando el problema es referido a n nodos o ciudades, éste se traduce a encontrar de un total de (n-1)! posibles recorridos hamiltonianos (permutando n nodos obtenemos n! posibilidades, de las cuales fijando uno de estos nodos obtenemos (n-1)! posibles recorridos hamiltonianos) aquel que optimice el recorrido total sobre los arcos.

Teóricamente el problema puede ser resuelto si generamos todos los (n-1)! recorridos y comparamos sus pesos totales. Sin embargo ésta no es una buena idea, ya que para un problema de 20 nodos se tiene un total $19! \cong 1.16*10^{17}$ posibles recorridos y nos llevaría años de trabajo el desarrollar cada uno de estos recorridos.

Por otra parte, para resolver de manera más "inteligente" el mismo problema, utilizamos algunos de los algoritmos existentes para

problemas de optimización combinatoria. Sin embargo aún no existe algoritmo que resuelva el problema de manera eficiente (con tiempo de convergencia polinomial) y la mayoría de los que se conocen requieren un tiempo de proceso que es exponencial respecto al número de nodos. En particular el algoritmo de ramificación y acotamiento aquí presentado, reduce drásticamente los efectos de la enumeración exahustiva de todas las posibles soluciones y por la técnica enumerativa que emplea, en el peor de los casos puede requerir un número exponecial de los cálculos y en algunos casos esto llega a ser aún mayor.

No obstante, la eficiencia de este algoritmo de ramificación y acotamiento sólo es superada por ciertos algoritmos de aproximación de soluciones (ejemplo: Algoritmo de Inserción [5]), los cuales tienen un tiempo de convergencia polinomial y aunque puede no producir una solución óptima, en algunos casos particulares puede proporcionar soluciones aceptables (muy próxima de la óptima). De este modo, si consideramos que para un problema de $n \geq 20$ se comienza a demandar un tiempo de proceso bastante grande (meses de trabajo), se sugiere utilizar Little para $n \leq 20$ y cuando $n \geq 20$ es mejor utilizar algún heurístico (algoritmo de inserción).

3.7. Discusión

Con lo ya presentado en este capítulo, apreciamos qué la técnica de ramificación y acotamiento es flexible en el sentido que puede ser utilizada para crear algoritmos para problemas más generales (Problemas de tipo binario) o bien algoritmos que resuelvan problemas más particulares, obteniendo, en estos últimos, mayor eficiencia, en cuanto a que reducen el consumo en el tiempo de proceso (algoritmos de asignación y secuenciación).

La contribución básica del capítulo 3 es la parte de Análisis de Sensibilidad para el algoritmo de enumeración implícita, la cual nos permite resolver un mismo problema, bajo ciertos cambios en los parámetros y aún cuando estas variaciones sean muy drásticas, los resultados en cuanto a tiempo de proceso se refiere, no han sido tan malos como para aquellos problemas que han sido reformulados y resueltos desde el principio. Para darse una mejor idea de esto, y aún cuando el ejemplo no es incluido [14], al haber experimentado con un problema de proyectos de inversión, involucrando 20 proyectos (variables) sobre 10 periodos (restricciones); un total de 1,091 soluciones parciales fueron examinadas en la fase solución y adicionalmente 1,143 soluciones parciales fueron examinadas en otros 5 estudios sobre proyectos de inversión (modificaciones en los parámetros del mismo problema). Lo cual contrasta mucho con el tremendo trabajo que se lleva resolver, desde un principio, las 6,515 soluciones parciales que se requieren analizar, para resolver los mismos 5 problemas separadamente, pero usando el óptimo original y una solución inicial en cada caso.

Analizando la eficiencia de los métodos de ramificación y acotamiento, concluimos que son ineficientes (su complejidad computacional en la mayoría de los casos es de tipo exponencial), pero de todos los algoritmos que existen para optimización combinatoria, los de ramificación y acotamiento son de los menos "malos" que existen. Así por ejemplo, el algoritmo de Balas es de los algoritmos más aceptables de todos los algoritmos de enumeración implícita y sólo es superado por algoritmos que utilizan la ramificación y acotamiento simultáneamente con el método simplex y los métodos de planos de corte. Así mismo, al comparar la eficiencia del algoritmo de Little con los demás algoritmos existentes para el problema general de asignación, concluimos que es ineficiente, ya que su eficiencia se ve claramente superarda por la del método húngaro y el algoritmo simplex especializado para asignación; sin embargo, el mismo

algoritmo de Little aplicado al problema del agente viajero, resulta ser de los mejores (o de los menos malos; según se prefiera), ya que en este caso su eficiencia sólo es superada por algoritmos heurísticos (algoritmo de inserción).

Sin embargo, aún cuando los algoritmos de ramificación y acotamiento resultan ser ineficientes, la ventaja de utilizar la técnica de ramificación y acotamiento radica en que es una técnica enumerativa de soluciones, lo cual hace posible que un mismo problema que es resuelto con un algoritmo cuya complejidad es $O(2^n)$, pueda ser resuelto al analizar m subproblemas pero en diferentes máquinas. De este modo, cuando para la solución de un mismo problema son utilizadas m máquinas (analizando un subproblema en cada máquina); el tiempo que consume el proceso de solución sería del orden $O(2^{n-m})$, reduciéndose drásticamente el orden, ya que si antes nos llevaba $10^{-6}*2^{50}$ *seg.* = 35.7 *años* resolver un problema con n=50 (esto es en el peor de los casos), ahora utilizando m=20 máquinas, el tiempo se reduce a $10^{-6}*2^{30}$ *seg.* = 18 *min.* o incluso a tan sólo un segundo cuando son utilizadas m=30 máquinas; ésta es la ventaja de utilizar los métodos de ramificación y acotamiento.

4. MODELOS CLASICOS

4.1. EL PROBLEMA DE PROYECTOS DE INVERSION

En el caso simple, el problema consiste en escoger de un total de n posibilidades de inversión (independientes), aquellos que maximizan el beneficio total de las inversiones, pero respetando la restricción sobre el capital disponible.

Para modelar el problema, sea B_j el beneficio redituado del proyecto J, A_j el costo de inversión en el proyecto J y C el total de capital disponible. Entonces, X_j es 0 (ó 1) cuando el proyecto J es rechazado (ó elegido) para invertir. Así, el modelo básico es:

$$\text{Maximizar} \quad \sum_{j=1}^{n} B_j \, X_j$$

$$\text{sujeto a} \quad \sum_{j=1}^{n} A_j X_j \leq C$$

$$X_j = 0 \text{ ó } 1 \qquad (J = 1,2,\ldots,n)$$

Un caso más generalizado se obtiene, al considerar la misma situación para varios períodos, presentandose diversos requerimientos de inversión y capital disponible, para cada período. En esta situación, los costos no-negativos del proyecto J en el período t es denotado por A_{tj} (j=1,...,n , t=1,...,T) y el capital disponible en el período t es C_t (t=1,...,T). Así, el problema de proyectos de inversión de multi-períodos es:

$$\text{Maximizar} \quad \sum_{j=1}^{n} B_j X_j$$

$$\text{sujeto a} \quad \sum_{j=1}^{n} A_{tj} X_j \leq C_t \quad (t = 1,\ldots,T)$$

$$X_j = 0 \text{ ó } 1 \quad (J = 1,2,\ldots,n)$$

Existen muchas otras extensiones al modelo, una de ellas ocurre cuando los n proyectos posibles son particionados en subconjuntos ajenos n_1 ,.... n_p (p≤n) y solamente un proyecto de cada subconjunto puede ser seleccionado. Esta condición es típica en un problema de la construcción de una carretera entre dos ciudades, cuando se tienen varias rutas u opciones de construcción, debiendo construirse una única ruta entre ambas ciudades. La anexión de este requerimiento está representada por:

$$\sum_{j \in n_i} X_j = 1 \qquad (i=1,\ldots,p \leq n)$$

4.2. El problema del cubrimiento

Considere un conjunto $I=\{1,\ldots,m\}$ y una clase $P=\{P_1,\ldots,P_n\}$, donde $P_j \subseteq I$, $j \in J = \{1,\ldots,n\}$. Un subconjunto $J^* \subseteq J$ define un cubrimiento de I si:

$$\bigcup_{j \in J} P_j = I$$

Supongamos un costo $C_j > 0$, asociado a cada $j \in J$; entonces el costo total del cubrimiento J es:

$$\sum_{j \in J} C_j$$

El problema del cubrimiento consiste en la búsqueda de un cubrimiento de costo mínimo y su modelo es:

$$\text{Minimizar} \quad \sum_{j=1}^{n} C_j \, X_j$$

$$\text{sujeto a} \quad \sum_{j=1}^{n} A_{ij} \, X_j \geq 1 \quad (i = 1,2,\ldots,n)$$

$$X_j = 0 \text{ ó } 1 \quad (j = 1,2,\ldots,n)$$

donde

$$X_j = \begin{cases} 1 & \text{si } j \text{ esta en el cubrimiento} \\ 0 & \text{en otro caso} \end{cases}$$

$$A_{ij} = \begin{cases} 1 & \text{si } i \in P_j \\ 0 & \text{en otro caso.} \end{cases}$$

Algunas generalizaciones del modelo son planteadas, cuando las restricciones principales, son consideradas como:

$$\sum_{j=1}^{n} A_{ij} \, X_j \geq b_i \qquad i=1,\ldots,m \text{ donde } b_i \in Z_+$$

lo que se traduce en cubrir el elemento i de I, b veces.

Por otro lado, la restricción $C_j > 0$ $j=1,\ldots,n$, no representa mayor problema, pues en caso de tener algunas variables con un costo negativo, como el problema es de minimizar y las restricciónes del tipo mayor o igual, dichas variables tomarían (en el óptimo) el valor uno, por lo que bastaría eliminar las restricciones en las que dichas variables aparecen y optimizar el problema en el que todos los costos son no-negativos.

EJEMPLO 1

Suponga que se tienen m tareas a realizar y para la realización de las mismas es posible contratar personal calificado. Hay n candidatos, a contratación , cada uno de los cuales puede realizar algunas de las tareas en función de sus capacidades. Para cada candidato se conocen las tareas que puede realizar y el costo por prestar sus servicios. La naturaleza de las tareas permite el abordar las mismas entre un grupo cualquiera de individuos.

El problema es determinar que candidatos deben ser contratados a un costo mínimo, de modo que se cubran todas las tareas.

I={Tarea 1, Tarea 2, ..., Tarea m}

P={Candidato 1, Candidato 2, ..., Candidato n}

y cada Candidato puede realizar algún conjunto de tareas de I.

EJEMPLO 2

Se quiere hacer un reconocimiento superficial, sobre el nivel de producción de determinados sectores de producción (Azucarero, siderúrgico, de servicio, etc.). Las unidades correspondientes a cada sector, se encuentran distribuidas en los diferentes

municipios de la población. Cada municipio tiene asociado un costo, necesario para el levantamiento de la información. Y se quiere determinar que municipios incluir en la muestra, para lograr obtener información, de por lo menos cada sector de producción.

$I=\{$Azucarero, Siderúrgico, de Servicios, etc.$\}$

$P=\{$Municipio 1, Municipio 2, ..., Municipio n$\}$

y en cada Municipio se encuentran localizados uno o varios sectores de la producción.

4.3. EL PROBLEMA DE LA MOCHILA

Un explorador tiene que realizar una larga travesía y para ello, necesita llevar consigo, una serie de artículos que le son necesarios en su viaje. Sin embargo, para llevar su equipo, el sólo cuenta con una mochila que tiene una capacidad total de K unidades de volumen.

Suponiendo que el artículo i tiene un volumen K_i y un valor de utilidad igual a V_i, el explorador necesita saber que artículos llevar consigo, de modo que obtenga la máxima utilidad posible. Así, el modelo matemático para el problema de la mochila es:

$$\text{Minimizar} \quad \sum_{j=1}^{n} V_i \; X_i$$

$$\text{sujeto a} \quad \sum_{i=1}^{n} K_i \; X_i \leq K$$

$$X_i = 0 \text{ ó } 1 \qquad (i = 1,2,\ldots,n)$$

donde

$$X_i = \begin{cases} 1 & \text{si el artículo } i \text{ se incluye} \\ 0 & \text{si el artículo } i \text{ no se incluye} \end{cases}$$

V_i = Valor de utilidad del artículo i.

K_i = Volumen del artículo i.

4.4. EL PROBLEMA DE ASIGNACION

Supongase que tenemos n tareas por realizar y m empleados que pueden realizar cada tarea. Cada empleado i realiza la tarea j con un costo C_{ij}

$$\begin{array}{c} \\ \text{EMPLEADOS} \end{array} \begin{array}{c} \text{TAREAS} \\ \begin{bmatrix} C_{11} & C_{12} & C_{13} & \cdots & C_{1n} \\ C_{21} & C_{22} & C_{23} & \cdots & C_{2n} \\ C_{31} & C_{32} & C_{33} & & C_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ C_{m1} & C_{m2} & C_{m3} & & C_{mn} \end{bmatrix} \end{array}$$

El problema es asignarle un empleado a cada tarea, de modo que se realizen todas las tareas, pero minimizando costos. Así el modelo matemático para el problema de asignación es:

$$\text{Maximizar} \quad \sum_{j=1}^{n} \sum_{j=1}^{n} C_{ij} \; X_{ij}$$

$$\text{sujeto a} \quad \sum_{j=1}^{n} X_{ij} = 1 \qquad (i = 1, 2, \ldots, m)$$

$$\sum_{i=1}^{n} X_{ij} = 1 \quad (i = 1,2,\ldots,m)$$

$$X_{ij} = 0 \text{ ó } 1$$

donde $X_i = \begin{cases} 1 & \text{si a la tarea J se le asignó al empleado 1} \\ 0 & \text{en otro caso} \end{cases}$

EJEMPLO 1

En una competencia de relevos de 400 metros incluye a 4 diferentes nadadores, quienes nadan sucesivamente 100 metros de dorso, de pecho, de mariposa y libre.

Un entrenador tiene 6 nadadores muy veloces, cuyos tiempos esperados (en segundos) de manera individual son

	EVENTO 1 (dorso)	EVENTO 2 (Nado de Pecho)	EVENTO 3 (Mariposa)	EVENTO 4 (libre)
Nadador 1	65	73	63	57
Nadador 2	67	70	65	58
Nadador 3	68	72	69	55
Nadador 4	67	75	70	59
Nadador 5	71	69	75	57
Nadador 6	69	71	66	59

¿ Como deberá el entrenador programar sus relevos, de modo que el equipo realize su mejor tiempo en la competencia ?.

EJEMPLO 2

Un bufete de abogados ha aceptado 5 nuevos casos, cada uno de los cuales puede ser llevado adecuadamente por cualquiera de los 5 asociados más recientes. Debido a la diferencia en experiencia y práctica, el tiempo que demora el llevar un mismo caso, varía dependiendo del abogado al que se le asigne el caso.

Uno de los asociados más experimentados, ha estimado las necesidades de tiempo (en horas) como sigue:

	CASO 1	CASO 2	CASO 3	CASO 4	CASO 5
Abogado 1	145	122	130	95	115
Abogado 2	80	63	85	48	78
Abogado 3	121	107	93	69	95
Abogado 4	118	83	116	80	105
Abogado 5	97	75	120	80	111

Determinar la forma óptima de asignar los casos a los abogados, de manera que cada uno de ellos se dedique a un caso diferente y que el tiempo total de horas empleadas sea mínimo.

4.5. UN PROBLEMA DE SECUENCIACION

Suponga, en el caso simple, un taller puede efectuar un sólo tipo de trabajo a la vez y un conjunto de n ordenes diferentes de trabajo, son contratados a la vez en una fecha base. Y suponiendo que cada trabajo i, es contratado para ser entregado en g_i dias a

partir de la fecha base. Además cada trabajo tiene una duración de trabajo d_i y en caso de retraso a la fecha de entrega, se tiene una multa de p_i pesos por dia de retraso despues de los g_i dias estipulados.

¿ Cuál debe ser el orden en que los trabajos deben ser realizados, de modo que el taller incurra en la mínima multa total, ocasionada por la entrega retrasada de todos y cada uno de los trabajos ?.

EJEMPLO

Considere el caso en que 6 trabajos son contratados por el taller en una misma fecha base.

TRABAJO i	DIA DE ENTREGA g_i	DURACION d_i	MULTA POR DIA DE RETRASO P_i
1	2	5	$ 5
2	4	4	$ 4
3	8	3	$ 2
4	12	5	$ 1
5	13	2	$ 7
6	17	7	$ 1

Así por ejemplo, si la secuencia de trabajos es: hacer primero el trabajo 1, después el trabajo 2 y así sucesivamente hasta la conclusión con la orden 6; entonces se incurre en una multa de $99, como lo muestra la figura 34.

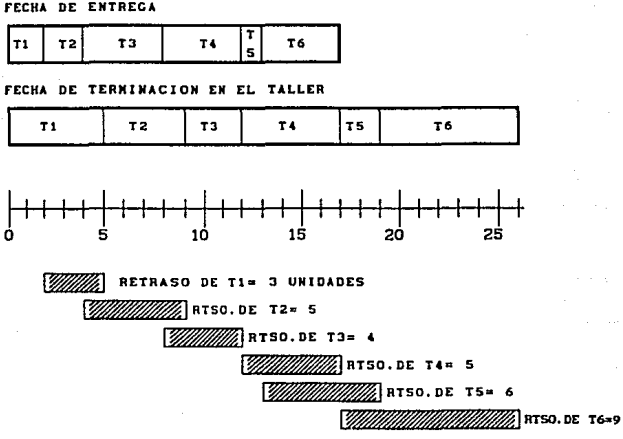


FIGURA 34

El problema es entonces, determinar el orden en que deben ser realizados los trabajos, de modo que se minimize la multa total ocasionada por los días que se retrasa la entrega de cada trabajo.

La formulación de este problema (sin mucho detalle) en el modelo de programación cero-uno es:

$$\text{Minimizar } Z = \sum_{i=1}^{6} P_i \sum_{t=g_i+1}^{T} \sum_{q=d_i}^{t-1} Y_{iq}$$

sujeto a

$$\sum_{t=d_i}^{T} Y_{it} = 1 \qquad i = 1,2,\ldots,6$$

$$\sum_{i=1}^{6} \sum_{q=t}^{t+d_i-1} Y_{iq} = 1 \qquad t = 1,2,\ldots,T$$

$$Y_{it} = 0 \text{ ó } 1 \qquad , T = \sum_{i=1}^{6} d_i = 26$$

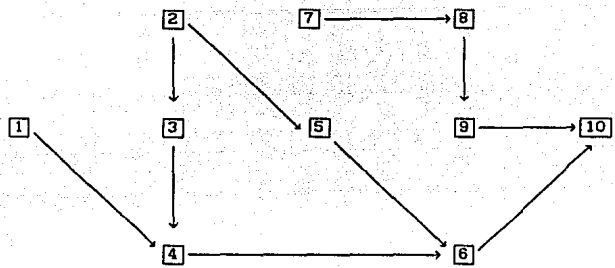
$$Y_{it} = \begin{cases} 1 \text{ si el trabajo } i \text{ se completa al final del período } t\,, \; i=1,\ldots 6 \quad t=1,\ldots,T \\ 0 \text{ si el trabajo } i \text{ no se completa al finalizar el período } t \end{cases}$$

4.6. BALANCE DE LINEAS DE PRODUCCION

Suponga que se tienen n actividades a realizar en una linea de producción, en donde existen relaciones de precedencia entre algunas actividades. Suponiendo que se cuenta con un grupo de trabajadores que pueden ser contratados para realizar cualquier actividad, entonces el problema consiste en contratar el mínimo número de trabajadores (en base a la tasa de producción que se desea obtener) a fin de que cada actividad, a realizar en turno, tenga asignado un trabajador.

EJEMPLO

TRABAJO i	DURACION (en minutos) D_i	TRABAJOS DE PRECEDENCIA
1	3	
2	5	
3	2	2
4	6	1,3
5	9	2
6	3	4,5
7	4	
8	1	7
9	3	8
10	5	6,9



Se supone que la tasa de producción es de 4 unidades por hora o más (entendiendose por una unidad, aquella que se logra con la realización de todas las 10 actividades).

Definamos las variables

$$X_{ij} = \begin{cases} 1 & \text{si la actividad } i \text{ es realizada por el trabajador } j \\ 0 & \text{si la actividad } i \text{ no es realizada por el trabajador } j \end{cases}$$

$$T_j = \begin{cases} 1 & \text{si el trabajador } j \text{ fue ocupado en la línea de producción} \\ 0 & \text{si el trabajador } j \text{ no fue ocupado en la línea de producción} \end{cases}$$

El objetivo es minimizar el número de trabajadores a ocupar en la línea de producción. Pero es claro que contrataremos como máximo 10 trabajadores (ocupando un trabajador para cada actividad), satisfaciendose automáticamente las actividades en la linea de producción. De este modo, la función objetivo es:

$$\text{Minimizar } Z = \sum_{j=1}^{10} T_j$$

sujeto a

(1) Cada actividad debe efectuarse

$$\sum_{j=1}^{10} X_{ij} = 1 \qquad i=1,2,\ldots,10$$

(2). Debido a que necesitan obtenerse al menos 4 unidades por hora (realizar 4 veces todas las tareas en una hora) o equivalentemente obtener al menos una unidad cada 15 minutos. Debemos entonces restringir a que cada trabajador no debe ser ocupado en más de 15 minutos, cuando se está produciendo una unidad. Esto es:

$$\sum_{i=1}^{10} D_i \, X_{ij} \leq 15 \qquad j=1,2,\ldots,10$$

(3) Cada vez que, un trabajo K debe hacerse antes qué el trabajo m (k≠m), es necesario agregar el conjunto de restricciones:

$$X_{k1} \geq X_{m1}$$

$$X_{k1} + X_{k2} \geq X_{m2}$$

$$X_{k1} + X_{k2} + X_{k3} \geq X_{m3}$$

. . . . . .

$$X_{k1} + X_{k2} + \ldots + X_{k10} \geq X_{m10}$$

De esta forma, la actividad m es asignada al trabajador i (i=1, ..., 10) sólo cuando la actividad k ya fue asignada a alguno de los trabajadores 1, 2, ..., i-1 .

(4) Por último, con las restricciones ya escritas, es posible que:

$$\sum_{j=1}^{10} X_{ij} = 0$$ (puede asumir algún valor entre 1 y 10)

y sin embargo, ninguna actividad es asignada al trabajador j. Por ello es necesario pedir que:

$$\sum_{i=1}^{10} X_{ij} \leq 10\, S_j \qquad j=1,2,\ldots,n$$

Así cuando el trabajador j no realiza ninguna actividad ($S_j = 0$) se tiene que:

$$\sum_{i=1}^{10} X_{ij} = 0$$

de otro modo, dicho término tomaría algún valor entre 1 y 10, dependiendo del número de actividades que se le asignaron al trabajador j.

4.7. Un problema combinatorio

Dado un cubo de 3x3x3 conteniendo 27 celdas, el problema es colocar en cada celda una bola blanca o una bola negra de tal forma que se minimize el número de "líneas rectas" en el cubo, que contengan bolas del mismo color. Entendiendo por "línea recta" un renglón de 3 celdas contiguas incluyendo todas las posibles diagonales (En total, el cubo contiene 49 líneas rectas). Y suponga que no existe restricción sobre el número de bolas blancas y bolas negras disponibles.

$$X_i = \begin{cases} 1 & \text{si la celda } i \text{ es ocupada por una bola blanca} \\ 0 & \text{si la celda } i \text{ es ocupada por una bola negra} \end{cases}$$

Si numeramos cada celda del cubo, entonces para cada una de las 49 líneas en el cubo (por ejemplo las celdas 1,2 y 3), es necesario agregar dos restricciones del siguiente tipo:

$$X_1 + X_2 + X_3 - Y \leq 2$$

$$X_1 + X_2 + X_3 + Y \geq 1$$

donde Y es una variable asociada a cada línea recta del cubo (en este caso la línea compuesta por las celdas 1, 2 y 3 del cubo) y cuyos valores son cero o uno, dependiendo de los valores de las variables asociadas a las celdas que componen dicha línea. En este caso:

$X_1=X_2=X_3=1$ ó $X_1=X_2=X_3=0 \longrightarrow Y=1$, de otro modo $Y=0$.

De esta forma, como la función objetivo es minimizar la suma de todas las variables Y_s. Al tratar de hacer ceros en la mayoría de las variables Y_s, se tendrá que las restricciones serían reducidas, en su mayor parte a:

$$X_1 + X_2 + X_3 \leq 2$$

$$X_1 + X_2 + X_3 \geq 1$$

lo que implica que en las celdas 1,2 y 3 existen al menos 1 ó 2 bolas blancas y entonces dicha línea recta puede tener todas las bolas del mismo color.

Por otro lado, en caso que Y=1, las restricciones son reducidas a

$$X_1 + X_2 + X_3 \leq 3$$

$$X_1 + X_2 + X_3 \geq 0$$

lo que no restringe el tipo de bolas a colocar en las celdas 1, 2 y 3 (puede ser que todas sean del mismo color o en el mejor de los casos, que sean de distinto color). Pero al minimizar la función objetivo, pocas líneas (tanto como fuese posible) tendran celdas con bolas del mismo color.

Nótese que un valor de 4 en la función objetivo, significa que a lo más 4 líneas presentarán bolas del mismo color.

Por lo tanto, el modelo en este caso es:

$$\text{Minimizar } Z = \sum_{j=1}^{49} Y_j$$

sujeto a

$$X_1 + X_2 + X_3 - Y_1 \leq 2$$

$$X_1 + X_2 + X_3 + Y_1 \geq 1$$

$$\vdots$$

$$X_1 + X_2 + X_3 - Y_{49} \leq 2$$

$$X_1 + X_2 + X_3 + Y_{49} \geq 1$$

$$X_i \, , \, Y_j = 0 \text{ ó } 1$$

$$i = 1,2,\ldots,27$$
$$j = 1,2,\ldots,49.$$

4.8. UN PROBLEMA DE EXTRACCION MINERA

El problema consiste en excavar, dentro de un área y en cierto orden permisible, con el propósito de obtener mineral preciado. Por cuestiones prácticas, tanto el mineral como la corteza que están encima de la mina, son divididas en bloques (normalmente de 50 pies cúbicos). Cada bloque tiene un cierto ingreso neto, obtenido de restar el costo de extracción, a la cantidad obtenida después del refinamiento y venta del mineral. Para bloques que están cerca de la superficie (corteza que está encima de la mina) el ingreso neto es frecuentemente negativo, pero normalmente es positivo conforme los bloques están más profundos y más cercanos a la parte medular de la mina.

En estas condiciones, el problema es saber que bloques extraer, de modo que se obtenga el mayor ingreso posible, pero cuidando que la excavación de un bloque es posible solamente si son removidos los bloques que están sobre éste. La figura 36 muestra un ejemplo particular, en donde el bloque 1 puede ser extraido, previo a la extracción de los bloques 2 y 3.

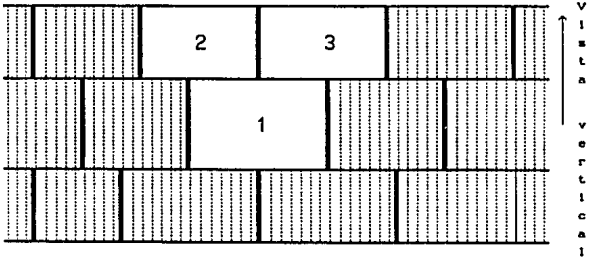


FIGURA 36

$$X_i = \begin{cases} 1 & \text{si el bloque } i \text{ será excavado} \\ 0 & \text{en otro caso} \end{cases}$$

Y supongamos que R_i es el ingreso neto estimado para el i-ésimo bloque (en algunos casos es negativo).

La función objetivo es entonces:

$$\text{Maximizar} \quad \sum_{j=1}^{n} R_i X_i$$

pero respetando la restricción de que cada bloque (excepto los que están sobre la superficie) puede ser extraido solamente si los que están arriba de éste son también extraidos. Así suponiendo, el caso que los bloques 2,3,4 y 5 se sobreponen al bloque 1, entonces debe agregarse la restricción:

$$X_2 + X_3 + X_4 - X_5 - 4 X_1 \geq 0$$

4.9. Un problema de diseño logico

El problema consiste en construir un sistema lógico, que responda a una operación lógica preestablecida, dadas ciertas entradas posibles. Así, un sistema tiene una salida y un número de entradas posibles, cada una de la cuales puede asumir los valores 0 ó 1.

Dada una función lógica que el sistema debe realizar, el problema es referido a construir un sistema de salida a base de compuertas NOR (función lógica $\overline{A \vee B}$) para ejecutar la función lógica preestablecida.

Una compuerta NOR tiene 2 entradas (A y B) y una salida, de acuerdo a la siguiente tabla de verdad:

TABLA 1

ENTRADAS		SALIDA
A	B	
0	0	1
0	1	0
1	0	0
1	1	0

Un "circuito" está compuesto por la conexión de varias compuertas NOR, de acuerdo al siguiente diagrama:

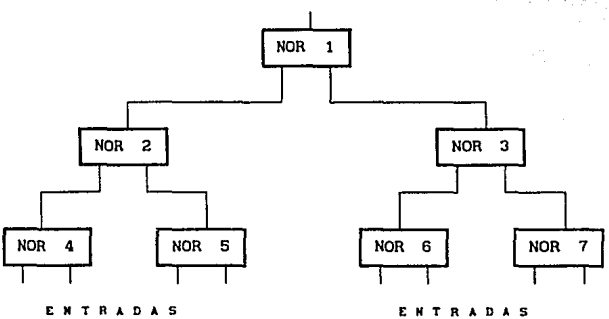


FIGURA 37

Una entrada externa (A o B) puede ser conectada a cualquiera de las entradas del circuito, incluso para más de una compuerta NOR.

Además, la salida de algún NOR puede ser guiada por más de una entrada, pero más de una salida de los NOR no puede ser guiada por la misma entrada.

Así el problema consiste en diseñar un circuito que pueda realizar una función lógica preestablecida, utilizando el mínimo número de compuertas NOR.

Considere, por ejemplo, el problema de construir un circuito de compuertas NOR con 2 entradas para poder realizar la función lógica de la tabla 2.

TABLA 2

ENTRADAS		SALIDA
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

Para simplificar el modelo, es necesario asumir que, el circuito óptimo a lo más contendrá tantos NOR's como lo indica la figura 37. En estas condiciones, definimos las siguientes variables de decisión:

$$S_i = \begin{cases} 1 & \text{si la compuerta del NOR } i \\ & \text{es utilizada. } i=1,\ldots,7 \\ 0 & \text{en otro caso} \end{cases}$$

$$T_{11} = \begin{cases} 1 & \text{si la entrada externa A es conectada a la compuerta del NOR 1.} \\ 0 & \text{en otro caso} \end{cases}$$

$$T_{12} = \begin{cases} 1 & \text{si la entrada externa B es conectada a la compuerta del NOR 1.} \\ 0 & \text{en otro caso} \end{cases}$$

X_{ij} = La salida para la compuerta i especificada por la combinación de entradas dadas en el j-ésimo renglón de la tabla 2.

La función objetivo es:

$$\text{Minimizar} \quad Z = \sum_{i=1}^{7} S_i$$

Las restricciónes impuestas son:

(i). Las entradas externas A y B son conectadas en el NOR i, sólo cuando la compuerta i forma parte del circuito.

$$T_{i1} + T_{i2} \leq 2\, S_i \qquad i=1,\ldots,7$$

(ii). Para las compuertas 1, 2 y 3 de la figura 37, si la compuerta tiene una (o dos) entradas externas, entonces solamente una (o ninguna) compuerta NOR alimentará a dicha compuerta.

$$S_2 + S_3 + T_{11} + T_{12} \leq 2$$

$$S_4 + S_5 + T_{21} + T_{22} \leq 2$$

$$S_6 + S_7 + T_{31} + T_{32} \leq 2$$

(iii). La salida para la i-ésima compuerta NOR debe ser la función lógica correcta (NOR), para cualquier combinación de entradas a la compuerta i.

Para ello definamos los parámetros:

α_{1j} = El valor de la entrada externa A según el j-ésimo renglón de la tabla 2.

α_{2j} = El valor de la entrada externa B según el j-ésimo renglón de la tabla 2.

Entonces la salida X_{ij}, para la compuerta i, dada la combinación de entradas especificadas en el i-ésimo renglón de la tabla 2, es correcta (según la función lógica NOR) si satisface:

$$\alpha_{1j}*T_{i1} + \alpha_{2j}*T_{i2} + X_{j1} + X_{k1} + 2X_{i1} \leq 2$$

$$-S_i + \alpha_{1j}*T_{i1} + \alpha_{2j}*T_{i2} + X_{j1} + X_{k1} + X_{i1} \geq 0$$

$$i=1,\ldots,7 \quad , \quad i=1,2,3,4$$

donde j y k son los números de las compuertas cuya salida está conectada a la compuerta i.

De este modo, $X_{11} = 1$ sólo cuando $X_{j1} = X_{k1} = \alpha_{11} \cdot T_{11} = \alpha_{21} \cdot T_{12} = 0$ es decir, cuando las entradas a la compuerta i son ambas cero (pudiendose conectar, en este caso, las entradas A y B a la compuerta i, siempre que $\alpha_{11} = 0$ y $\alpha_{21} = 0$).

En una compuerta particular existe una señal de salida igual a 1, para cualquier combinación de entradas A y B; siempre que dicha compuerta es contemplada para formar parte del circuito.

Esto es representado en el modelo, al imponer las siguientes restricciones:

$$4S_i - X_{11} + X_{12} + X_{13} - X_{14} \geq 0 \qquad i=1,\ldots,7$$

(iv) Adicionalmente, para pedir que el circuito establecido efectue la función lógica correcta, de acuerdo a la tabla 2; es necesario instanciar las variables X_{1j} de la siguiente manera:

$$X_{11} = 0 \,, \quad X_{12} = 1 \,, \quad X_{13} = 1 \,, \quad X_{14} = 0$$

ya que de esta forma, no importa donde conectemos las entradas externas A y B, la salida por la compuerta 1 (salida final) será el valor correcto según la función lógica mostrada en la tabla 2.

Así, el modelo completo para el problema de diseño lógico, involucra 73 restricciónes y 45 variables, todas del tipo cero-uno. Y la solución óptima, que se obtiene de aplicar la técnica de ramificación y acotamiento, es la mostrada en la figura 38.

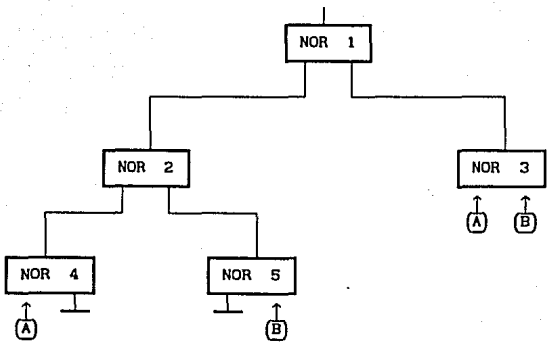


FIGURA 38

CONCLUSIONES

La Técnica de Ramificación y Acotamiento es útil cada vez que necesitamos resolver problemas de optimización discreta, donde una función va a ser maximizada o minimizada sobre un conjunto finito de alternativas, de tal forma, que el proceso de solución puede ser realizado con la enumeración de cada alternativa, pero de manera "inteligente".

La Técnica es flexible en el sentido de que puede ser utilizada para crear algoritmos generales (algoritmos de enumeración implícita) o bien algoritmos que resuelven problemas más particulares, logrando mayor eficiencia en cuanto a que reducen el consumo en el tiempo de proceso (algoritmos de asignación y secuenciación).

La técnica es eficiente, si la comparamos con todos los algoritmos que existen para programación discreta, más aún en el caso del algoritmo de enumeración implícita, la eficiencia se ve

incrementada al ser implementada la parte de análisis de sensibilidad , lo que permite resolver un mismo problema, bajo ciertas variaciones en el valor de los parámetros y aún cuando estas variaciones sean muy drásticas, los resultados en cuanto a tiempo de proceso se refiere, no son tan malos como para aquellos problemas que han sido reformulados y resueltos desde el principio.

Es justo mencionar, además, que los algoritmos de ramificación y acotamiento pueden ser programados en máquinas de proceso compartido, esto es, el análisis de las alternativas de solución, es realizada en procesos separados pero efectuados de manera simultánea y coordinada, reduciendo así, el tiempo desde que es iniciado el proceso hasta que es encontrada la solución óptima.

No obstante, después de todo, hay que tomar muy en cuenta que el objetivo básico al desarrollar un modelo de programación discreta de un problema operacional, es poder predecir cual sería la solución óptima, dadas las condiciones iniciales del problema modelado. Pero teniendo mucho cuidado de no abusar de la herramienta matemática, ya que en muchas ocasiones, cuando un problema es difícil de modelar, inmediatamente se pretenda hacer suposiciones tales que, permitan ajustar (en cualquier modo) nuestro problema al modelo metamático. Es por ello que debemos refrenar nuestra libertad para asumir suposiciones, ya que sólo deben permanecer aquellas apropiadas y que permitan desarrollar modelos que capten los aspectos esenciales del problema y sobre todo que nos produzca un modelo en el cual las soluciones puedan ponerse en práctica.

Por último, suponiendo que la modelación es realizada bajo todas las suposiciones pertinentes y que posteriormente el modelo es resuelto de manera extraordinariamente rápida; no debemos perder de vista, que dichas soluciones deben ser comprensibles a las personas responsables del proceso que se está estudiando y que

frecuentemente, estas soluciones no determinan de manera definitiva, las acciones a realizar para la solución del problema y más bien sólo se utilizan como indicadores para el tomador de decisiones, quien al estudiar otros factores que no pudieron ser contemplados en los modelo (eventos fortuitos, políticas a nivel empresarial, políticas a nivel estatal, reglamentos de tipo laboral, reglamentos de tipo produtivo o capacidad de recursos, entre otros que son muy difíciles de modelar) sea el que finalmente decida las acciones a realizar para la solución del problema en cuestión. Y es en este punto donde puede ser extendido el presente trabajo, aunque si se prefiere seguir buscando una mejora en cuanto a la eficiencia de los algoritmos aquí mostrados, es necesario seguir desarrollando alguna parte de análisis de sensibilidad para los algoritmos de asignación y secuenciación, así como los programas computacionales respectivos.

BIBLIOGRAFIA

Libros

1.- Boffey, T.B.: *Graf Theory in Operations Research*, London Macmillan 1982.

2.- Hayhurts, George: *Mathematical Programming Applications*, New York 1987.

3.- Fco. Jauffred M., Alberto Moreno Bonett, J. Jesús Acosta: *Métodos de optimización -programación lineal y gráficas-* Edit. Representaciones y Servicios de Ingeniería S.A., México 1980.

4.- Katta Murty: *Linear and combinatorial programing*, New York, J. Wiley, 1976.

5.- Kalvelagen Erwin, Henk C. Tijms: *Exploring operation research and statistics in micro lab*, New Jersey, 1990.

6.- Linwood A.Johnson, Douglas C. Montgomery: *Operations Research in Production Planning, Scheduling and Inventory Control*, New York, J. Wiley, 1974.

7.- Nemhauser George L., Laurence A. Wolsey: *Integer and combinatorial optimization*, J Wiley & Sons, 1988.

8.- Papadimitriou Christos, Steiglitz Kenneth: *Combinatorial optimization -algorithms and complexity-* Englewood, Prentice-Hall.

9.- Syslo Mciej M., Narsingh Deo, Janusz S. Kowalik: *Discrete Optimization Algorithms with Pascal Programs*, 1983.

Articulos

10.- E. Balas: *A note on the Branch-and-Bound Principle*, Operation Research, Vol. 16, 442-445 (1968).

11.- Carlier, J. and Pinson, E.: *An Algorithm for Solving the Job Shop Problem*, Management Science, Vol. 35, No. 2 (1989).

12.- González Pérez Jorge L., Allende Sirla M., *El Problema del Suministro*, Investigación Operacional, Vol II, No. 1, 6-45 (1981).

13.- Narula , S.C., and J.R. Kindorf, *Linear 0-1 programming: A comparison of implicit enumeration algorithms*, Comput, Operations Research, Vol. 6, (1979) 47-51.

14.- Roodman Gary M.: *Postoptimality Analysis in zero-one Programming by implicit enumeration*.

15.- L.Schrage and L. Wolsey: *Sensitivity Analysis for Branch and Bound Integer Programming*, Operation Research, Vol. 33, 1008-1023 (1985).

16.- H. P. Williams: *Experiments in the formulation of integer Programming problems*, Mathematical Programming Study, Vol 2, 180-197 (1974).