UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

DIVISION DE ESTUDIOS DE POSGRADO

Facultad de Ingeniería

"ANALISIS DE DOS NUEVOS ALGORITMOS PARA EL PROBLEMA DE FLUJO A COSTO MINIMO"

TESIS

PRESENTADA A LA DIVISION DE ESTUDIOS DE

POSGRADO DE LA

FACULTAD DE INGENIERIA

DE LA

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

COMO REQUISITO PARA OBTENER

EL GRADO DE

MAESTRO EN INGENIERIA
(INVESTIGACION DE OPERACIONES)

POR

VICTOR RAFAEL PEREZ PEREZ

CIUDAD UNIVERSITARIA

FEBRERO DE 1995

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

AGRADECIMIENTOS

Deseo dar las gracias a todas las personas que me ayudaron en la realización de este trabajo:

A Fernando y Ofelia, por la gran suerte de tenerlos como Padres, por su gran ejemplo y apoyo incondicional en todo momento.

A mi familia: Yolanda, Jesús, Ulises e Iván; Cristina; Humberto y Yolanda; Mónica, Sergio y un nuevo sobrinito; Pilar y Rebeca.

Al Dr. Miguel Angel Gutierrez Andrade, director de esta tesis.

A los demás miembros del jurado, Dr. Sergio Fuentes Maya, M. en I. Idalia Flores de la Mota, M. en I. Carmen Hernández Ayuso, M. en I. Ricardo Aceves García, a todos ellos gracias por sus valiosas enseñanzas.

A mis compañeros y amigos, por los gratos momentos.

Y porque buscamos coincidir, a Maru, una persona muy especial.

Finalmente debo agradecer a todas las intituciones que me brindaron apoyo económico en la realización de mis estudios y tesis de Maestría:

DGAPA, CONACYT, FUNDACION-UNAM, C.F.E.

VICTOR RAFAEL PEREZ PEREZ.

FEBRERO DE 1995.

RESUMEN

En este trabajo se analizan los diferentes enfoques existentes para resolver un problema de flujo a costo mínimo en una red con capacidades, presentando la teoría básica y enfatizando dos clases de métodos recientemente propuestos.

El primero, llamado de relajación, es un algoritmo primal-dual que mantiene factibilidad primal en las restricciones de capacidad, mientras trata de satisfacer las restricciones de conservación de flujo en cada nodo utilizando tres procedimientos básicos y flexibles que son: ajuste de flujo, ajuste en las variables duales y mejoramiento de la función objetivo dual mediante ascenso. La flexibilidad de estos tres procedimientos y el incorporamiento de características clásicas de los métodos de relajación hacen que esta clase de algoritmos se vuelvan muy eficientes.

El segundo enfoque esta basado en la interpretación del problema de asignación como una subasta real, en la que sin manipular objetos globales, como son trayectorias, árboles o cortaduras, y utilizando la noción de ϵ-Holguras Complementarias, el problema puede ser resuelto eficientemente.

Correspondiendo a cada uno de estos dos enfoques, se presentan respectiva y detalladamente los algoritmos de relajación y de ϵ-relajación, para resolver problemas de flujo a costo mínimo. Se debe mencionar que el algoritmo de ϵ-relajación es interpretado como una generalización del algoritmo de subasta para el problema de asignación. Se presenta, por último, la comparación entre tres implementaciones muy eficientes, representativas de cada uno de los métodos y que deben ayudar en la discusión de los aspectos del comportamiento práctico de estos.

INDICE

1 CONCEPTOS BASICOS 7

1.1 ALGUNAS DEFINICIONES BASICAS 7

1.2 FLUJO Y DIVERGENCIA 9

1.3 EL PROBLEMA DE FLUJO A COSTO MINIMO. 9

1.4 EL PROBLEMA DE ASIGNACION 11

1.5 EL PROBLEMA DE FLUJO MAXIMO 12

1.6 EL PROBLEMA DE TRANSPORTE 14

1.7 EQUIVALENCIAS 14

1.8 ALGORITMOS BUENOS, MALOS Y POLINOMIALES 16

2 DUALIDAD 18

2.1 EL DUAL DEL PROBLEMA DE ASIGNACION 18

2.2 TRES IDEAS ALGORITMICAS BASICAS PARA RESOLVER UN PROBLEMA DE FCM. 20

2.2.1 MEJORAMIENTO DEL COSTO PRIMAL 21

2.2.2 MEJORAMIENTO DEL COSTO DUAL 21

2.2.3 SUBASTA 23

2.2.4 EL ALGORITMO DE SUBASTA INEXPERTO 23

2.2.5 ε–HOLGURAS COMPLEMENTARIAS 25

3 METODOS DE ASCENSO DUAL 30

3.1 DUALIDAD DEL PROBLEMA DE FCM 30

3.2 VISTAZO DE LOS ALGORITMOS DE ASCENSO DUAL 35

3.3 METODO PRIMAL-DUAL 36

3.4 EL METODO DE RELAJACION 42

3.4.1 BUSQUEDA LINEAL E ITERACIONES DE ASCENSO COORDINADO . 49

4 ALGORITMOS TIPO SUBASTA 54

4.1 EL ALGORITMO SUBASTA PARA EL PROBLEMA DE ASIGNACION 54

4.2 EL ALGORITMO SUBASTA PRINCIPAL 55

4.3 INTERPRETACION DEL ALGORITMO DE SUBASTA COMO UN DESCENSO COORDINADO APROXIMADO 60

4.4 ALGORITMO GENERICO DE SUBASTA PARA EL PROBLEMA DE FLUJO A COSTO MINIMO 62

4.4.1 ALGUNAS OPERACIONES ALGORITMICAS BASICAS 63

4.4.2 EL ALGORITMO GENERICO 66

4.4.3 EL METODO DE ϵ-RELAJACION 69

4.4.4 COMPLEJIDAD DEL ALGORITMO DE ϵ-RELAJACION 73

5 RESULTADOS COMPUTACIONALES 76

5.1 ESTRUCTURA DE DATOS DE LA IMPLEMENTACION 76

5.2 RESULTADOS COMPUTACIONALES 79

6 COMENTARIOS FINALES 83

7 BIBLIOGRAFIA 85

INTRODUCCION

El problema de flujo a costo mínimo surge de diversas situaciones, por ejemplo, piense que necesitamos enviar unidades de un bien que se localiza en diferentes ciudades del país a ciudades en el extranjero que lo demandan. Se incurre en un costo por cada unidad del bien que se transporte en rutas de exportación establecidas y que tienen cierta capacidad. ¿ Cómo debemos enviar el bien para incurrir en el mínimo costo posible ?

Las redes son una manera natural para representar este problema, porque los nodos pueden representar ciudades, los arcos las rutas de exportación y el flujo, el número de unidades del bien. En general el problema de flujo a costo mínimo (FCM) es el siguiente: Si se incurre en un costo por unidad de flujo en una red con capacidades en los arcos y necesitamos enviar unidades de un bien que se localiza en uno a más nodos de la red a otros nodos, ¿Cómo debemos enviar el material para incurrir en el mínimo costo posible?. Este problema posee una gran cantidad de aplicaciones en diversos campos de estudio, por ejemplo, en los sistemas de transportación, los nodos pueden representar intersecciones o aeropuertos. Los arcos, carreteras, líneas de ferrocarril, rutas aéreas y el flujo, pasajeros, vehículos u operadores. En los sistemas hidráulicos, los nodos pueden representar estaciones de bombeo, lagos, reservas. A las tuberías, las podemos interpretar como arcos y el flujo podría ser agua, petróleo o gas.

El problema de FCM es un problema de optimización lineal y forma parte de la teoría de redes, y junto con diversas variantes del problema tales como flujo máximo, asignación, transporte y rutas más cortas, son los que más frecuentemente se presentan en la práctica. A pesar de que problemas de este tipo de gran dimensión se pueden resolver actualmente sin grandes dificultades, el análisis teórico de la estructura del problema ha proporcionado ideas que han sido contribuciones muy importantes tanto en el desarrollo de la Programación Matemática como en la Optimizacion Combinatoria.

Hasta fines de los 70's se habían desarrollado dos clases de algoritmos para obtener la solución de un problema de optimización lineal en redes, por una parte, el método simplex y sus variantes, siendo Dantzig en 1951 el primero que especializó el método simplex para resolver el problema de transporte, posteriormente en 1963 lo extendió al problema de flujo a costo mínimo. Los algoritmos

de este tipo, son conocidos como métodos primales, porque mejoran el costo primal a través de la modificación del flujo en ciclos simples.

La otra clase es el método primal-dual y una variación muy cercana, el método out-of-kilter (o de bondad de ajuste). Estos métodos mejoran iterativamente el costo dual cambiando el precio de un subconjunto de nodos en igual cantidad. El enfoque de mejoras en el costo dual fué iniciado por Kuhn, en 1951 con su famoso método húngaro para resolver el problema de asignación, posteriormente, en 1956, Ford y Fulkerson lo generalizaron para resolver el problema de flujo a costo mínimo.

Una fuente de controversia se encuentra en cual de estos enfoques es el mejor, pero se ha demostrado [*Zad*79] que a pesar de parecer fundamentalmente diferentes (con el método simplex se busca mejorar el costo primal, mientras que en el método primal-dual su busca mejorar el costo dual) estos métodos están sorpresivamente relacionados, aunque parece ser que el método simplex es más eficiente en la mayor parte de los problemas, si no se toma en cuenta el análisis de sensibilidad.

Sin embargo, desarrollos algorítmicos en los 80's han cambiado esta situación, nuevos enfoques han venido a inyectar mejoras, tanto prácticas como en el análisis del comportamiento del peor caso. Uno de estos enfoques es el método de relajación propuesto por Bertsekas [*Ber*81], que es un método de ascenso dual y que surge de los métodos de ascenso coordinado de la optimización no lineal no restringida. Este método mantiene factibilidad primal con respecto a las restricciones de capacidad, mientras trata de satisfacer las restricciones de conservación de flujo en cada nodo utilizando tres pasos básicos y flexibles que son: ajuste de flujo, ajuste en las variables duales y mejoramiento de una función dual mediante ascenso. Algunas implementaciones de estos pasos hacen eficientes esta clase de algoritmos, lo que los hace comparables a los más rápidos códigos simplex primales, sobre todo en implementaciones en paralelo, donde han demostrado ser los mejores [*BeT*89]. Además, estos métodos puden ser utilizados para encontrar soluciones iniciales bastante buenas para otros métodos.

La otra idea se presenta en los algoritmos tipo subasta, que son métodos de ascenso coordinado dual y cuya interpretación en el problema de asignación da origen a su nombre, en este contexto se

consideran a un conjunto de nodos de una gráfica bipartita como personas realizando ofertas por objetos que son el otro conjunto de nodos. Las variables duales correspondientes a los nodos objetos, se consideran los precios de los objetos en la subasta. Este método mantiene factibilidad primal e itera con la función dual tratando de mejorarla, al mover una sola variable dual.

Desafortunadamente esto no siempre funciona bien, debido a que la función dual no es diferenciable, pudiendo suceder que al cambiar una sola variable no se pueda mejorar la solución, dando lugar a que se itere indefinidamente sin mejorar el valor de la función objetivo dual. Este fenómeno correspondería a que dos personas ofrezcan la misma cantidad por el mismo objeto. En una subasta, tal conflicto se resuelve cuando una persona hace una oferta ligeramente mejor, con lo que se incrementa el precio del objeto, hasta que uno de los oferentes se da por vencido y el conflicto se acaba. En el contexto matemático, esta idea se transforma en una relajación de las condiciones de Holguras Complementarias (H.C.) de la Teoría de la Programación Lineal, lo que hace que el problema satisfaga estas nuevas condiciones, llamadas ε-Holguras Complementarias, pero normalmente no las condiciones regulares de H.C. Al finalizar el algoritmo, bajo ciertos supuestos iniciales, por ejemplo que las capacidades de los arcos y el flujo sean enteros, las soluciones a los problemas primal y dual satisfacen H.C., por lo que son soluciones óptimas a sus respectivos problemas. Los algoritmos de subasta basados en la idea de $\varepsilon-$Holguras Complementarias e ideas de escalamiento tienen un comportamiento muy eficiente para los problemas de asignación y de flujo máximo.

Uno de los objetivos de este trabajo es proporcionar una revisión de los algoritmos para resolver problemas lineles de flujo en redes. Como extensos análisis se han realizado sobre los métodos de mejoramiento del costo primal, haremos énfasis en los dos nuevos algoritmos que han demostrado ser eficientes o que tienen ideas metodológicas importantes: el algoritmo de relajamiento y el algoritmo de subasta. En este trabajo, dos ideas fundamentales de la Programación Matemática se enfatizan dualidad y mejoramiento iterativo del costo. Un segundo objetivo es utilizar varios códigos muy eficientes de los algoritmos presentados. Estos códigos ilustran técnicas de implementación del estado del arte, usadas en la optimización de redes y deberán ayudar en la discusión de los aspectos del comportamiento práctico de los diferentes métodos.

Por lo tanto, este trabajo se desarrolla de la siguiente manera

En el primer capítulo veremos la terminología y conceptos básicos que se utilizan en el trabajo, así como las definiciones y equivalencias entre los diferentes problemas de optimización lineal en redes.

En el segundo capítulo estudiaremos la teoría de dualidad que juega un papel importante en las tres ideas algoritmicas básicas para resolver un problema de flujo a costo mínimo, haciendo énfasis en el mejoramiento dual y en los algortimos subasta.

En el tercer capítulo se tratan dos métodos de ascenso dual: el método primal-dual y el método de relajamiento, que tienen en común que ambos métodos tratan de encontrar direcciones que mejoran el costo dual. El método primal-dual tiene como ventaja que puede ser implementado usando una subrutina de rutas más cortas, sin embargo su desventaja con respecto al método de relajación es que calcula mucho más lento las direcciones de ascenso.

En el cuarto capítulo se tratan los algoritmos tipo subasta: comenzamos con el algoritmo para el problema de asignación que dió origen a este nuevo enfoque, posteriomente se generalizan estos conceptos para el problema de flujo a costo mínimo dando origen al algoritmo genérico. Como resultado de la aplicación de este último algoritmo a un caso particular, donde los incrementos en el flujo y los aumentos de precios involucran solamente un nodo, se presenta el método de ϵ-relajación.

En el quinto capítulo se comenta la estructura de datos utilizada, que es muy importante para lograr la eficiencia de la implementación. También se proporcionan los tiempos obtenidos en la solución de diversos problemas de FCM, con una implementación, programada en FORTRAN, de los algoritmos de relajación y subasta. Por último, se presentan las conclusiones de este trabajo y la bibliografía.

1 CONCEPTOS BASICOS

En este capítulo se ven los conceptos asociados de la Teoría de Gráficas que se utilizan en el presente trabajo. También se definen los problemas básicos de redes y se proporcionan ejemplos de los cuales surgen, así como las equivalencias entre los diferentes tipos de problemas. Por último se menciona una manera común para comparar diferentes tipos de algoritmos: el análisis de la complejidad del peor caso. Comenzaremos con la definición de varios conceptos.

1.1 ALGUNAS DEFINICIONES BASICAS

Una *gráfica dirigida* $G = (N, A)$ está formada por dos conjuntos , uno de ellos N cuyos elementos son llamados *nodos* y otro conjunto A formado por pares de nodos distintos de N llamados *arcos.* El número de nodos y de arcos de G se denotan N y A respectivamente, suponemos que $1 \leq N < \infty$ y $0 \leq A < \infty$. Un arco (i, j) es un par ordenado y es distinto del par (j, i). Si (i, j) es un arco, se dice que (i, j) es un *arco exterior* de i y un *arco interior* de j , también que j es *sucesor* de i y que i es *predecesor* de j. El arco (i, j) es *incidente* a i y a j, i es el *nodo inicial* y j es el *nodo final* del arco. El *grado de un nodo* i es el número de arcos que son incidentes a él.

Una gráfica es *bipartita* si sus nodos pueden ser particionados en 2 conjuntos S y T tales que todo arco tenga su nodo inicial en S y su nodo final en T o viceversa .

En este trabajo, no se permiten más de un arco en la misma dirección entre un par de nodos, por lo que nos referiremos sin ambigüedad al arco con nodo inicial i y nodo final j como el arco (i, j), esta restricción se hace únicamente por conveniencia en la notación.

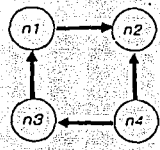
Una *cadena* P en una gráfica dirigida es una secuencia de nodos $n_1, n_2, \ldots n_k$ con $k \geq 2$ y una secuencia correspondiente de $k - 1$ arcos tales que el i-ésimo arco en la secuencia es (n_i, n_{i+1}) (en cuyo caso es llamado *arco positivo o hacia adelante* de la cadena) o (n_{i+1}, n_i) (en cuyo caso es llamado *arco negativo o hacia atrás* de la cadena).

Una cadena es una *trayectoria positiva* (*negativa*) si sus arcos son positivos (negativos, respecti-

vamente). Se denota con P^+ y P^- los conjuntos de arcos positivos y negativos, respectivamente. Los nodos n_1 y n_k son el nodo *inicial* (u *origen*) y el nodo *final* (o *destino*) de P , respectivamente.

Un *ciclo* es una cadena para la cual el nodo inicial y el final coinciden. Una cadena es *simple* si no contiene arcos y nodos repetidos, excepto cuando el nodo inicial y final coinciden (en cuyo caso la cadena es un *ciclo simple*). Estas definiciones se ilustran en la figura 1.1

a) $P = (n_1, n_2, n_3, n_4)$ es una trayectoria positiva simple.
$P = (n_1, n_2, n_3, n_2, n_3, n_4)$ es una cadena, que no es simple, ni es trayectoria positiva ni negativa.



b) Un ciclo simple $C = (n_1, n_2, n_3, n_4, n_1)$ que no es positivo ni negativo
$C^+ = \{(n_1, n_2), (n_3, n_4), (n_4, n_1)\}$ y $C^- = \{(n_3, n_2)\}$

c) $P = (n_1, n_2, n_3, n_4)$ con la secuencia de arcos $\{(n_1, n_2), (n_3, n_2), (n_4, n_3)\}$

Figura 1.1 Ejemplos de diferentes tipos de cadenas

Note que la secuencia de nodos $n_1, n_2, \ldots, n_k$ no es suficiente para especificar una cadena, por lo que la secuencia de arcos es también importante para que quede bien definida. Las dificultades se presentan cuando para dos nodos sucesivos n_i y n_{i+1} de la cadena tanto (n_i, n_{i+1}) como (n_{i+1}, n_i) son arcos, por lo que se presenta una ambigüedad sobre cual de los dos es el arco correspondiente a la cadena (ver el diagrama c). Sin embargo, cuando la trayectoria se sabe que es positiva o negativa, está definida únicamente por la secuencia de sus nodos.

Una gráfica que no contenga ciclos simples es *acíclica*. Una gráfica es *conexa o conectada* si para cada par de nodos i y j , existe una cadena iniciando en i y terminando en j. Se llama *fuertemente conexa* si para cada par de nodos i y j hay una trayectoria positiva del nodo i al nodo j.

$G' = (N', A')$ es una *subgráfica* de $G = (N, A)$ si G' es una gráfica, $N' \subset N$ y $A' \subset A$. Un *árbol* es una gráfica conexa y acíclica. Un *árbol expandido* de una gráfica G es una subgráfica de G que es árbol e incluye todos los nodos de G.

1.2 FLUJO Y DIVERGENCIA

Un *vector de flujo* X en una gráfica $G = (N, A)$ es un conjunto de escalares $\{x_{ij} \mid (i,j) \in A\}$

Nos referiremos a x_{ij} como el *flujo del arco* (i,j) y no existen restricciones (tales como la no negatividad) en su valor. El *vector de divergencia* Y asociado con un vector de flujo X es el vector N-dimensional con coordenadas

$$y_i = \sum_{\{j \mid (i,j) \in A\}} x_{ij} - \sum_{\{j \mid (j,i) \in A\}} x_{ji} \qquad \forall \ i \in N \tag{1.1}$$

Así, y_i es el flujo total que sale del nodo i menos el flujo total que llega a i, al cual se le llama *divergencia de* i.

Un nodo i es una *fuente* (*sumidero*) para el vector de flujo X, si $y_i > 0$ ($y_i < 0$, respectivamente). Si $y_i = 0$ para todo $i \in N$, entonces X es una *circulación*. Estas definiciones se ilustran en la figura 1.2.

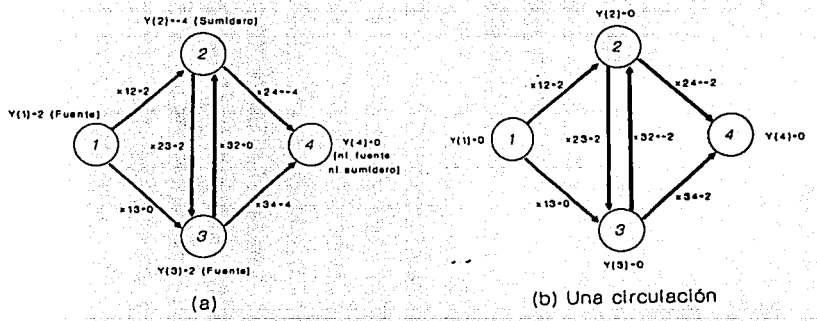


Figura 1.2 Ejemplos de diferentes tipos de flujos

Note que al sumar la ecuación (1.1) para todo $i \in N$, obtenemos $\sum_{i \in N} y_i = 0$

1.3 EL PROBLEMA DE FLUJO A COSTO MINIMO.

Este problema consiste en encontrar un conjunto de flujos en los arcos que minimiza una función de costo lineal sujeta a restricciones producidas por un vector de divergencia dado y que está en un cierto intervalo, esto es

$$\min_{s.a.} \sum_{(i,j)\in A} a_{ij}x_{ij}$$

$$\sum_{\{j|(i,j)\in A\}} x_{ij} - \sum_{\{j|(j,i)\in A\}} x_{ji} = s_i \qquad \forall \quad i \in N \tag{1.2}$$

$$b_{ij} \leq x_{ij} \leq c_{ij} \qquad \forall\, (i,j) \in A \tag{1.3}$$

donde a_{ij}, v_{ij}, c_{ij} y s_i son escalares dados.

a_{ij} es el coeficiente de costo (o simplemente costo) de (i,j)

b_{ij} y c_{ij} son las cotas inferior y superior de flujo de (i,j)

$[b_{ij}, c_{ij}]$ es el rango de flujo factible de (i,j)

s_i es la oferta del nodo i.

Las restricciones (1.2) y (1.3) son también llamadas *restricciones de conservación de flujo* y *de capacidad*, respectivamente. Un vector de flujo que satisface las restricciones de capacidad, es de *capacidad factible*. Si existe al menos un vector de flujo factible, el problema de flujo a costo mínimo (FCM) es *factible*, de otra manera es *no factible*. Note que para la factibilidad se debe tener

$$\sum_{i\in N} s_i = 0 \tag{1.4}$$

dado que por la ecuación (1.2), para cualquier vector de flujo, la suma de todas las divergencias correspondientes de los nodos deben ser cero.

Como una aplicación típica del problema de FCM, piense que los nodos son lugares tales como ciudades, bodegas o fábricas, donde un cierto bien es producido o consumido. Piense en los arcos como rutas de transportación entre los lugares, cada uno con costo de transportación a_{ij} por unidad transportada. El problema es transladar el bien de los centros de producción a los centros de consumo a un costo mínimo, respetando las restricciones de capacidad de las rutas de transportación.

El problema de flujo a costo mínimo es un caso especial de un problema de programación lineal, pero tiene una estructura mucho más favorable, con ciertas propiedades especiales que afectan fuertemente el comportamiento de algoritmos. Por ejemplo, el problema de FCM con datos enteros, puede resolverse usando exclusivamente calculos enteros. Sin embargo, algunos algoritmos que son eficientes para resolver este problema (relajación o subasta), son menos eficientes o inaplicables para problemas lineales generales.

Tres importantes casos del problema de FCM son el problema de asignación, el problema de flujo máximo y el problema de transporte. En seguida se describen

1.4 EL PROBLEMA DE ASIGNACION

Suponga que existen n personas y n objetos, existe además un beneficio o valor a_{ij} para acoplar la persona i con el objeto j, se desea asignar personas a objetos, de tal manera que un objeto sólo pueda asignarse a una persona y viceversa, de tal manera que se maximice el beneficio total. Existe la restricción de que la persona i puede ser asignada al objeto j si y sólo si (i,j) pertenece a un conjunto de pares dados A. Matemáticamente se desea encontrar un conjunto de pares personas-objetos $(1,j_1),..,(n,j_n)$ de A tal que los objetos j_1 , .. , j_n son todos distintos y el beneficio total $\sum_{i=1} a_{ij_i}$ sea maximizado.

El problema de asignación es importante en muchos contextos prácticos, por ejemplo en los problemas de colocación de recursos, como la asignación de empleados a trabajos, máquinas a tareas, etc. También hay situaciones donde el problema de asignación aparece como un subproblema en varios algoritmos para resolver problemas más complejos (El problema del cartero chino, el problema del agente viajero).

Si asociamos cualquier asignación con el conjunto de variables $\{x_{ij} \mid (i,j) \in A\}$ donde $x_{ij} = 1$ si la persona i es asignada al objeto j y $x_{ij} = 0$ en otro caso, podemos formular el problema de asignación como un problema lineal de la siguiente manera

$$\max_{s.a.} \sum_{(i,j)\in A} a_{ij}x_{ij}$$

$$\sum_{\{j|(i,j)\in A\}} x_{ij} = 1 \qquad \forall\, i = 1,..,n$$

$$\sum_{\{i|(i,j)\in A\}} x_{ij} = 1 \qquad \forall\, j = 1,..,n$$

$$0 \le x_{ij} \le 1 \quad \forall\, (i,j) \in A$$

Se debería agregar la restrición de que x_{ij} sea 0 o 1, sin embargo, este problema tiene la siguiente propiedad: si tiene una solución factible, entonces tiene una solución óptima donde todo x_{ij} es 0 o 1. Esta propiedad se debe a que la matriz de restricciones es unimodular (es decir, cualquier

submatriz cuadrada tiene determinante con valor 0,±1). Otra propiedad importante del problema de asignación es que puede representarse como una gráfica mostrada en la figura 1.3

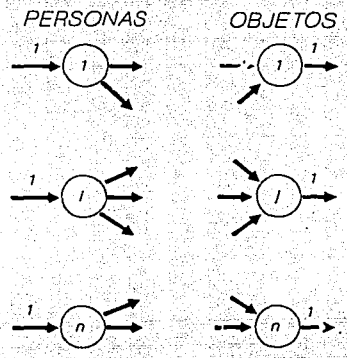


Figura 1.3 Representación del problema de asignación por medio de una gráfica

En la gráfica de la figura 1.3 existen 2n nodos divididos en dos grupos; n corresponden a personas y n corresponden a objetos. Para cada $(i,j) \in A$ existe un arco conectando a la persona i con el objeto j. En la terminología de problemas de redes, la variable x_{ij} es el flujo del arco (i,j). La restricción $\sum_{\{j|(i,j)\in A} x_{ij} = 1$ indica que el flujo total que sale del nodo i ser igual a 1, que puede ser visto como la oferta del nodo. Similarmente $\sum_{\{i|(i,j)\in A} x_{ij} = 1$ indica que el flujo total que llega al nodo j debe ser igual a 1, que puede ser vista como la demanda del nodo.

1.5 EL PROBLEMA DE FLUJO MAXIMO

En el problema de flujo máximo hay 2 nodos especiales: el fuente u origen (s) y el sumidero o destino (t). El objetivo es enviar el máximo flujo posible de s a t respetando las restricciones de capacidad. Formalmente, queremos hacer que la divergencia de s se maximice (o equivalentemente, minimizar la divergencia de t).

Para formular este problema como un caso especial del problema de flujo a costo mínimo, se debe asignar un costo cero a todos los arcos e introducir un arco (t,s) con costo -1 y con cotas de flujo superior e inferior convenientemente grande y pequeña respectivamente, como se puede ver en la figura 1.4.

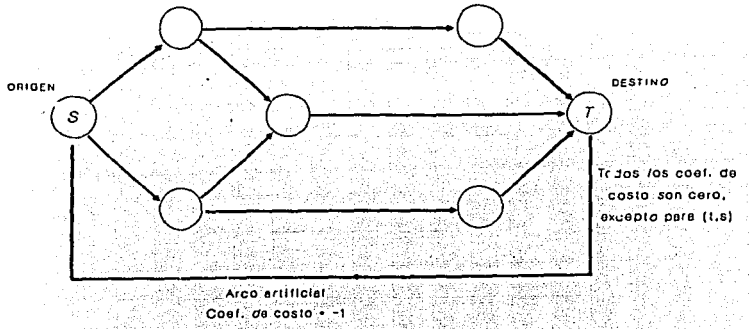


Figura 1.4 La representación como problema de FCM de un problema de flujo máximo. En el óptimo, el flujo x_{ts} es igual al flujo máximo que puede enviarse de s a t a través de la subgráfica obtenida al suprimir el arco (t,s).

Matemáticamente el problema es:

$$\underset{s.a.}{Max}\ x_{ts}$$

$$\sum_{\{j|(i,j)\in A\}} x_{ij} - \sum_{\{j|(j,i)\in A\}} x_{ji} = 0 \qquad \forall\, i \in N \quad \text{con } i \neq s \text{ e } i \neq t$$

$$\sum_{\{j|(s,j)\in A\}} x_{sj} = \sum_{\{i|(i,t)\in A\}} x_{it} = x_{ts}$$

$$b_{ij} \leq x_{ij} \leq c_{ij} \qquad \forall\,(i,j) \in A \ \text{ con } (i,j) \neq (t,s)$$

$$\sum_{\{i|(i,t)\in A\}} b_{it} \leq x_{ts} \leq \sum_{\{i|(i,t)\in A\}} c_{it}$$

Las cotas inferiores y superiores en x_{ts} son introducidas para lograr el formato de flujo a costo mínimo, sin embargo éstas son redundantes, puesto que las cotas superior e inferior de los flujos en los arcos de A, las incluyen. Del mismo modo, viendo el problema como uno de maximización su interpretación intuitiva es consistente. Alternativamente, el problema se puede escribir como minimizar $-x_{ts}$ sujeto a las mismas restricciones.

En una formulación alternativa la cota de flujo en x_{ts} puede descartarse, puesto que es implicada por otras las demás cotas.

1.6 EL PROBLEMA DE TRANSPORTE

Es el mismo problema de asignación excepto que las ofertas no deben ser 1 o -1 y en vez de maximizar se tiene que minimizar la función.

Por lo que este problema tiene la forma:

$$\begin{aligned} \underset{s.a.}{Min} \quad & \sum_{(i,j)\in A} a_{ij}x_{ij} && (1.5)\\ & \sum_{\{j|(i,j)\in A\}} x_{ij} = \alpha_i && \forall\, i = 1,..,m \\ & \sum_{\{i|(i,j)\in A\}} x_{ij} = \beta_j && \forall\, j = 1,..,n \\ & 0 \le x_{ij} \le min\{\alpha_i, \beta_j\} && \forall\,(i,j) \in A \end{aligned}$$

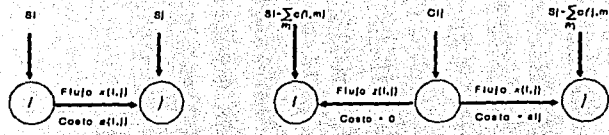
donde α_i y β_j son escalares positivos, y para que el problema sea factible se debe satisfacer

$$\sum_{i=1}^{m} \alpha_i = \sum_{j=1}^{n} \beta_j$$

La restricción de la cota superior $x_{ij} \le min\{\alpha_i, \beta_j\}$ puede ser descartada en una formulación alternativa, dada que es implicada por la conservación de flujo y las restricciones de no negatividad.

1.7 EQUIVALENCIAS

El problema de FCM puede ser transformado en un problema de transporte de la forma (1.5). La idea es introducir un nuevo nodo para cada arco y una variable de holgura z_{ij} para todo flujo en un arco (i,j), es decir z_{ij} debe cumplir $x_{ij} + z_{ij} = c_{ij}$ esto no sólo elimina la restricción de la cota superior en el flujo de los arcos, también crea una estructura de gráfica bipartita, porque para cada arco (i,j) tenemos un nodo extra con oferta c_{ij} y dos arcos exteriores, correspondientes a los flujos x_{ij} y z_{ij}. Vea la figura 1.5



Rango de factibilidad $[0, c_{ij}]$
Arco original

Rango de factibilidad $[0, \infty]$
Después de la transformación

Figura 1.5 Transformación en los nodos de un problema de FCM a un problema de transporte

En particular, se toma como orígenes del problema de transporte a los arcos de la red original, y como destinos los nodos de la red original. Cada origen de este nuevo problema tiene dos arcos que salen de él con coeficientes de costo 0 y a_{ij}. La oferta de cada origen es la longitud del rango de flujo factible del arco correspondiente en la red original. La demanda de cada destino es la suma de las longitudes del rango de flujo factible de los arcos que tienen su extremo inicial (los que salen) el nodo correspondiente en la red original menos la oferta de ese nodo. Un flujo x_{ij} en el problema de FCM corresponde a flujos iguales a x_{ij} y $c_{ij} - b_{ij} - x_{ij}$ del problema de transporte en arcos $((i,j),j)$ y $((i,j),i)$ respectivamente, todo lo anterior se puede observar en la figura 1.6

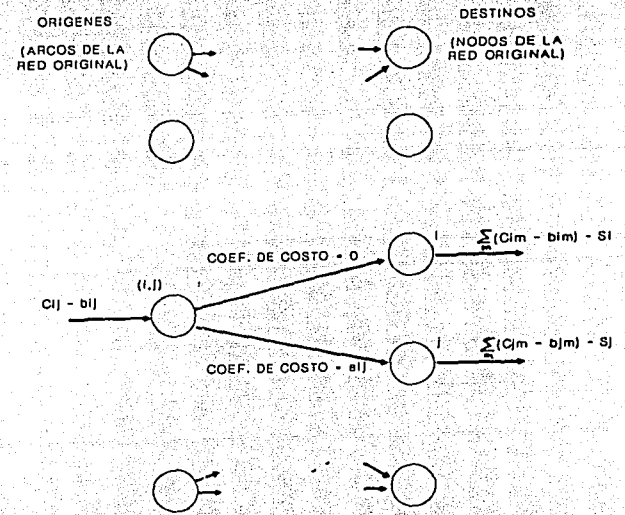


Figura 1.6 Transformación de un problema de FCM a un problema de Transporte.

Por otra parte, el problema de transporte (ecuación 1.5) puede ser convertido en un problema de asignación creando α_i orígenes con oferta unitaria (β_j destinos con demanda unitaria) para cada origen i del problema de transporte (destino j, respectivamente). Por esta razón, cualquier algoritmo que resuelva el problema de asignación puede ser extendido a un algoritmo para el problema de flujo a costo mínimo. Esto motiva un camino útil para desarrollar y analizar nuevas ideas algoritmicas, que consiste en aplicarlas al problema de asignación y generalizarlas al problema de flujo a costo mínimo usando la construcción dada para pasar de un problema al otro.

1.8 ALGORITMOS BUENOS, MALOS Y POLINOMIALES

Al discutir diferentes tipos de métodos una pregunta natural es la siguiente: ¿Hay algún método que sea el mejor y si es así, bajo que criterios ordenarlos?

Normalmente estamos interesados en algunas de las siguientes características

1) El tiempo de solución.

2) Flexibilidad para poder utilizar buenas soluciones iniciales.

3) Poder realizar análisis de sensibilidad.

4) Flexibilidad para poder lograr su implementación en paralelo.

5) Requerimientos de memoria.

Por la variación de las caracteristicas, no es sorprendente que no haya un algoritmo que domine a los demás en todas las situaciones.

Existen diferentes enfoques para el análisis de algoritmos, el más común es el análisis de la complejidad del peor caso. En este enfoque se trata de acotar el número de operaciones númericas elementales necesarias que realiza un algoritmo dado, generalmente con alguna medida del tamaño del problema, es decir con alguna expresión de la forma

$$K\, f(N, A, C, U, S)$$

donde

N es el número de nodos

A es el número de arcos

C es el rango máximo del costo del arco $\max_{(i,j)\in A} |a_{ij}|$

U es el rango máximo del flujo del arco $\max_{(i,j)\in A} (c_{ij} - b_{ij})$

S es el rango de la oferta $\max_{i\in N} |S_i|$

f es una función conocida

K es una constante (usualmente desconocida)

Si una cota de esta forma puede darse, se dice que el tiempo de ejecución o de corrida es $O(f(N, A, C, U, S))$. Si $f(N, A, C, U, S)$ puede ser escrito como una función polinomial del número

de bits necesarios para expresar los datos del problema, el algoritmo es *polinomial*. Las siguientes funciones son ejemplos de cotas de complejidad polinomial $O(N^{\alpha}A^{\beta})$ y $O(N^{\alpha}A^{\beta}\log C)$ con $\alpha, \beta \geq 0$. La cota $O(N^{\alpha}A^{\beta})$ es llamada algunas veces *fuertemente polinomial* porque involucra solamente parámetros del tamaño de la gráfica. Una cota de la forma $O(N^{\alpha}A^{\beta}C)$ no es polinomial porque C no es una expresión polinomial de $\log C$, el número de bits necesarios para expresar un solo número del valor de C. Cotas como $O(N^{\alpha}A^{\beta}C)$ que son polinomiales en los datos del problema, en vez del número de bits necesarios para expresar los datos son llamados *pseudopolinomiales*.

Un supuesto común en la teoría de la Ciencia de la Computación es que un algoritmo polinomial es mejor que uno pseudopolinomial y estos son mejores que los exponenciales (por ejemplo, aquellos con una cota de la forma $K2^{g(N,A)}$, donde g es polinomial en N y A). Aún más, dos algoritmos polinomiales pueden compararse en terminos del grado de la cota polinomial, es decir un algoritmo $O(N^2)$ es mejor que un algoritmo $O(N^3)$. Sin embargo, existen contraejemplos que hacen que esto no sea siempre cierto, por lo que la estimación de la complejidad del peor caso puede fallar para predecir el comportamiento práctico de unos algoritmos. El ejemplo más famoso es el método simplex que es un algoritmo exponencial para resolver problemas lineales generales y sin embargo se utiliza ampliamente porque tiene muy buen comportamiento en la práctica.

Las dos principales razones en la utilización de la complejidad del peor caso, son que la cota superior que se da puede ser muy pesimista porque corresponde a posibles problemas poco probables, para tales situaciones, la estimación de la complejidad promedio podría ser mas apropiada, sin embargo, obtenerla es usualmente difícil y los supuestos estadísticos necesarios pueden ser no apropiados para muchos tipos de problemas practicos. Segundo, la estimación de la complejidad del peor de los casos involucra la constante K (usualmente desconocida) que puede dominar la estimación para todos los tamaños de los problemas, excepto para problemas de gran tamaño que no pueden ser reales. Por lo que en casos extremos, una comparación entre dos algoritmos que esta basada en los terminos dependientes del tamaño, del tiempo de ejecución estimado y que no toma en cuenta las correspondientes constantes puede dar una idea errónea del comportamiento de los algoritmos.

2 DUALIDAD

En este capítulo se introduce intuitivamente el concepto de dualidad, que es la base para la solución de los diferentes problemas de redes, pues es una herramienta que puede utilizarse para detectar optimalidad en las soluciones. Se comentan las tres ideas principales para resolver un problema de flujo a costo mínimo y que son i) el mejoramiento del costo primal, 2) el mejoramiento del costo dual y iii) los algoritmos tipo subasta, con énfasis en la más reciente de las tres, los algoritmos tipo subasta para el problema de asignación, que surgen en los años 80's.

Varios autores [*Gri*86] y [*AMO*93] indican que la implementación de este algoritmo para resolver el problema de FCM, conocida como RELAX, junto con la implementación del algoritmo simplex especializado en redes son dos de los algoritmos más veloces para resolver en la práctica el problema de flujo a costo mínimo. En la sección 2.1.3, se puede apreciar que en el problema de asignación, si a los nodos de un conjunto de la bipartición los interpretamos como personas que compiten por objetos, siendo estos los nodos restantes de la gráfica, la asociación de una subasta verdadera surge de manera natural. La primera aproximación para resolver el problema, el algoritmo subasta inexperto contiene una pequeña imperfección que se resuelve empleando el concepto de ϵ-Holguras Complementarias, que presentamos en la última sección del capítulo, y que son una relajación de las condiciones de Holgura Complementarias que se utilizan para checar optimalidad en la Programación Lineal.

2.1 EL DUAL DEL PROBLEMA DE ASIGNACION

Todos los problemas que se han mencionado forman parte de la Programación Lineal (P. L.), dentro de esta teoría, la dualidad trata con la relación entre el problema original de P.L. y otro problema lineal llamado dual. Una interpretación intuitiva de la dualidad, se dará por medio del problema de asignación, considerando un problema muy cercano de equilibrio económico. Para esto debemos acoplar n objetos con n personas a través de un mecanismo de mercado, viendo cada objeto como un agente económico actuando en su propio interés, suponga que el objeto j tiene un precio p_j y que la persona que recibe el objeto debe pagar el precio p_j. Entonces el valor neto del objeto j para la persona i es $a_{ij} - p_j$ y cada persona i lógicamente desea ser asignada a un objeto j_i con valor

máximo, esto es con

$$a_{ij_i} - p_{j_i} = \max_{j \in A(i)} \{a_{ij} - p_j\}$$

donde $A(i) = \{j \mid (i,j) \in A\}$ es el conjunto de objetos que pueden ser asignados a la persona i. Cuando esta condición se cumple para todas las personas i, decimos que la asignación y el conjunto de precios satisfacen *Holguras Complementarias (H.C.)*, cuyo nombre proviene de la terminología de la P.L. El sistema económico está en equilibrio, en el sentido de que ninguna persona puede tener un incentivo unilateral para buscar otro objeto, por lo que tales condiciones de equilibrio son naturalmente de gran interés para los economistas, pero además existe una relación fundamental con el problema de asignación. Esto se aprecia en la siguiente proposición

Proposición 2.1 Si una asignación factible y un conjunto de precios satisfacen las condiciones de H.C. para todas las personas i, entonces la asignación es óptima y los precios son una solución óptima del siguiente problema

$$\min_{\substack{p_j \\ j=1,\ldots,n}} \left\{ \sum_{i=1}^{n} \max_{j \in A(i)} \{a_{ij} - p_j\} + \sum_{j=1}^{n} p_j \right\}$$

llamado *problema dual*. Aún más, el beneficio de la asignación óptima y el costo óptimo del problema dual son iguales.

Demostración. El costo total de cualquier asignación factible $\{(i,k_i) \mid i = 1,..,n\}$ satisface

$$\sum_{i=1}^{n} a_{ik_i} \leq \sum_{i=1}^{n} \max_{j \in A(i)} \{a_{ij} - p_j\} + \sum_{j=1}^{n} p_j \qquad (2.1)$$

para cualquier conjunto de precios $\{p_j \mid j = 1,..,n\}$ dado que el primer término del lado derecho no es menor que

$$\sum_{i=1}^{n} (a_{ik_i} - p_{k_i}),$$

mientras que el segundo termino es igual a $\sum_{i=1}^{n} p_{k_i}$. Por otro lado, la asignación y el conjunto de precios dados, denotados por $\{(i,j_i) \mid i = 1,..,n\}$ y $\{\bar{p}_j \mid j = 1,..,n\}$, respectivamente, satisfacen las condiciones de H.C., así que

$$a_{ij_i} - \bar{p}_{j_i} = \max_{j \in A(i)} \{a_{ij} - \bar{p}_j\} \quad i = 1,..,n$$

sumando esta relación para todo i, vemos que

$$\sum_{i=1}^{n} a_{ij_i} = \sum_{i=1}^{n} \left(\max_{j \in A(i)} \{a_{ij} - \tilde{p}_j\} + \tilde{p}_{j_i} \right)$$

Así, la asignación $\{(i, j_i) \mid i = 1, .., n\}$ alcanza el máximo del lado izquierdo de la ecuación (2.1) y es óptima para el problema primal, mientras que $\{\tilde{p}_j \mid j = 1, .., n\}$ alcanza el mínimo del lado derecho de la ecuación (2.1) y es óptima para el problema dual, aún más, los dos valores óptimos son iguales. □.

2.2 TRES IDEAS ALGORITMICAS BASICAS PARA RESOLVER UN PROBLEMA DE FCM.

Existen tres ideas principales para resolver un problema de FCM

1) **Mejoramiento del costo primal.**

Se trata de mejorar iterativamente el costo primal hasta su valor óptimo construyendo una secuencia de flujos factibles.

2) **Mejoramiento del costo dual.**

Se define un problema relacionado con el problema FCM llamado *problema dual*, cuyas variables son llamadas *precios*. Se trata de mejorar iterativamente el costo dual hasta su valor óptimo construyendo una secuencia de precios. Los algoritmos que mejoran el costo dual también iteran en los flujos, que se relacionan con los precios a través de la propiedad de holguras complementarias.

3) **Subasta.**

Este es un proceso que genera una secuencia de precios de tal manera que es una reminicencia de las subastas de la vida real. Hablando estrictamente, no hay mejoramiento del costo primal o dual, aunque se puede pensar el proceso de subasta como un mejoramiento iteractivo del costo dual en un sentido aproximado. Además de los precios, los algoritmos subasta también iteran en los flujos, que se relacionan con los precios a través de la propiedad llamada ϵ-holguras complementarias, que es una forma aproximada de la propiedad de holguras complementarias mencionada.

2.2.1 MEJORAMIENTO DEL COSTO PRIMAL

Una idea algoritmica importante para el problema de FCM es iniciar con un vector de flujo factible y generar una secuencia de vectores de flujo factible, teniendo cada uno un mejor costo que el precedente. La diferencia de cualesquiera dos flujos sucesivos debe ser una circulación, porque ambos son factibles, y para muchos algoritmos, incluyendo el método simplex, esta circulación involucra únicamente un ciclo simple.

2.2.2 MEJORAMIENTO DEL COSTO DUAL

En analogía con los algoritmos de mejoramiento del costo primal, se puede inciar con un vector de precios y tratar de obtener sucesivamente nuevos vectores de precios mejorando el costo dual. La mayoría de los algoritmos de este tipo involucran cambios en los precios a través de un tipo particular de direcciones, llamadas *direcciones elementales*. Tales direcciones son de la forma $d = (d_1, .., d_n)$ donde

$$d_i = \begin{cases} 1 & \text{si } i \in S \\ 0 & \text{si } i \notin S \end{cases}$$

y S es un conjunto conectado de nodos. Los diferentes algoritmos corresponden a diferentes métodos para determinar el conjunto de nodos S. Dada una dirección elemental y su correpondiente conjunto S, los precios se modifican de tal manera que el nuevo vector de precios tiene un mejoramiento en el costo dual.

La existencia de al menos una dirección elemental para el mejoramiento de un vector de precios que no es óptimo se demostrará posteriormente. Este importante resultado, puede ser visto como la versión dual de la siguiente proposición *Si un vector de flujo no es óptimo, entonces existe al menos un ciclo simple con costo negativo*, ya que ambos son casos especiales de un teorema más general concerniente a vectores elementales de subespacios, que son centrales en la teoría de la programación monotrópica [Roc84].

La mayoría de los métodos de mejoramiento del costo dual, al cambiar el vector de precios P en la dirección de un mejoramiento del costo dual iteran también con un vector de flujo X que satisface

H.C. junto con P. Los algoritmos terminan cuando X es factible y como (X, P) satisfacen H.C., entonces son soluciones optimas primal y dual.

Se discuten dos métodos que seleccionan direcciones elementales del mejoramiento del costo dual de diferentes maneras

1. En el método primal-dual, la dirección elemental tiene la propiedad de pendiente máxima, es decir, proporciona la tasa máxima de mejoramiento del costo dual por unidad de cambio en el vector de precios.

2. En el método de relajación (o ascenso coordinado), las direcciones elementales son calculadas de tal manera que el conjunto S tenga el menor número de nodos. Tal dirección puede ser no óptima en términos de la tasa de mejoramiento del costo dual, pero generalmente puede ser calculada mucho más rápido que la dirección de pendiente máxima. A menudo la dirección elemental tiene únicamente un elemento no cero, en cuyo caso solamente el precio de un nodo coordinado es cambiado; esto motiva el nombre de ascenso coordinado. Note, sin embargo, que las direcciones de ascenso coordinado no pueden ser usadas exclusivamente para mejorar el costo dual, como se muestra en la figura 2.1

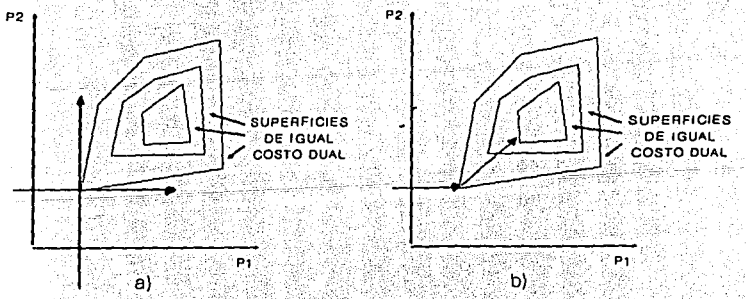


Figura 2.1 a) Aquí podemos observar la dificultad al usar iteraciones de ascenso coordinado exclusivamente. El costo dual es lineal por pedazos, así en alguna esquina (punto extremo) será imposible mejorar el costo dual cambiando cualquier precio coordinado simple (es decir uno solo).
b) El mejoramiento del costo dual es posible al cambiar varios precios coordinados en cantidades iguales, que correspondan a una dirección elemental.

Ambos métodos, el primal-dual y el de relajación terminan si los datos del problema son enteros. Aún más, simultáneamente con un vector de precios óptimo, proporcionan un vector de flujo óptimo.

2.2.3 SUBASTA

El tercer tipo de algoritmo representa un alejamiento significativo de la idea de mejorar el costo; en cualquier iteración, uno puede deteriorar tanto el costo primal como el dual, y a pesar de esto al final se encuentra una solución óptima primal. Esta idea data de 1981 en los trabajos de Bertsekas [Ber81,Ber85b,Ber88] cuyo *algoritmo tipo subasta* es un procedimiento para el problema de asignación (acoplamiento bipartito) que precede a la familia de algoritmos RELAX.

Así, en el contexto del problema de asignación surge un proceso natural para encontrar una asignación y un vector de precios en equilibrio, al considerar a los nodos de un conjunto de la bipartición como personas o agentes realizando ofertas por los objetos, que son representados por el otro conjunto. Las variables duales p_j, correspondientes a los nodos objetos, son vistas como los precios actuales de los objetos en la subasta. Hemos dicho que el problema consiste en asignar a cada persona un sólo objeto y viceversa. Al inicio se cuenta con algunas parejas asignadas, de tal manera que el objeto que les toca bajo la asignación es el de su máxima preferencia. Es claro que si todas las personas están asignadas de esta manera, la subasta termina, en otro caso, existe una persona que no esta asignada. Esta persona busca su objeto preferido y realiza una oferta, dando como resultado que el precio del objeto se encarezca y además la persona que estaba asignada a él (si había) quede sin asignar. Este proceso se repite hasta que todas las personas consigan un objeto. Este proceso se llama algoritmo de subasta inexperto porque tiene una seria imperfección, sin embargo precisamente esta imperfección motiva un algoritmo correcto aunque un poco más sofisticado. Enseguida se describe el algoritmo.

2.2.4 EL ALGORITMO DE SUBASTA INEXPERTO

El algoritmo de subasta inexperto procede en iteraciones y genera una secuencia de vectores de precios y de asignaciones parciales. Al inicio de cada iteración, las condiciones de H.C.

$$a_{ij_i} - p_{j_i} = \max_{j \in A(i)} \{a_{ij} - p_j\}$$

se satisfacen para todos los pares (i, j_i) de la asignación parcial. Si todas las personas están asignadas, el algoritmo termina. De otra forma seleccionamos alguna persona sin asignar, digamos

la persona i. Esta persona encuentra un objeto j_i, que ofrece valor máximo, esto es

$$u_i = \max_{j \in A(i)} \{a_{ij} - p_j\} \tag{2.2}$$

y entonces

PASO 1

Hacer la asignación de i al mejor objeto j_i; la persona que estaba asignada a j_i al principio de la iteración (si había) llega a ser no asignada

PASO 2

Colocar el precio de j_i al nivel en el cual el o ella es indiferente entre j_i y el segundo mejor objeto, esto es, el o ella coloca

$$p_{j_i} \text{ en } p_{j_i} + \gamma_i \tag{2.3}$$

$$\text{donde } \gamma_i = v_i - w_i \tag{2.4}$$

con v_i el valor del mejor objeto $\quad v_i = \max_{j \in A(i)} \{a_{ij} - p_j\}$ (2.5)

y con w_i el valor del segundo mejor objeto $\quad w_i = \max_{j \in A(i), j \neq j_i} \{a_{ij} - p_j\}$ (2.6)

Note que como p_{j_i} es incrementado, el valor $a_{ij_i} - p_{j_i}$ ofrecido por el objeto j_i a la persona i se decrementa. γ_i es el máximo incremento por el cual p_{j_i} puede ser incrementado, mientras se mantiene la propiedad de que j_i ofrece valor máximo a i.

Este proceso se repite hasta que cada persona tiene un objeto asignado.

En cada iteración de la subasta, la persona i incrementa el precio del objeto que prefiere, por un incremento en la oferta de γ_i, donde γ_i no puede ser negativo dado que $v_i \geq w_i$. (Ver las ecuaciones (2.5) y (2.6)), así que los precios de los objetos tienden a incrementarse. La elección γ_i se ilustra en la figura 2.2

Tal como en una subasta real, incrementos en las pujas y en los precios hacen que el objeto seleccionado por el que realiza la puja sea menos atractivo para los demás participantes en la subasta.

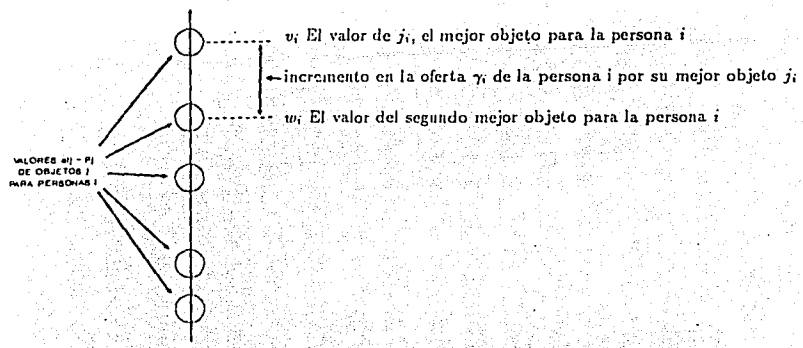


Figura 2.2 En el algoritmo de subasta inexperto, aún después de que el mejor objeto j_i es incrementado por una oferta γ_i, j_i continúa siendo el mejor objeto para la persona i, así que las condiciones de holgura complementarias se satisfacen al final de la iteración. Sin embargo, $\gamma_i = 0$ si existe un empate entre dos o más objetos que sean los más preferidos por la persona i.

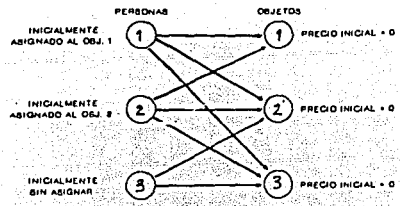
2.2.5 ε–HOLGURAS COMPLEMENTARIAS

Desafortunadamente, el algoritmo de subasta inexperto no siempre trabaja (pero a pesar de esto, es un procedimiento de inicialización excelente para otros métodos, como el primal-dual o el de relajación y es útil en otros contextos, por ejemplo rutas más cortas). La dificultad es que el incremento propuesto γ_i es cero cuando dos o más objetos ofrecen valor máximo para el participante i. Como resultado de esto, se puede crear una situación donde varias personas compiten por un número pequeño de objetos igualmente deseables sin incrementar sus precios creando un ciclo que nunca termine. Esto se observa en la figura 2.3

Para romper tales ciclos, se introduce un mecanismo de perturbación, motivado por las subastas verdaderas donde cada oferta por un objeto debe aumentarle el precio, aunque sea mínimamente, y las personas deben tomar riesgos para ganar sus objetos preferidos. En particular, fijemos un escalar positivo ε, decimos que una asignación parcial y un vector de precios P satisfacen $\varepsilon - Holguras\ Complementarias$ ($\varepsilon - H.C.$) si

$$a_{ij} - p_j \geq \max_{k \in A(i)} \{a_{ik} - p_k\} - \varepsilon \qquad (2.7)$$

para todos los pares asignados (i, j). En otras palabras, para satisfacer $\varepsilon - H.C.$, todas las personas asignadas parcialmente deben estar asignadas a objetos que están a ε de distancia de ser los mejores.



$a_{ij} = c > 0$ para todo (i, j) con $i = 1, 2, 3$ y $j = 1, 2$ y $a_{ij} = 0$ para todo (i, j) con $i = 1, 2, 3$ y $j = 3$

Al inicio de la iteración #	Precios de los objetos	Asignación	Persona Ofertante	Objeto Preferido	Incremento en la oferta
1	0, 0, 0	(1, 1), (2, 2)	3	2	0
2	0, 0, 0	(1, 1), (3, 2)	2	2	0
3	0, 0, 0	(1, 1), (2, 2)	3	2	0

Figura 2.3 Ejemplo de como el algoritmo subasta inexperto puede no terminar nunca. Este ejemplo involucra 3 personas y 3 objetos. Aquí, los objetos 1 y 2 ofrecen un beneficio $c > 0$ a todas las personas y el objeto 3 ofrece un beneficio de 0 a todas las personas. El algoritmo entra en un ciclo por que las personas 2 y 3 desean alternadamente el objeto 2 sin incrementar el precio, porque prefieren igualmente el objeto 1 y el objeto 2 (por lo que $\gamma_i = 0$, vea la figura 2.2)

Ahora se puede reformular el proceso de subasta previo, de tal manera que el incremento de la oferta sea siempre al menos igual a ε. El método resultante, el algoritmo de subasta, es el mismo que el algoritmo de subasta inexperto, excepto que el incremento γ_i es

$$\gamma_i = v_i - w_i + \varepsilon \tag{2.8}$$

que es mayor que $\gamma_i = v_i - w_i$ de la ecuación (2.4). Con esta elección, las condiciones de $\varepsilon - H.C.$ se satisfacen como se ilustra en la figura 2.4

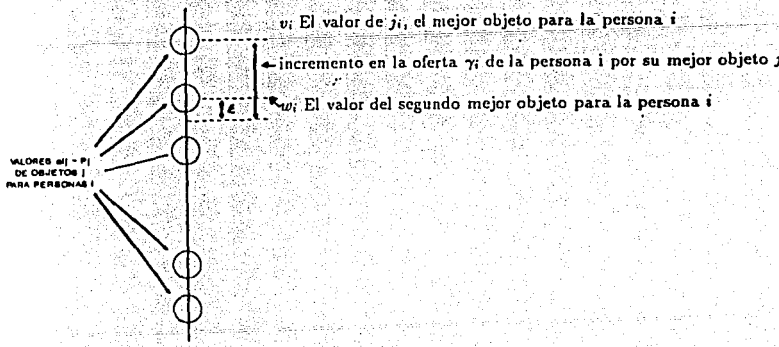
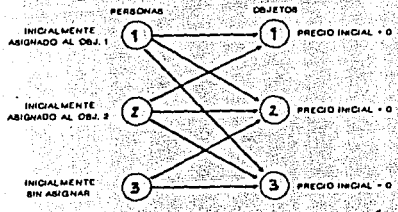


Figura 2.4 En el algoritmo subasta, aún después de que el precio del objeto preferido j_i es incrementado por una oferta γ_i, j_i estará a ε de ser el más preferido, así que la condición de ε-HC se mantiene al final de la iteración.

El incremento particular $\gamma_i = v_i - w_i + \varepsilon$ usado en el algortimo subasta es la cantidad máxima con esta propiedad. Incrementos menores γ_i pueden trabajar tanto como $\gamma_i \geq \varepsilon$, pero usando el mayor incremento posible se acelera el algoritmo. Lo que es consistente con la experiencia de las subastas reales, que tienden a terminar más rápido cuando las ofertas son agresivas.

Para demostrar que este proceso termina necesariamente con una asignación factible y un conjunto de precios que satisfacen $\varepsilon - H.C$, note que si un objeto recibe una oferta en m iteraciones, su precio debe exceder a su precio inicial en al menos $m\varepsilon$. Así para una m suficientemente larga, el "objeto" será suficientemente caro para ser juzgado "inferior" a algún objeto que no ha recibido una oferta. Por lo que únicamente para un número limitado de iteraciones un objeto puede recibir ofertas mientras que otro objeto no recibe ninguna oferta. Por otra parte, una vez que todo objeto ha recibido al menos una oferta , la subasta termina.

La figura 2.5 muestra como el algoritmo subasta, basado en el incremento $\gamma_i = v_i - w_i + \varepsilon$ termina el problema del ciclado del ejemplo en la figura anterior



$a_{ij} = c > 0$ para todo (i,j) con $i = 1,2,3$ y $j = 1,2$ y $a_{ij} = 0$ para todo (i,j) con $i = 1,2,3$ y $j = 3$

Al inicio de la iteración #	Precios de los objetos	Asignación	Persona Ofertante	Objeto Preferido	Incremento en la oferta
1	0, 0, 0	(1,1), (2,2)	3	2	ε
2	0, ε, 0	(1,1), (3,2)	2	1	2ε
3	2ε, ε, 0	(2,1), (3,2)	1	2	2ε
4	2ε, 3ε, 0	(1,2), (2,1)	3	1	2ε
5	4ε, 3ε, 0	(1,2), (3,1)	2	2	2ε
6	...			1	

Figura 2.5 Ejemplo de como el algoritmo subasta corrige el problema de ciclado del ejemplo de la figura 2.3, haciendo un incremento en la oferta en al menos ε. La tabla muestra una secuencia posible de las ofertas y asignaciones generadas por el algoritmo subasta, iniciando con todos los precios iguales a cero y con la asignación parcial $\{(1,1),(2,2)\}$. En cada iteración, excepto la última, la persona asignada al objeto 3 ofrece por el objeto 1 o 2, incrementando el precio en ε en la primera iteración y por 2ε en las subsecuentes. En la última iteración, después de que los precios de 1 y 2 alcanzan o exceden c, el objeto 3 recibe una oferta y la subasta termina.

Cuando el algoritmo de subasta termina, se tiene una asignación que satisface $\varepsilon - H.C.$, pero ¿Es una asignación óptima? La respuesta depende fuertemente del tamaño de ε. En una subasta real, una persona prudente no desea hacer una oferta excesivamente alta para obtener el objeto a un precio innecesariamente alto. Consistente con esta intuición, podemos demostrar que si ε es pequeña, entonces la asignación final será "casi óptima". En particular, demostraremos que el beneficio total de la asignación final está a $n\varepsilon$ de ser óptima. La idea es que una asignación factible y un conjunto de precios satisfaciendo $\varepsilon - H.C.$ puede ser visto como uno que satisface $H.C.$ para un problema ligeramente diferente, donde todos los beneficios a_{ij} son los mismos que antes, excepto en los beneficios de los n pares asignados, que son modificados en no más de ε.

Proposición. 2.2 Una asignación factible que satisface $\varepsilon - H.C$ junto con un vector de precios, está dentro de $n\varepsilon$ de ser óptimo. Además, el vector de precios está dentro de $n\varepsilon$ de ser una solución óptima del problema dual.

Demostración. Sea A^* el beneficio total de la asignación óptima

$$A^* = \max_{k_i,\ i=1,..,n \atop k_i \neq k_m \text{ si } i \neq m} \sum_{i=1}^{n} a_{ik_i}$$

y sea D^* el costo dual óptimo

$$D^* = \min_{\substack{p_j \\ j=1,..,n}} \left\{ \sum_{i=1}^{n} \max_{j \in A(i)} \{a_{ij} - p_j\} + \sum_{i=1}^{n} p_j \right\}$$

Si $\{(i, j_i) \mid i = 1, .., n\}$ es la asignación dada que satisface las condiciones de $\varepsilon - H.C$ junto con un vector de precios $\bar{p}$, tenemos

$$\max_{j \in A(i)} \{a_{ij} - p_j\} - \varepsilon \leq a_{ij_i} - \bar{p}_{j_i}$$

sumando sobre todo i esta relación, se ve que

$$D^* \leq \sum_{i=1}^{n} (\max_{j \in A(i)} \{a_{ij} - \bar{p}_j\} + \bar{p}_{j_i}) \leq \sum_{i=1}^{n} a_{ij_i} + n\varepsilon \leq A^* + n\varepsilon$$

Dado que hemos demostrado en la proposición 2.1 que $A^* = D^*$, se sigue que el beneficio de la asignación total $\sum_{i=1}^{n} a_{ij_i}$ está a menos de $n\varepsilon$ del valor óptimo A^*, mientras el costo dual de $\bar{p}$ está a menos de $n\varepsilon$ del costo dual óptimo. □

Si los beneficios a_{ij} son todos enteros, que en la practica es el caso típico (Si los a_{ij} son números racionales, pueden ser transformados a enteros multiplicandolos por un número conveniente), entonces el beneficio total de cualquier asignación es entera, así si $n\varepsilon < 1$, cualquier asignación completa que esté a menos de $n\varepsilon$ de ser óptima, debe serlo. Se sigue entonces que si $\varepsilon < 1/n$ y los coeficientes a_{ij} son todos enteros, entonces la asignación obtenida al término del algoritmo subasta es óptima.

La figura 2.6 muestra la secuencia de los precios de los objetos generados para el ejemplo de la figura anterior, en relación a los contornos de la función de costo dual.

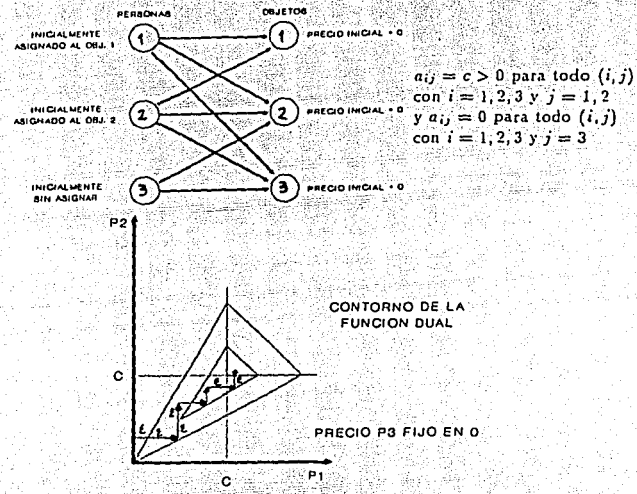


Figura 2.6 Secuencia de precios P_1 y P_2 generados por el algoritmo subasta para el ejemplo de la figura 2.3 y de la figura 2.5. La figura muestra las superficies de igual costo dual en el espacio P_1 y P_2 con P_3 fijo en cero.

Se puede ver en la figura 2.6 que cada oferta tiene el efecto de colocar el precio del objeto que recibe la oferta, casi igual (a menos de $n\varepsilon$) al precio que minimiza el costo dual con respecto a ese precio, con todos los otros precios mantenidos constantes. La minimización sucesiva de una función de costo a través de coordenadas simples es una tarea central del descenso coordinado y de los métodos de relajación, que son populares para la minimización no restringida de funciones suaves y para resolver sistemas de ecuaciones no lineales. Así, veremos que el algoritmo subasta puede ser interpretado como un método de descenso coordinado aproximado, por lo que está relacionado con el método de relajación.

3 METODOS DE ASCENSO DUAL

En la primera sección de este capítulo se obtiene el problema dual del problema del flujo a costo mínimo, así como las condiciones bajo las cuales, soluciones a cada uno de los problemas son óptimas. Posteriormente, se tratan las ideas fundamentales de los algoritmos generales de ascenso dual para el problema de FCM y se describen detalladamente dos métodos: el primal-dual y el de relajación. El primero utiliza la dirección elemental con máxima derivada direccional para mejorar el costo dual, mientras que el método de relajación utiliza direcciones que no son necesariamente de máximo crecimiento, pero que se calculan fácil y eficientemente. En la última subsección se presenta el método de relajación con iteración en un solo nodo, que permite ahorrar cálculos y hace más eficiente la implementación computacional.

3.1 DUALIDAD DEL PROBLEMA DE FCM

El problema de flujo a costo mínimo se puede escribir de la siguiente manera

$$\begin{aligned} &\underset{s.a}{Min} \sum_{(i,j)\in A} a_{ij}x_{ij} \quad \text{(P)} \\ &\sum_{\{j|(i,j)\in A\}} x_{ij} - \sum_{\{j|(j,i)\in A\}} x_{ji} = s_i \qquad \forall\, i \in N \\ &b_{ij} \le x_{ij} \le c_{ij} \qquad \forall\,(i,j) \in A \end{aligned} \tag{3.1}$$

para obtener el problema dual, consideramos a p_i como un multiplicador de Lagrange asociado a la restricción de conservación de flujo para el nodo i, así la función Lagrangiana es

$$\begin{aligned} L(X,P) &= \sum_{(i,j)\in A} a_{ij}x_{ij} + \sum_{i\in N}\left(s_i - \sum_{\{j|(i,j)\in A\}} x_{ij} + \sum_{\{j|(j,i)\in A\}} x_{ji}\right)p_i \\ &= \sum_{(i,j)\in A} (a_{ij} + p_j - p_i)\,x_{ij} + \sum_{i\in N} s_i p_i \end{aligned} \tag{3.2}$$

entonces el valor de la función dual $Q(P)$ en un vector P, se obtiene minimizando $L(X,P)$ sobre todos los flujos de capacidad factible

$$Q(P) = \underset{x}{Min}\ \{L(X,P) \mid b_{ij} \le x_{ij} \le c_{ij},\ (i,j) \in A\}$$

como la función lagrangiana $L(X, P)$ es separable, su minimización se lleva para cada uno de los arcos (i, j). Cada una de estas minimizaciones pueden ser escritas, finalmente, de la siguiente forma

$$Q(P) = \sum_{(i,j)\in A} q_{ij}(p_i - p_j) + \sum_{i\in N} s_i p_i$$

donde
$$q_{ij}(p_i - p_j) = \min_{x_{ij}} \{(a_{ij} + p_j - p_i)x_{ij} \mid b_{ij} \le x_{ij} \le c_{ij}\} \qquad (3.3)$$

$$= \begin{cases} (a_{ij} + p_j - p_i)b_{ij} & \text{Si } p_i \le a_{ij} + p_j \\ (a_{ij} + p_j - p_i)c_{ij} & \text{Si } p_i > a_{ij} + p_j \end{cases}$$

la forma de la función q_{ij} se muestra en la figura 3.1

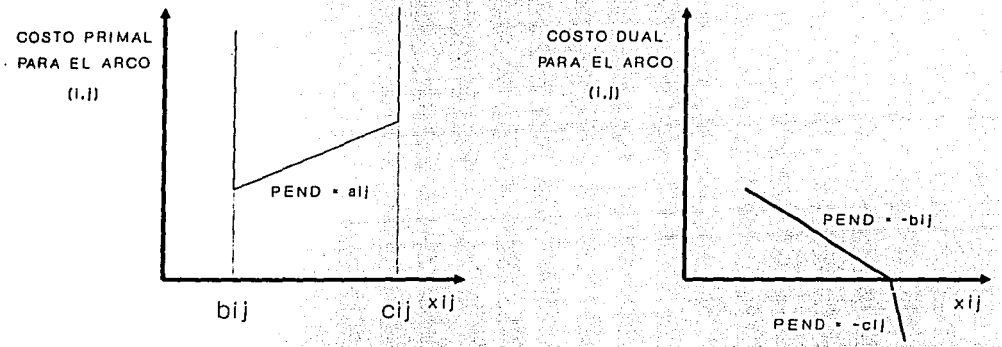


Figura 3.1 Costos primal y dual para el arco (i, j). Note que los pntos de ruptura de cada función corresponden a pendientes de segmentos lineales en la otra función.

por lo que el problema dual (D) es

$$\underset{s.a.}{Max}\ Q(P) \qquad (D)$$

Ninguna restricción en P

Es necesario introducir alguna terminología: Para cualquier vector de precios P, decimos que un arco (i, j) es

inactivo si $p_i < a_{ij} + p_j$

balanceado si $p_i = a_{ij} + p_j$ (3.4)

activo si $p_i > a_{ij} + p_j$

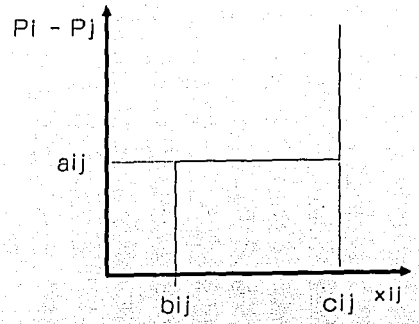


Figura 3.2 Ilustración de las condiciones de H.C. Para cada arco (i,j) el par $(x_{ij}, p_i - p_j)$ debe ubicarse en la gráfica mostrada.

Las condiciones de holguras complementarias para el vector de flujos-precios (X,P) son

$$\begin{array}{lll} \text{Si } p_i - p_j \leq a_{ij} \quad \forall\,(i,j) \in A & x_{ij} < c_{ij} & \\ \text{Si } p_i - p_j \geq a_{ij} \quad \forall\,(i,j) \in A & b_{ij} < x_{ij} & \end{array} \Rightarrow \begin{array}{l} p_i = a_{ij} + p_j \;\forall\,(i,j) \in A \\ \text{tal que } b_{ij} < x_{ij} < c_{ij} \end{array} \tag{3.5}$$

o equivalentemente

$$\text{Si } \forall\,(i,j)\ \ b_{ij} \leq x_{ij} \leq c_{ij}$$

$$x_{ij} = \begin{cases} c_{ij} \text{ si } p_i > a_{ij} + p_j \\ b_{ij} \text{ si } p_i < a_{ij} + p_j \end{cases} \tag{3.6}$$

que puede reescribirse como sigue

$$\begin{array}{ll} x_{ij} = b_{ij} & \forall\,(i,j) \text{ inactivo} \\ b_{ij} \leq x_{ij} \leq c_{ij} & \forall\,(i,j) \text{ balanceado} \\ x_{ij} = c_{ij} & \forall\,(i,j) \text{ activo} \end{array} \tag{3.7}$$

En la figura 3.2 se ilustran las condiciones de holguras complementarias.

El siguiente es un resultado básico de la teoría de dualidad

Proposición 3.1 Si un vector de flujo factible X^* y un vector de precios P^* satisfacen las condiciones de holguras complementarias, entonces X^* es una solución primal óptima y P^* es una solución dual óptima. Además, el costo óptimo primal y el costo óptimo dual son iguales.

Demostración. Primero demostraremos que para cualquier vector de flujo factible X y para cualquier vector de precios P, el costo primal de X no es menor que el costo dual de P.

Así, tenemos que

$$Q(P) \leq L(X \cdot P)$$

$$= \sum_{(i,j)\in A} a_{ij}x_{ij} + \sum_{i\in N}\left(s_i - \sum_{\{j|(i,j)\in A\}} x_{ij} + \sum_{\{j|(j,i)\in A\}} x_{ji}\right)p_i = \sum_{(i,j)\in A} a_{ij}x_{ij} \tag{3.8}$$

donde la última desigualdad se debe a la factibilidad de X. Por otro lado, tenemos que

$$Q(P^*) = \min_X \{\, L(X;P^*) \mid b_{ij} \leq x_{ij} \leq c_{ij},\ (i,j) \in A \,\}$$

$$= L(X^*;P^*) = \sum_{(i,j)\in A} a_{ij}x^*_{ij} \tag{3.9}$$

donde la segunda igualdad es verdadera porque (X^*, P^*) satisface H.C. si y sólo si x^*_{ij} minimiza $(a_{ij} + p^*_j - p^*_i)x_{ij}$ sobre todo $x_{ij} \in [b_{ij}, c_{ij}]$, para todo $(i,j) \in A$.

y la última igualdad se sigue de la expresión lagrangiana (3.2) y la factibilidad de X^*.

Así, X^* alcanza el mínimo del costo primal en el lado derecho de (3.8) y P^* alcanza el máximo de $Q(P)$ en el lado izquierdo de la ecuación (3.8), mientras que los valores optimos primal y dual son iguales. □

El más importante de los algoritmos de ascenso dual selecciona en cada iteración un subconjunto conexo de nodos S y cambia los precios de estos nodos en cantidades iguales mientras que deja los precios de los otros nodos sin cambio. En otras palabras, cada iteración involucra un cambio en el vector de precios a través de una dirección de la forma $d_S = (d_1, .., d_n)$ con

$$d_i = \begin{cases} 1 & \text{si } i \in S \\ 0 & \text{si } i \notin S \end{cases} \quad \text{y } S \text{ un subconjunto conexo de nodos.}$$

Tales direcciones son llamadas elementales. Para checar si d_S es una dirección de ascenso dual, necesitamos calcular la derivada direccional correspondiente del costo dual hacia d_S y checar si es positiva

De la expresión (3.3), tenemos que la derivada direccional es

$$q'(P;d_S) = \lim_{\alpha \downarrow 0} \frac{q(P+d_S)-q(P)}{\alpha}$$

$$= \sum_{(j,i):activo, j\notin S, i\in S} c_{ji} + \sum_{(j,i):inactivo\ o\ balanceado, j\notin S, i\in S} b_{ji} -$$

$$- \sum_{(i,j):act.\ o\ balanceado, i\in S, j\notin S} c_{ij} - \sum_{(i,j):inactivo, j\notin S, i\in S} b_{ij} + \sum_{i\in S} s_i \qquad (3.10)$$

En palabras, la derivada direccional $q'(P;d_S)$ es la diferencia entre el flujo que llega y el flujo que sale a través del conjunto de nodos S cuando los flujos de los arcos activos o inactivos son colocados en su cota superior e inferior, respectivamente, y el flujo de cada arco balanceado incidente a S es colocado en su cota inferior o superior dependiendo de si el arco tiene su vértice final en S o fuera de S.

Para obtener un conjunto adecuado S, con derivada direccional positiva $q'(P;d_S)$, es conveniente mantener un vector de flujo X, satisfaciendo H.C. junto con P. Esto ayuda a organizar la búsqueda de una dirección de ascenso y enseguida explicaremos como detectar optimalidad.

Para un vector de flujo X, definamos la *holgura g_i del nodo i* como la diferencia entre el flujo total que llega a i menos el flujo total que sale de i, esto es

$$g_i = \sum_{\{j|(j,i)\in A\}} x_{ji} - \sum_{\{j|(i,j)\in A\}} x_{ij} + s_i \qquad (3.11)$$

Tenemos

$$\sum_{i\in S} g_i = \sum_{\{(j,i)\in A | j\notin S, i\in S\}} x_{ji} - \sum_{\{(i,j)\in A | j\notin S, i\in S\}} x_{ij} + \sum_{i\in S} s_i \qquad (3.12)$$

y si X satisface H.C. con P, usando (3.10) y (3.12)

$$\sum_{i\in S} g_i = q'(P;d_S) + \sum_{(j,i):balanceados, j\notin S, i\in S} (x_{ji} - b_{ji}) + \sum_{(i,j):balanceados, i\in S, j\notin S} (c_{ij} - x_{ij}) \qquad (3.13)$$
$$\geq q'(P;d_S)$$

Así, únicamente los conjuntos de nodos S que tienen holgura total positiva son candidatos para generar una dirección d_S de ascenso dual. En particular , si no hay arcos balanceados (i,j) con $i \in S$, $j \notin S$ y $x_{ij} < c_{ij}$ y no hay arcos balanceados (j,i) con $j \notin S$, $i \in S$ y $b_{ji} < x_{ji}$

$$\sum_{i\in S} g_i = q'(P;d_S) \qquad (3.14)$$

así, si S tiene una holgura total positiva entonces d_S es una dirección de ascenso. El siguiente lema expresa esta idea y proporciona las bases para los algoritmos subsecuentes.

Lema 3.2 Suponga que X y P satisfacen las condiciones de H.C. y sea S un subconjunto de nodos. Sea $d_S = (d_1, .., d_n)$ un vector con $d_i = 1$ si $i \in S$ y $d_i = 0$ si $i \notin S$, supongamos que $\sum_{i \in S} g_i > 0$. Entonces o bien d_S es una dirección de ascenso dual, esto es $q'(P; d_S) > 0$ o existen nodos $i \in S$ y $j \notin S$ tal que (i, j) es un arco balanceado con $x_{ij} < c_{ij}$ o bien (j, i) es un arco balanceado con $b_{ji} < x_{ji}$.

Demostración. Se sigue de la ecuación (3.13). □

3.2 VISTAZO DE LOS ALGORITMOS DE ASCENSO DUAL

Estos algoritmos comienzan con un vector entero de flujo-precio (X, P), que satisface H.C. Al inicio de cada iteración, se tiene un subconjunto de nodos S tal que $\sum_{i \in S} g_i > 0$.

Inicialmente S consiste de uno o más nodos con holgura positiva, de acuerdo al lema 3.2 de la sección anterior, existen dos posibilidades

a) S define una dirección de ascenso dual $d_S = (d_1, .., d_n)$ donde $d_i = 1$ si $i \in S$ y $d_i = 0$ si $i \notin S$

b) S puede crecer agregando un nodo $j \notin S$ con la propiedad descrita en el lema, es decir,para algún $i \in S$, (i, j) es un arco balanceado con $x_{ij} < c_{ij}$, o (j, i) es un arco balanceado con $b_{ji} < x_{ji}$

En el caso b), hay dos posibilidades

1. Si $g_j \geq 0$, entonces $\sum_{i \in S \cup \{j\}} g_i > 0$ y el proceso puede continuar con $S \cup \{j\}$ reemplazando a S
2. $g_j < 0$, en cuyo caso, se puede ver que existe una cadena no *bloqueada* comenzando en algún nodo i del conjunto inicial S y finalizando en el nodo j, es decir, en todos sus arcos se puede modificar el flujo para obtener un incremento en la dirección de i a j. Tal cadena es llamada *cadena aumentante*. Al aumentar el flujo en los arcos de la cadena con dirección de i a j y decrementando en los arcos de dirección de j a i, podemos acercar a cero las hoguras g_i y g_j en una cantidad entera, dejando las holguras de todos los demás nodos sin modificar, manteniendo holguras complementarias.

Como la holgura total absoluta $\sum_{i \in N} \mid g_i \mid$ no puede ser reducida indefinidamente utilizando cantidades enteras, se nota que iniciando con un vector entero de flujo-precio que satisfaga H.C. y después de un número finito de iteraciones en las que incrementos de flujo ocurren sin encontrar una dirección de ascenso, una de las siguientes tres cosas ocurren

1. Una dirección de ascenso dual es encontrada; esta dirección puede usarse para mejorar el costo dual en una cantidad entera

2. $g_i = 0 \ \forall \ i$; en este caso el vector de flujo X es factible y como satisface H.C. junto con P, X es un óptimo primal y P es un óptimo dual.

3. $g_i \leq 0 \ \forall \ i$ pero $g_i < 0$ para al menos un nodo i; De la ecuación (3.12) al sumar sobre todo $i \in N$ tenemos $\sum_{i \in N} S_i = \sum_{i \in N} g_i$ de lo que se sigue que $\sum_{i \in N} S_i < 0$, así que el problema es no factible.

Por lo tanto, para un problema factible el procedimiento mencionado puede ser usado para encontrar una dirección de ascenso dual y mejorar con esto el costo dual, inicando con cualquier vector de precios no óptimo. Una aplicación de este algoritmo se ilustra en la figura 3.3

Enseguida se discuten dos diferentes métodos de ascenso dual. El primero, conocido como primal-dual, en su forma clásica, trata en cada iteración de usar la dirección elemental con máxima derivada direccional (steepest ascent direction). Este método también puede implementarse por medio de una subrutina de rutas más cortas. El segundo método, llamado *relajación*, es usualmente más rápido en la práctica. Trata de usar direcciones que no son necesariamente de máximo crecimiento, pero que pueden ser calculadas más eficientemente que las direcciones de ascenso máximo.

3.3 METODO PRIMAL-DUAL

El algoritmo primal-dual comienza con cualquier par (X, P) que cumpla H.C. Una posibilidad es escoger el vector entero P arbitrariamente y hacer $x_{ij} = b_{ij}$ si (i, j) es inactivo o balanceado y $x_{ij} = c_{ij}$ en otro caso. El algoritmo preserva la integralidad y la propiedad del par (X, P).

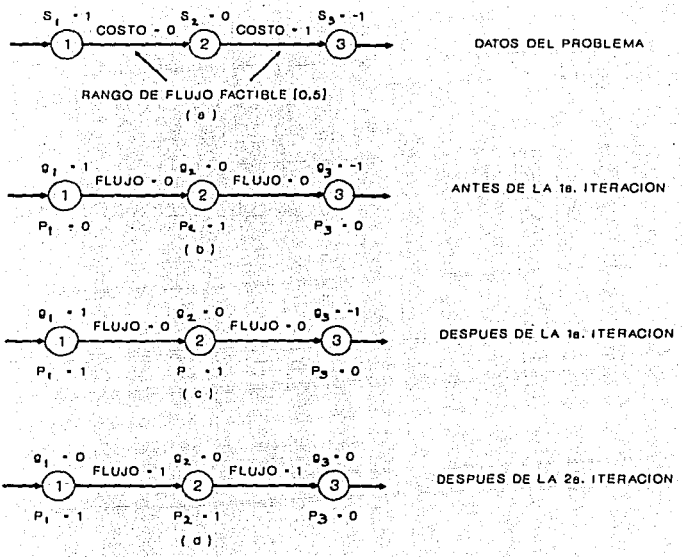


Figura 3.3 Ilustración del método de ascenso dual para el problema mostrado en (a). Inicialmente $x = (0,0)$ y $P = (0,0,0)$ como se aprecia en (b)

La primera iteración comienza con $S = \{1\}$. Se puede ver, usando la ecuación (3.13), que la derivada direccional $q'(P;d_S) = -4$, así que $d_S = (1,0,0)$ no es una dirección de ascenso. Agrandamos S al añadirle el nodo 2 usando el arco balanceado (1,2). Como no hay un arco balanceado incidente a $S = \{1,2\}$, la dirección $d_S = (1,1,0)$ es una dirección de ascenso (Usando la ecuación 3.13, $q'(P;d_S) = 1$). Incrementamos los precios de los nodos en S en $\gamma = 1$, porque este es el incremento que maximiza la función dual a través de la dirección d_S comenzando en P; esto puede ser visto checando la derivada direccional de q en el vector de precios $P = (\gamma, \gamma, 0)$ a través de la dirección d_S y encontrando que cambia de positiva (=1) a negativa (=-4) en $\gamma = 1$, donde el arco (2,3) llega a ser balanceado.

La segunda iteración comienza otra vez con $S = \{1\}$. Como en la primera iteración S es agrandado a $S = \{1,2\}$. Como la correspondiente dirección $d_S = (1,1,0)$ no es una dirección de ascenso ($q'(P;d_S) = -4$), al explorar el arco balanceado incidente (2,3) descubrimos la holgura negativa del nodo 3. La trayectoria aumentante (1,2,3) ha sido obtenida y la correspondiente aumentación coloca los flujos de los arcos (1,2) y (2,3) en 1. Como todas las holguras son cero, el algoritmo termina; $X = (1,1)$ es una solución óptima primal y $P = (1,1,0)$ es una solución óptima dual.

Al inicio de una iteración típica, tenemos un par entero (X, P) que satisface H.C. La iteración indica si el problema primal es no factible, indica si (X, P) es óptimo, o transforma este par de vectores en otro par que también satisface H.C.; en particular si $g_i \leq 0$ para todo nodo i , en vista del hecho $\sum_{i \in N} S_i = \sum_{i \in N} g_i$ [ecuación (3.12) con $S = N$] existen dos posibilidades

1. $g_i < 0$ para algún i, en cuyo caso $\sum_{i \in N} S_i < 0$ y el problema es no factible o
2. $g_i = 0 \ \forall \ i$, en cuyo caso X es factible y además óptimo, porque satisface H.C. junto con P.

En cualquier caso el algoritmo termina.

De otra forma tenemos $g_i > 0$ para al menos un nodo i, la iteración inicia seleccionando un subconjunto no vacío I de nodos i con $g_i > 0$. La iteración mantiene dos conjuntos de nodos S y L con $S \subset L$. Inicialmente, S es vacío y L consiste del conjunto I. Usaremos la siguiente terminología.

S conjunto de nodos revisados (Estos son los nodos cuyos arcos incidentes han sido examinados durante la iteración).

L conjunto de nodos etiquetados (Estos son los nodos que no han sido revisados durante la iteración o son candidatos actuales para la revisión).

En el transcurso de la iteración continuamos agregando nodos a L y S hasta que una cadena aumentante es encontrada o $L=S$, en cuyo caso demostraremos que d_S es una dirección de ascenso. La iteración también mantiene una etiqueta para todo nodo $i \in L-I$ que es un arco incidente de i. Estas etiquetas son útiles para la construcción de cadenas aumentantes.

ITERACION TIPICA DEL ALGORITMO PRIMAL-DUAL

PASO 0 (Inicialización)

Seleccionar un conjunto I de nodos con $g_i > 0$ (Si no hay ningún nodo, terminar; el par (X, P) es óptimo si $g_i = 0$ para todo nodo i; de otra manera el problema es no factible). Hacer $L = I$ y $S = \emptyset$, ir al paso 1.

PASO 1 (Escoger un nodo para revisar)

Si $L = S$, ir al paso 4, si no seleccionar un nodo $i \in L - S$, hacer $S = S \cup \{i\}$ e ir al paso 2.

PASO 2 (Etiquetación de los nodos vecinos de i)

Agregar a L todos los nodos $j \notin L$ tal que (j, i) es balanceado y $b_{ji} < x_{ji}$ o (i, j) es balanceado y $x_{ij} < c_{ij}$; para tales j, dar a j las etiquetas "(j, i)" en el primer caso y "(i, j)" en el segundo. Si para todos los nodos j agregados a L se tiene $g_j \geq 0$ ir al paso 1. De otra forma, escoger uno de estos nodos j con $g_j < 0$ e ir al paso 3.

PASO 3 (Aumentación de flujo)

Una cadena aumentante P ha sido encontrada comenzando en un nodo i perteneciente al conjunto inicial I y terminando en el nodo j identificado en el paso 2. Esta cadena es construida rastreando regresivamente etiquetas, a partir del nodo j, y se tiene que:

$$x_{mn} < c_{mn} \quad \forall\, (m, n) \in \mathrm{P}^+$$

$$x_{mn} > b_{mn} \quad \forall\, (m, n) \in \mathrm{P}^-$$

con P^+ y P^- los conjuntos de arcos positivos y negativos de la cadena. Sea

$$\delta = \min\{g_i, -g_j, \{c_{mn} - x_{mn} \mid (m, n) \in \mathrm{P}^+\}, \{x_{mn} - b_{mn} \mid (m, n) \in \mathrm{P}^-\}\}$$

Incrementar en δ el flujo de todos los arcos en P^+, decrementar en δ el flujo de todos los arcos en P^- y realizar otra iteración.

PASO 4 (Cambio de precios)

Sea $\gamma = \min\{\{p_j + a_{ij} - p_i \mid (i, j) \in A, x_{ij} < c_{ij}, i \in S, j \notin S\},$

$$\{p_j + a_{ji} - p_i \mid (j, i) \in A, b_{ji} < x_{ji}, i \in S, j \notin S\} \tag{3.15}$$

Hacer $$p_i = \begin{cases} p_i + \gamma & \text{Si} \quad i \in S \\ p_i & \text{En otro caso.} \end{cases}$$

Agregar a L todos los nodos j para los cuales el mínimo en (3.15) se logra por un arco (i,j) o un arco (j,i); también para tales j, dar a j la etiqueta "(i,j)" si el mínimo se logra con un arco (i,j), de otra manera dar a j la etiqueta "(j,i)".Si para todos los nodos j agregados a L se tiene $g_j \geq 0$, ir al paso 1. Si no, seleccionar uno de estos nodos j con $g_j < 0$ e ir al paso 3. {Nota: Si no existe un arco (i,j) con $x_{ij} < c_{ij}, i \in S$ y $j \notin S$, o un arco (j,i) con $b_{ji} < x_{ji}, i \in S$ y $j \notin S$, el problema es no factible y el algoritmo termina [Ver la proposición 3.3]}.

Note los siguientes puntos importantes de la iteración primal-dual

1. Todas las operaciones de la iteración preservan la integralidad del par de vectores flujo-precio.

2. La iteración mantiene H.C. en el par de vectores flujo-precios. Para observar esto , note que los arcos con ambos extremos en S, que están balanceados justo antes de un cambio de precios, continuán siendolo después de un cambio de precios. Esto significa que un paso en la aumentación de flujo, aún si ocurre seguida de varias ejecuciones del paso 4, cambia únicamente flujo de arcos balanceados, así que no puede destruir H.C. También, un cambio de precios en el paso 4 mantiene H.C. porque no modifica el flujo en los arcos y el incremento de precios de γ de (3.15) es tal que no hay cambios en la posición de activo a inactivo o viceversa.

3. En todo tiempo tenemos $S \subset L$. Aún más, cuando el paso 4 es alcanzado, se tiene $S = L$ y L no contiene nodos con holgura negativa. Además, basados en la lógica del paso 2, no hay arcos balanceados (i,j) con $x_{ij} < c_{ij}, i \in S$ y $j \notin S$, ni arcos balanceados (j,i) con $b_{ji} < x_{ji}, i \in S$ y $j \notin S$. Se sigue de la discusión precedente que d_S es una dirección de ascenso.

4. Unicamente un número finito de cambios en precios ocurre en cada iteración, así cada iteración se realiza hasta terminar con un flujo aumentante en el paso 3, o con la indicación de no factibilidad en el paso 4. Note que entre dos cambios de precio, el conjunto L es agrandado en al menos un nodo, así que no puede haber más de N cambios de precio por iteración.

5. Unicamente un número finito de pasos en la aumentación de flujo son ejecutados por el algoritmo, dado que cada uno de estos reduce la holgura absoluta total $\sum_{i \in N} \mid g_i \mid$ en una cantidad entera, (por 1) de arriba), mientras que cambios en los precios no afectan la holgura total absoluta.

6. El algoritmo termina. La razón es que en cada iteración se ejecuta completamente (por 4)) y envuelve exactamente una aumentación, mientras que hay únicamente un número finito de aumentaciones (por 3)).

La siguiente proposición establece la validez del método:

Proposición 3.3 Considere el problema de flujo a costo mínimo y suponga que a_{ij}, b_{ij}, c_{ij} y s_i son todos enteros.

a) Si el problema es factible, entonces el método primal-dual termina con un vector entero X de flujo óptimo y un vector entero P de precios óptimo.

b) Si el problema es no factible, entonces el método primal-dual termina a causa de que $g_i \leq 0$ para todo nodo i y $g_i < 0$ para al menos un nodo i o porque no hay un arco (i,j) con $x_{ij} < c_{ij}, i \in S$ y $j \notin S$ o un arco (j,i) con $b_{ji} < x_{ji}, i \in S$ y $j \notin S$ en el paso 4.

Demostración. El algoritmo termina como se argumento y existen tres posibilidades

1. El algoritmo termina porque todos los nodos tienén holgura cero. En este caso, el par de vectores flujo-precios obtenido al termino es factible y satisface H.C., por lo que es óptimo.

2. El algoritmo termina porque $g_i \leq 0 \; \forall \; i$ y $g_i < 0$ para al menos un i. En este caso, el problema es no factible, dado que para que un problema sea factible debemos tener $\sum_{i \in N} g_i = 0$.

3. El algoritmo termina porque no hay un arco (i,j) con $x_{ij} < c_{ij}, i \in S$ y $j \notin S$ o un arco (j,i) con $b_{ji} < x_{ji}, i \in S$ y $j \notin S$ en el paso 4. Entonces el flujo a través de la cortadura $Q = (S, N - S)$ es igual a la capacidad $C(Q)$ y también es igual a la suma de las divergencias de los nodos de S, que es $\sum_{i \in S}(s_i - g_i)$. Dado que $g_i \geq 0 \; \forall \; i \in S$, $g_i > 0$ para los nodos $i \in I$ e $I \subset S$, vemos que

$$C(Q) < \sum_{i \in S} s_i$$

Esto implica que el problema es no factible, dado que para cualquier vector de flujo factible debemos tener

$$\sum_{i \in S} s_i = F(Q) \leq C(Q)$$

donde $F(Q)$ es el flujo correspondiente a través de Q.

Como la terminación puede ocurrir únicamente bajo las circunstancias descritas, la conclusión deseada se sigue. □

Existen diferentes variaciones del método primal-dual, usando diferentes elecciones del conjunto inicial I de nodos con holgura positiva. Las dos posibilidades más comúnes son:

1. I consiste de un solo nodo i con $g_i > 0$
2. I consiste de todos los nodos i con $g_i > 0$

El método primal-dual fué originalmente propuesto con la última elección. En este caso, cuando hay un cambio de precios, el conjunto S contiene todos los nodos con holgura positiva, y de las fórmulas de las derivadas direccionales (3.13) y (3.14) se sigue que la dirección de ascenso usada en el paso 4 tiene la máxima derivada direccional posible entre las direcciones elementales.

Esto nos lleva a la interpretación del método primal-dual como un método de ascenso.

La figura 3.4 es un ejemplo que ilustra los pasos del método primal-dual.

3.4 EL METODO DE RELAJACION

Este método admite una implementación similar a la del método primal-dual pero calcula las direcciones de ascenso mucho más rápido. En particular, mientras que en el método primal-dual continuamos ampliando al conjunto S de nodos revisados hasta que sea igual al conjunto L de nodos etiquetados (en cuyo caso estamos seguros que d_S es una dirección de ascenso), en el método de relajación nos detendremos de agregar nodos a S inmediatamente después de que d_S llega a ser una dirección de ascenso. {Esto se hace calculando la derivada direccional $q'(P; d_S)$ usando un método

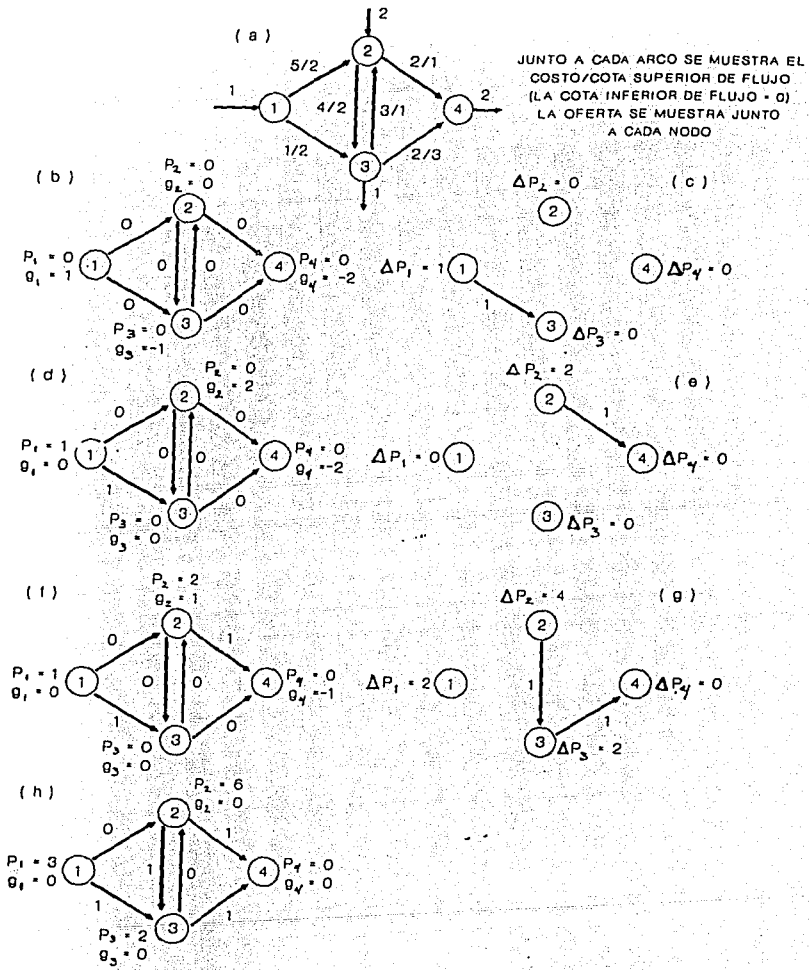


Figura 3.4 Ejemplo ilustrando el método primal-dual, comenzando con precios igual a cero

a) Datos del problema

b) Flujos, precios y holguras iniciales

c) Cadena aumentante y cambio de precios ΔP_i de la primera iteración ($I = \{1\}$).

d) Flujos, precios y holguras después de la primera iteración.

e) Cadena aumentante y cambio de precios ΔP_i de la segunda iteración ($I = \{2\}$).

f) Flujos, precios y holguras después de la segunda iteración.

g) Cadena aumentante y cambio de precios ΔP_i de la tercera iteración ($I = \{2\}$). En esta iteración ha 2 incrementos : primero P_2 se incrementa en 2 P_1, P_2 y P_3 se incrementa en 2.

h) Flujos, precios y holguras después de la tercera iteración. Comolas holguras son cero, el algoritmo termina y el flujo mostrado es óptimo.

incremental eficiente y checando el signo}. En la práctica, S a menudo consta de un sólo nodo, en cuyo caso la dirección de ascenso es una coordinación de un precio simple dando lugar a la interpretación del método como un método de ascenso coordinado. Sin embargo el método de relajación, no puede ser implementado usando un cálculo de R.M.C.

Como en el método Primal-Dual, al inicio de una iteración tenemos un par entero (X, P) que satisfacen H.C. La iteración indica que el problema primal es no factible, que (X, P) es óptimo o transforma este par en otro que satisface H.C. En particular si $g_i \leq 0$ para todo nodo i, hay dos posibilidades (1) $g_i < 0$ para algún nodo i, en cuyo caso $\sum_{i \in N} s_i < 0$ y el problema es no factible o (2) $g_i = 0$ para todo nodo i, en cuyo caso X es factible y por lo tanto óptimo, dado que satisface H.C. junto con P. En cualquier caso el algoritmo termina.

Por otra parte si tenemos $g_i > 0$ para al menos un nodo i, la iteración empieza seleccionando un nodo i^* con $g_{i^*} > 0$. Como en el método primal-dual, la iteración mantiene dos conjuntos de nodos S y L, con $S \subset L$. Al inicio de la iteración, S es vacío y L consiste del nodo i^* con $g_{i^*} > 0$. La iteración también mantiene una etiqueta para todos los nodos $i \in L$, excepto para el nodo inicial i^*; la etiqueta es un arco incidente de i.

ITERACION TIPICA DEL ALGORITMO DE RELAJACION

PASO 0 (Inicialización)

Seleccionar un nodo i^* con $g_{i^*} > 0$. {Si tal nodo no existe, terminar. El par (X, P) es óptimo si $g_i = 0$ para todo nodo i; de otra manera el problema es no factible} Hacer $L = \{i^*\}$ y $S = \emptyset$ e ir al paso 1.

PASO 1 (Escoger un nodo para revisar).

Si $S = L$ ir al paso 4, de otra manera seleccionar un nodo $i \in L - S$. Hacer $S = S \cup \{i\}$ e ir al paso 2.

PASO 2 (Etiquetación de los nodos vecinos de i).

$$\text{Si } q'(P; d_S) > 0 \tag{3.16}$$

ir al paso 4, de otra forma agregar a L todos los nodos $j \notin L$ tal que (j, i) es balanceado y

$b_{ji} < x_{ji}$, o (i,j) es balanceado y $x_{ij} < c_{ij}$; también para todo j, darle la etiqueta "(j,i)" si j es balanceado y $b_{ji} < x_{ji}$, de otra forma darle la etiqueta "(i,j)". Si para todo j, agregado a L, se tiene $g_j \geq 0$, ir al paso 3; si no seleccionar uno de los nodos j con $g_j < 0$ e ir al paso 3.

PASO 3 (Aumentación de flujo)

Una cadena aumentante P ha sido encontrada, que comienza en el nodo i^* y termina en el nodo j, identificado en el paso 2. La cadena es construida recorriendo etiquetas regresivamente desde el nodo j, de tal forma que

$$x_{mn} < c_{mn} \ \forall \ (m,n) \in \mathrm{P}^+ \tag{3.17}$$

$$x_{mn} > b_{mn} \ \forall \ (m,n) \in \mathrm{P}^- \tag{3.18}$$

donde P^+ y P^- son los conjuntos de arcos positivo y negativo de P, respectivamente.

Hacer

$$\delta = \min\{g_i, -g_j, \{c_{mn} - x_{mn} \mid (m,n) \in \mathrm{P}^+\}, \{x_{mn} - b_{mn} \mid (m,n) \in \mathrm{P}^-\}\}.$$

Incrementar en δ el flujo de todos los arcos en P^+, decrementar en δ el flujo de todos los arcos en P^- y realizar otra iteración.

Paso 4 (Cambio de precios)

Hacer

$$x_{ij} = c_{ij} \quad \forall \text{ arco balanceado } (i,j) \text{ con } i \in S, j \notin S \tag{3.19}$$

$$x_{ji} = b_{ji} \quad \forall \text{ arco balanceado } (j,i) \text{ con } i \in S, j \notin S \tag{3.20}$$

Hacer

$$\gamma = \min\{\{p_j + a_{ij} - p_i \mid (i,j) \in A, x_{ij} < c_{ij}, i \in S, j \notin S\},$$

$$\{p_j + a_{ji} - p_i \mid (j,i) \in A, b_{ji} < x_{ji}, i \in S, j \notin S\} \tag{3.21}$$

Hacer

$$p_i = \begin{cases} p_i + \gamma & \text{Si } \ i \in S \\ p_i & \text{En otro caso.} \end{cases} \tag{3.22}$$

Ir a la siguiente iteración

Nota: Como en el caso de la iteración primal-dual, si después de los ajustes de flujo de las ecuaciones (3.19) y (3.20) no hay un arco (i,j) con $x_{ij} < c_{ij}$, $i \in S$, $j \notin S$ o un arco (j,i) con $b_{ji} < x_{ji}$, $i \in S$ y $j \notin S$, entonces el problema es no factible y el algoritmo termina.

Se puede ver que la iteración del método de relajación es muy similar a la iteración del método primal-dual. Sin embargo, existen dos diferencias importantes. Primero, en la iteración de relajación, después de un cambio de precios en el paso 4, no regresamos al paso 1 a continuar la búsqueda de una cadena aumentante como en el caso del método primal-dual. Así la iteración de relajación termina con una aumentación como en el paso 3 o con un cambio de precios en el paso 4, en contraste con la iteración primal-dual, que únicamente puede terminar con una aumentación. La segunda y más importante diferencia es que en la iteración de relajación, un cambio de precios puede ser realizado en el paso 4, aún si $S \neq L$ (Ver la ecuación 3.16). Es a causa de esta característica que el método de relajación identifica direcciones de ascenso más rápido que el método primal-dual. Note que en contraste con el método primal-dual, la holgura total $\sum_{i \in N} \mid g_i \mid$ puede incrementarse como resultado de una iteración de relajación.

Una propiedad importante del método es que cada vez que entramos en el paso 4, d_S es una dirección de ascenso. Para ver esto, note que hay dos posibilidades: (1) tenemos $S = L$ (Ver paso 1) en cuyo caso d_S es una dirección de ascenso similar a la correspondiente en el método primal-dual o (2) tenemos $S \neq L$ (Ver paso 2) en cuyo caso, por la ecuación (3.16), también d_S es una dirección de ascenso.

Es posible combinar varias iteraciones del método de relajación en una sola iteración con el propósito de ahorrar tiempo de cálculo, y esto es hecho inteligentemente en los códigos RELAX que son implementaciones de dominio público de los métodos de relajación, estos códigos han sido elaborados en su mayoría por Bertsekas y su equipo.

La figura 3.5 ilustra los pasos del método con un ejemplo

La siguiente proposición establece la validez del método

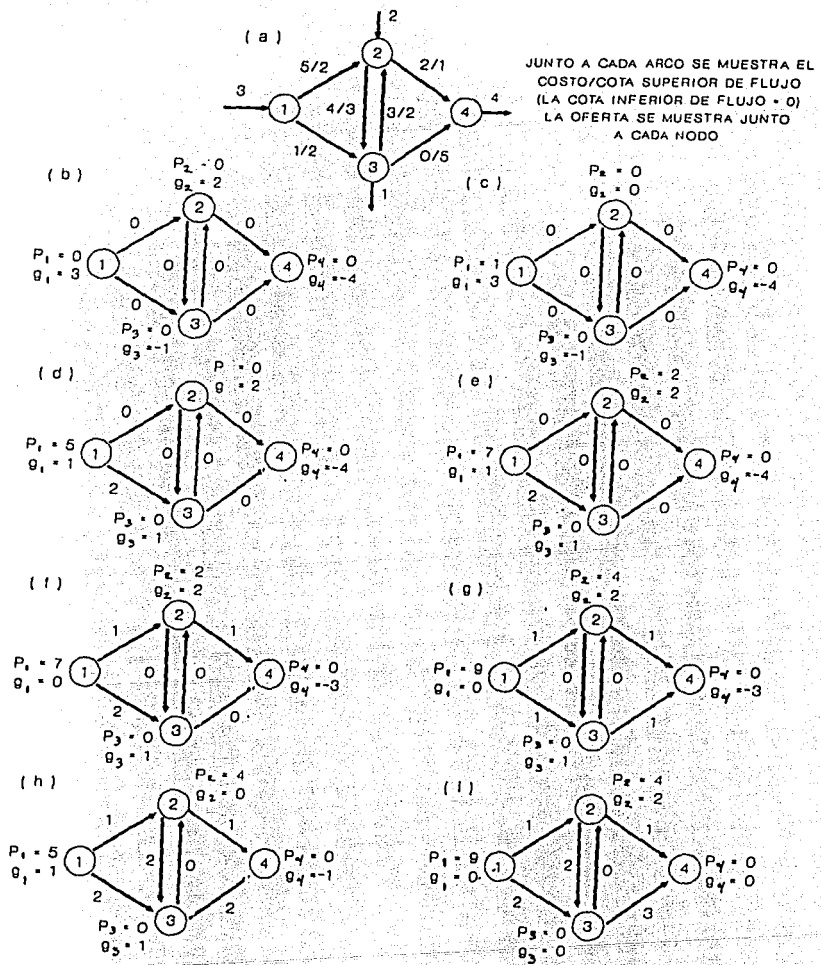


Figura 3.5 Ejemplo del método de relajación, comenzando con precios igual a cero

a) Datos del problema

b) Flujos, precios y holguras iniciales

c) Después de la 1era iteración, que consiste de un cambio de precio en el nodo 1.

d) Después de la segunda iteración, que consiste de otro cambio de precio en el nodo 1. [Note que el flujo del arco (1,3) cambia, vea la ecuación 3.19]

e) Después de la tercera iteración, que consiste de un cambio de precio de los nodos 1 y 2 .

f) Después de la cuarta iteración, que consiste de una aumentación a través de la cadena (1,2,4)

g) Después de la quinta iteración, que consiste de un cambio de precios de los nodos 1 y 2 .

h) Después de la sexta iteración, que consiste de una aumentación a través de la cadena (2,3,4).

i) Después de la séptima iteración, que consiste de una aumentación a través de la cadena (3,4).

Proposición 3.4. Considere el problema de F.C.M. y suponga que a_{ij}, b_{ij}, c_{ij}, s_i son todos enteros. Si el problema es factible, entonces el método de relajación termina con un vector de flujo X óptimo entero y un vector de precios P óptimo entero.

Demostración. La demostración es similar a la prueba correspondiente al método primal-dual. Primero debemos notar que todas las operaciones de la iteración preservan la integralidad del par de vectores flujo-precio. Para ver que las condiciones de H.C. son conservadas también, note que una iteración en la aumentación de flujo cambia solamente flujos de arcos balanceados, por lo que no puede destruir H.C. Aún más, los cambios de flujo de las ecuaciones (3.19) y (3.20) y los cambios de precio de las ecuaciones (3.21) y (3.22) mantienen H.C., porque ellos hacen que los flujos de los arcos balanceados y los cambios en los precios, cambien de activos (o inactivos) a su correspondiente cota superior (o inferior).

Cada vez que hay un cambio de precios en el paso 4, hay un mejoramiento estricto en el costo dual en una cantidad entera $\gamma\, q'(P; d_S)$ {Usando la propiedad de H.C., se pude ver que $\gamma > 0$, y por lo argumentado anteriormente, d_S es una dirección de ascenso, así que $q'(P; d_S) > 0$}. Entonces para un problema factible, no podemos tener un número infinito de cambios en los precios. Por otro lado, es imposible tener un número infinito de aumentos de flujo entre dos cambios sucesivos de precio, dado que cada uno de estos reduce la holgura absoluta total en una cantidad entera. Se sigue entonces que el algoritmo puede ejecutar únicamente un número finito de operaciones y por lo tanto debe terminar. Como al término X es factible y satisface H.C., junto con P, se sigue que X es óptimo primal y P es dual óptimo. □

Si el problema es no factible, el método puede terminar porque $g_i \leq 0$ para todo nodo i y $g_i < 0$ para al menos un nodo i o porque después de los ajustes de flujo de las ecuaciones (3.19) y (3.20) en el paso 4, no hay un arco (i,j) con $x_{ij} < c_{ij}$, $i \in S$, $j \notin S$ o un arco (j,i) con $b_{ji} < x_{ji}$, $i \in S$ y $j \notin S$. Sin embargo, hay también la posibilidad de que el método ejecute un número infinito de iteraciones y cambio de precios, con los precios de algunos nodos incrementandose hasta ∞. Se puede demostrar que, cuando el problema es factible el precio de los nodos permanece por debajo de una cierta cota precalculada en el trancurso del algoritmo. Este hecho puede ser usado como una

prueba adicional para detectar no factibilidad.

Es importante notar que la derivada direccional $q'(P; d_S)$ necesaria para la prueba de ascenso (3.16) en el paso 2 puede ser calculada incrementalmente (cuando nuevos nodos son añadidos uno a uno a S) usando la siguiente ecuación

$$q'(P; d_S) = \sum_{i \in S} g_i - \sum_{(j,i):balanceados, j \notin S, i \in S} (x_{ji} - b_{ji}) - \sum_{(i,j):balanceados, i \in S, j \notin S} (c_{ij} - x_{ij})$$

se sigue de la ecuación que, dado $q'(P; d_S)$ y un nodo $i \notin S$, se pueda calcular la derivada direccional correspondiente al conjunto agrandado $S \cup \{i\}$ usando la fórmula

$$q'(P; d_{S \cup \{i\}}) = q'(P; d_S) + \sum_{\{j | (i,j):balanceado, j \in S\}} (x_{ij} - b_{ij}) + \sum_{\{j | (j,i):balanceado, j \in S\}} (c_{ji} - x_{ji}) -$$
$$- \sum_{\{j | (j,i):balanceado, j \notin S\}} (x_{ji} - b_{ji}) - \sum_{\{j | (i,j):balanceado, j \notin S\}} (c_{ij} - x_{ij})$$

Esta fórmula es conveniente porque involucra únicamente los arcos balanceados incidentes al nuevo nodo i, que debe ser examinado de cualquier forma mientras se ejecuta el paso 2.

En la práctica, el método es implementado usando iteraciones que comienzan de nodos con holguras tanto positivas como negativas. Esto parece mejorar sustancialmente el comportamiento del método. Se pude demostrar que para un problema factible, el algoritmo termina adecuadamente bajo estas circustancias. Otra importante implementación práctica es hecha con la elección inicial adecuada de flujos y precios. Una posibilidad es tratar de escoger un vector inicial de precios que sea tan cercano al óptimo como sea posible; para esto uno puede entonces escoger los flujos en los arcos para que satisfagan las condiciones de H.C.

3.4.1 BUSQUEDA LINEAL E ITERACIONES DE ASCENSO COORDINADO

Supongamos que el cambio de precios del paso 4 ocurre con S igual al nodo inicial i, esto sólo sucede cuando la iteración revisa los arcos incidentes a i en la primera iteración del paso 2 y encuentra que la dirección correspondiente mejora el costo dual $[q'(P; d_{\{i\}}) > 0]$. Si una búsqueda lineal del tamaño óptimo que maximice la función dual a través de $d_{\{i\}}$ es realizada, el precio p_i cambia a un punto de ruptura donde la derivada por la derecha es no positiva y la derivada por la izquierda es no negativa (ver la figura 3.6)

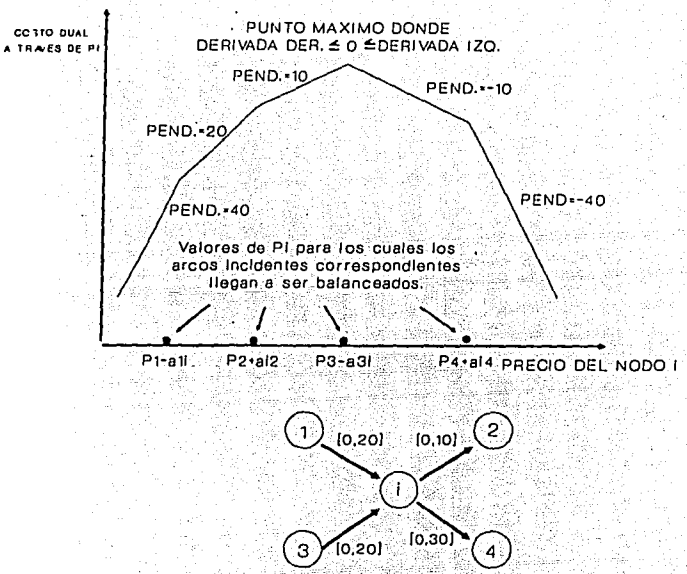


Figura 3.6 Ilustración de una iteración de relajación con un solo nodo. Aquí, el nodo i tiene tiene cuatro arcos incidentes $(1,i)$, $(3,i)$, $(i,2)$, $(i,4)$ con rangos de flujo [0,20], [0,20], [0,10], [0,30], respectivamente y oferta $s_i = 0$. Los costos de los arcos y los precios actuales son tales que

$$p_1 - a_{1i} \leq p_2 + a_{i2} \leq p_3 - a_{3i} \leq p_4 + a_{i4}$$

como se mustra en la figura. Los puntos de ruptura del costo dual a través del precio p_i corresponden a los valores de p_i en los cuales uno o más arcos incidentes al nodo i llegan a ser balanceados. Para valores entre dos puntos de ruptura sucesivos, no hay arcos balanceados. Para cualquier precio p_i a la izquierda del punto máximo, la holgura g_i debe ser positiva para que satisfaga H.C. En una iteración con un solo nodo con búsqueda lineal p_i se incrementa al punto máximo.

Una descripción detallada de esta iteración de relajación con un solo nodo utilizando búsqueda lineal, iniciando con un par (X, P) que satisface H.C. es la siguiente

ITERACION DE RELAJACION CON SOLO UN NODO

Escoger un nodo i con $g_i > 0$. Hacer

$$B_i^+ = \{j \mid (i,j) : balanceado,\ x_{ij} < c_{ij}\} \tag{3.23}$$

$$B_i^- = \{j \mid (j,i) : balanceado,\ b_{ji} < x_{ji}\} \tag{3.24}$$

PASO 1

Si $$g_i \geq \sum_{j \in B_i^+} (c_{ij} - x_{ij}) + \sum_{j \in B_i^-} (x_{ji} - b_{ji})$$

ir al paso 4. De otra manera, si $g_i > 0$, escoger un nodo $j \in B_i^+$ con $g_j < 0$ e ir al paso 2, o escoger un nodo $j \in B_i^-$ con $g_j < 0$ e ir al paso 3; si ninguno de tales nodos puede ser encontrado, o si $g_i = 0$, ir a la siguiente iteración.

PASO 2 AJUSTE DE FLUJO EN LOS ARCOS EXTERNOS

Sea $$\delta = \min \{g_i, -g_j, c_{ij} - x_{ij}\}$$

Hacer

$$x_{ij} = x_{ij} + \delta_i \qquad g_i = g_i - \delta_i \qquad g_j = g_j + \delta$$

y si $x_{ij} = c_{ij}$, eliminar j de B_i^+, ir al paso 1.

PASO 3 AJUSTE DE FLUJO EN LOS ARCOS INTERNOS

Sea $$\delta = \min \{g_i, -g_i, x_{ji} - b_{ji}\}$$

Hacer

$$x_{ji} = x_{ji} - \delta_i \qquad g_i = g_i - \delta_i \qquad g_j = g_j + \delta$$

y si $x_{ji} = b_{ji}$, eliminar j de B_i^+, ir al paso 1.

PASO 4 INCREMENTAR EL PRECIO DE i

Hacer $$g_i = g_i - \sum_{j \in B_i^+} (c_{ij} - x_{ij}) - \sum_{j \in B_i^-} (x_{ji} - b_{ji}) \tag{3.25}$$

$$x_{ij} = c_{ij} \qquad \text{para todo } j \in B_i^+ \tag{3.26}$$

$$x_{ji} = b_{ji} \qquad \text{para todo } j \in B_i^- \tag{3.27}$$

$$p_i = \min \{\{p_j + a_{ij} \mid (i,j) \in A,\ p_i < p_j + a_{ij}\}, \{p_j - a_{ji} \mid (j,i) \in A,\ p_i < p_j - a_{ji}\}\} \tag{3.28}$$

Si después de estos cambios $g_i > 0$, recalcular los conjuntos B_i^+ y B_i^- usando (3.23) y (3.24), e ir al paso 1; de otra forma realizar la siguiente iteración [Nota: Si el conjunto de arcos en que el mínimo de la ecuación (3.28) se calcula es vacío, entonces hay dos posibilidades: (a) $g_i > 0$, en cuyo caso el costo dual se incrementa ilimitadamentea través de p_i y por lo tanto el problema primal no es factible o (b) $g_i = 0$, en cuyo caso el costo permanece constante a través de p_i; en este caso P queda sin cambio y se realiza la siguiente iteración].

Note que la iteración con un solo nodo puede fallar en no cambiar ni X ni P. En este caso, la iteración debe de ser seguida por una iteración regular de relajación que etiquete los nodos adyacentes apropiados del nodo i. La manera eficiente de implementar la iteración de relajación es primero intentar la versión con un solo nodo, si falla se procede con la versión de nodos múltiples, ahorrando cálculos cuando sea posible la utilización de un sólo nodo. Los códigos RELAX [*BeT88b*], [*BeT89*] hacen uso de esta idea.

Una examinación cuidadosa de la iteración de relajación con un solo nodo muestra que en el paso 4, después del cambio de holgura de la ecuación (3.25), la holgura g_i puede ser igual a cero, esto sucederá si $g_i = 0$ y simultáneamente no hay un arco balanceado (i, j) con $x_{ij} < c_{ij}$ o un arco balanceado (j, i) con $b_{ji} < x_{ji}$. En este caso, se puede demostrar (ver la figura 3.6) que el cambio de precios de la ecuación (3.28) deja el costo dual sin cambios, correspondiendo el movimiento de p_i a un segmento horizontal hacia el próximo punto de ruptura del costo dual, como se muestra en la figura 3.7. Esto se conoce como una *iteración degenerada de ascenso*. La experiencia computacional llevada a cabo por Bertsekas muestra que generalmente es preferible permitir tales iteraciones siempre que sea posible. Para tipos especiales de problemas como el de asignación, el uso de iteraciones degeneradas de ascenso puede reducir dramaticamente el tiempo total de computación.

Finalmente, se puede mencionar que las iteraciones de relajación en un solo nodo puede ser usadas como inicialización para el método primal-dual. En particular, se puede iniciar con varios ciclos de iteraciones en un solo nodo, donde cada nodo con holgura no cero se utiliza para la relajación una vez en cada ciclo, el par resultante (X, P) es entonces usado como un par inicial para el método primal-dual y este procedimiento ha demostrado ser muy efectivo.

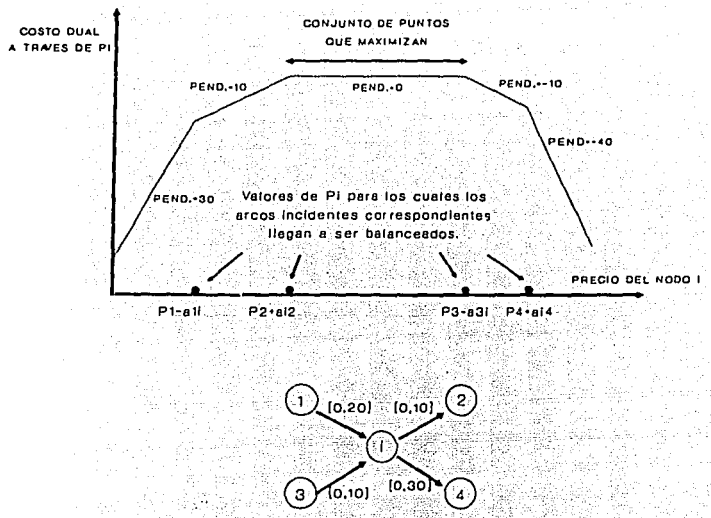


Figura 3.7 Ilustración de un incremento degenerado de precios. La diferencia entre este ejemplo y el de la figura 3.6, es que el rango de flujo factible del arco $(3, i)$ es ahora [0,10] en vez de [0,20]. Aquí, hay un segmento horizontal de la gráfica del costo dual a través de p_i, correspondiente a los puntos que maximizan la función. Un incremento degenerado de precios mueve p_i del extremo izquierdo del segmento maximizando al extremo derecho del mismo.

4 ALGORITMOS TIPO SUBASTA

En este capítulo se tratan los algoritmos tipo subasta para el problema de flujo a costo mínimo. En las primeras secciones se describe el algoritmo subasta principal para el problema de asignación, dando la interpretación de este algoritmo como un descenso coordinado. Estas ideas, junto con la equivalencia entre el problema de asignación y el de FCM, dan origen a la generalización del algoritmo subasta para el problema de FCM y que recibe el nombre de algoritmo genérico de subasta, y que se presenta en la sección 4.4. Por último se presenta también, el método de ϵ-relajación como un caso especial del algoritmo genérico, donde se explotan características especiales para hacerlo más eficiente en la solución del problema de FCM. Se proporciona un análisis de la complejidad de este último algoritmo que muestra la metodología para obtener la complejidad de los demás algoritmos presentados en este trabajo.

4.1 EL ALGORITMO SUBASTA PARA EL PROBLEMA DE ASIGNACION

Vimos que en el problema de asignación (sección 1.4) se desea relacionar n personas y n objetos uno a uno, donde se cuenta con los valores o beneficios a_{ij} de asignar a la persona i con el objeto j y se desea encontrar la asignación que maximice el beneficio total. El conjunto de objetos para el cual la persona i puede ser asignada es un conjunto no vacío denotado $A(i)$. Una *asignación* S es un (posiblemente vacío) conjunto de pares personas-objetos (i,j) tal que $j \in A(i)$ para todo arco $(i,j) \in S$ y tal que para cada persona i, exista a lo más un par $(i,j) \in S$. Dado una asignación S, decimos que la *persona i está asignada*, si existe el par $(i,j) \in S$, de otra manera decimos que i está *no asignada*. Usamos una terminología similar para objetos. Una asignación es *factible* si contiene n parejas, así que toda persona y todo objeto están asignados, de otra manera la asignación es *parcial*.

Vimos en la sección 2.1.3 el algoritmo de subasta inexperto que surge de una manera natural para resolver este problema, con el incoveniente que para ciertos ejemplos se presentan dificultades

en la terminación del algoritmo, mencionamos también que con una ligera variante, el algoritmo trabaja adecuadamente.

Enseguida se describe este último algoritmo llamado algoritmo subasta principal

4.2 EL ALGORITMO SUBASTA PRINCIPAL

Los algoritmos subasta proceden iterativamente y terminan cuando una asignación factible es obtenida. Al inicio de la iteración genérica, se tiene una asignación parcial S y un vector de precios P satisfaciendo $\varepsilon - H.C.$, esto es la condición

$$a_{ij} - p_j \geq \max_{k \in A(i)} \{a_{ik} - p_k\} - \varepsilon \quad \forall (i,j) \in S \tag{4.1}$$

Como elección inicial, se puede utilizar un conjunto de precios arbitrarios junto con la asignación vacía, que trivialmente satisface $\varepsilon - H.C.$ La iteración consiste de dos fases, la *fase de ofertas* y la *fase de asignación* que enseguida se describen. Veremos posteriormente que la iteración preserva las condiciones de $\varepsilon - H.C.$

FASE DE OFERTAS

Sea I un subconjunto no vacío de personas i que no están asignadas bajo S. Para cada persona $i \in I$:

PASO 1

Encontrar el "mejor" objeto j_i, teniendo valor máximo, esto es

$$j_i = \arg \max_{j \in A(i)} \{a_{ij} - p_j\} \text{ y el correspondiente valor}$$

$$v_i = \max_{j \in A(i)} \{a_{ij} - p_j\} \tag{4.2}$$

encontrar el mejor valor ofrecido por los objetos distintos de j_i

$$w_i = \max_{j \in A(i),\ j \neq j_i} \{a_{ij} - p_j\} \tag{4.3}$$

(Si j_i es el único objeto en $A(i)$, definimos $w_i = -\infty$, o para propósitos computacionales, un número mucho menor que v_i).

PASO 2

Calcular la oferta de la persona i que está dada por

$$b_{ij_i} = p_{j_i} + v_i - w_i + \varepsilon = a_{ij_i} - w_i + \varepsilon \tag{4.4}$$

(Caracterizamos esta situación diciendo que la persona i ofrece una cantidad por el objeto j_i, y que el objeto j_i recibe una oferta de la persona i. El algoritmo trabaja si la oferta tiene cualquier valor entre $p_{j_i} + \varepsilon$ y $p_{j_i} + v_i - w_i + \varepsilon$, pero trabaja más rápido con la elección máxima de la ecuación (4.4))

FASE DE ASIGNACION

PASO UNICO

Para cada objeto j, sea $p(j)$ el conjunto de personas de las cuales j recibe una propuesta en la fase de oferta de la iteración. Si $p(j)$ es no vacío, p_j se incrementa a su oferta mayor

$$p_j = \max_{i \in p(i)} b_{ij} \tag{4.5}$$

remover de la asignación S cualquier par (i, j) (Si j fue asignado a algún i bajo S) y agregar a S el par (i_j, j) donde i_j es una persona en $p(j)$ proporcionando el máximo valor.

Note que hay algunas libertades para escoger el subconjunto de personas I que hacen propuestas durante una iteración, una posibilidad es tomar a I como una sola persona no asignada. Esta versión, conocida como Gauss-Seidel por su similitud con el método del mismo nombre para resolver sistemas de ecuaciones no lineales, usualmente ([BeE88b]) trabaja mejor con una implementación de cálculos seriales. La versión donde I consiste de todas las personas no asignadas, es una de las de mejor comportamiento para computación paralela y es conocida como la versión Jacobi, por su similitud con los métodos Jacobi para resolver sistemas de ecuaciones no lineales.

Durante una iteración, los objetos cuyos precios son modificados, son los que recibieron una oferta durante la iteración. Cada cambio de precio involucra un incremento de al menos ε. Para ver esto, note que de las ecuaciones (4.2) a (4.4) tenemos

$$b_{ij_i} = a_{ij_i} - w_i + \varepsilon \geq a_{ij_i} - v_i + \varepsilon = p_{j_i} + \varepsilon$$

así que cada oferta para un objeto, incluyendo la ganadora, supera el precio actual del objeto en al menos ε. Al finalizar la iteración, tenemos una nueva asignación que difiere de la anterior en que cada objeto que recibió una oferta está asignado ahora a alguna persona que estaba sin asignar al inicio de la iteración.

La elección del incremento en la oferta $v_i + w_i + \varepsilon$ para una persona i [Ver ecuación (4.4)] es tal que la condición de $\varepsilon - H.C.$ es preservada por el algoritmo, como lo demuestra la siguiente proposición (en efecto, se puede ver que es el incremento mayor en las ofertas para lo cual esto es así).

Proposición 4.1 El algoritmo subasta preserva $\varepsilon - H.C.$ a través de su ejecución, esto es, si la asignación y el vector de precios disponibles al inicio de la iteración satisfacen $\varepsilon - H.C.$, lo mismo es verdadero para la asignación y el vector de precios obtenido al final de la iteración.

Demostración. Suponga que el objeto j^* recibió una oferta de la persona i y fue asignada a i durante la iteración. Sean p_j y p'_j los precios del objeto antes y después de la fase de asignación, respectivamente. Entonces tenemos (ver ecuaciones (4.4) y (4.5))

$$p'_{j^*} = a_{ij^*} - w_i + \varepsilon$$

usando esta ecuación, obtenemos que

$$a_{ij^*} - p'_{j^*} = w_i - \varepsilon = \max_{j \in A(i),\ j \neq j^*} \{a_{ij} - p_j\} - \varepsilon$$

como $p'_j \geq p_j \ \forall\, j$, esta ecuación implica que

$$a_{ij^*} - p'_{j^*} \geq \max_{j \in A(i)} \{a_{ij} - p'_j\} - \varepsilon \tag{4.6}$$

que demuestra que la condición de $\varepsilon - H.C.$ (4.1) se sigue satisfaciendo después de la fase de asignación de una iteración para todos los pares (i, j^*) que entraron a la asignación durante la iteración.

Considere también cualquier par (i, j^*) que pertenece a la asignación justo antes de la iteración y que también pertenece a ella después de la iteración. Entonces j^* no debió recibir una oferta durante la iteración, así que $p'_{j^*} = p_{j^*}$. Por lo tanto, la ecuación (4.6) es válida en vista de la condición de $\varepsilon - H.C.$ que daba prioridad a la iteración y el hecho de $p'_j \geq p_j$ para todo j. Así, la condición de

$\varepsilon - H.C.$ (4.1), se presenta para todas las parejas (i, j^*) que pertenecen a la asignación después de la iteración. □

El siguiente resultado establece la validez del algoritmo. La prueba se basa en los siguientes hechos

1. Cuando un objeto es asignado, permanece así a través de todo el algoritmo. Aún más, excepto al final, siempre existe al menos un objeto que nunca ha sido asignado y que tiene un precio igual a su precio inicial. La razón es que una fase de ofertas y asignación puede reasignar algunos objetos asignados, pero no puede resultar que el objeto llegue a estar sin asignar.

2. Cada vez que un objeto recibe una oferta, su precio se incrementa en al menos ε. [Ver las ecuaciones (4.4) y (4.5)]. Así, si el objeto recibe un número infinito de veces su precio se incrementa a ∞.

3. Cada $| A(i) |$ ofertas de la persona i, donde $| A(i) |$ es el número de objetos en el conjunto $A(i)$, el escalar v_i definido por la ecuación

$$v_i = \max_{j \in A(i)} \{a_{ij} - p_j\} \tag{4.7}$$

decrese en al menos ε. La razón es que una oferta de la persona i, decrementa v_i en al menos ε, o deja v_i sin cambio porque hay más de un objeto j que produce el máximo en la ecuación (4.7). Sin embargo, en el último caso, el precio del objeto j_i recibiendo la oferta incrementará su precio en al menos ε, y el objeto j_i no recibirá otra oferta por la persona i hasta que v_i decrezca al menos ε. La conclusión es que si una persona i hace ofertas un número infinito de veces, v_i debe decreser a $-\infty$.

Proposición 4.2 Si existe al menos una asignación factible, el algoritmo subasta principal termina con una asignación factible que está a $n\varepsilon$ de ser óptima (y es óptima si los datos del problema son enteros y $\varepsilon < 1/n$).

Demostración. La haremos por contradicción.

Si la terminación no ocurre, el subconjunto I^∞ de objetos que recibieron un número infinito de ofertas es no vacío. También, el subconjunto J^∞ que hicieron un número infinito de ofertas es no vacío. Por lo señalado en 2), los precios de todos los objetos en J^∞ deben tender a ∞, mientras

que por lo señalado en 3), los escalares $v_i = \max_{j \in A(i)} \{a_{ij} - p_j\}$ deben decreser a $-\infty$ para todas las personas $i \in I^\infty$, esto implica que

$$A(i) \subset J^\infty \quad \forall\, i \in I^\infty \tag{4.8}$$

La condición de $\varepsilon - H.C.$ (4.1) dice que $a_{ij} - p_j \geq v_i - \varepsilon$ para toda pareja (i, j) asignada, así después de un número finito de iteraciones, cada objeto en J^∞ puede ser únicamente asignado a una persona de I^∞. Como después de un número finito de iteraciones, al menos una persona de I^∞ será no asignada al inicio de cada iteración, se sigue que el número de personas en I^∞ es estrictamente mayor que el número de objetos en J^∞. Esto contradice la existencia de una asignación factible dada la ecuación (4.8), en la que personas en I^∞ pueden ser asignados únicamente a objetos en J^∞. Por lo tanto, el algoritmo debe terminar. La asignación factible obtenida al termino, satisface $\varepsilon - H.C.$ por la proposición 4.1 , por lo que la asignación esta dentro de $n\varepsilon$ de ser óptima. □

Considere ahora el caso de un problema de asignación no factible. En este caso, el algoritmo subasta puede probablemente no terminar, al continuar incrementando los precios de algunos objetos en al menos ε. Aún más, algunas personas i estarán realizando ofertas indefinidamente y los valores máximos correspondientes

$$v_i = \max_{j \in A(i)} \{a_{ij} - p_j\}$$

estarán decrementandose hasta $-\infty$. Uno puede detectar esta situación haciendo uso de una cota inferior precalculada para los valores de v_i, que se toman cuando el problema es factible. Una vez que v_i toma valores debajo de esta cota para algún i, nosotros sabemos que el problema es no factible. Desafortunadamente, pueden pasar muchas iteraciones para que algún v_i alcance esta cota. Un método alternativo para detectar no factibilidad es convertir el problema en un problema factible agregando arcos artificiales a la gráfica de asignación. Los valores de estos arcos deben ser pequeños (es decir, números negativos muy grandes), para que así, estos arcos nunca participen en una asignación óptima a menos de que el problema sea no factible. El valor de umbral apropiado para los arcos artificiales puede ser cuantificado.

4.3 INTERPRETACION DEL ALGORITMO DE SUBASTA COMO UN DESCENSO COORDINADO APROXIMADO

La versión de Gauss-Seidel del algoritmo de subasta es similar a los algoritmos de descenso coordinado y al método de relajación en particular, porque involucra el cambio de precio de un sólo objeto con todos los demás precios fijos, sin embargo en constraste con el método de relajación, tal cambio de precio puede empeorar estrictamente el valor de la función dual

$$Q(P) = \sum_{i=1}^{n} \max_{j \in A(i)} \{a_{ij} - p_j\} + \sum_{j=1}^{n} p_j$$

que fue introducida anteriormente.

Generalmente podemos interpretar las fases de oferta y asignación como un paso aproximado del descenso coordinado simultáneo para todos los precios coordinados que se incrementan durante la iteración. Los pasos coordinados son dirigidos a la minimización aproximada de la función dual. En particular, tenemos la siguiente proposición, que se ilustra en la figura 4.1

Proposición 4.3 El precio p_j de cada objeto j que recibe una oferta durante la fase de asignación se incrementa en un valor que minimiza $Q(P)$ cuando los demás precios se mantienen constantes o a lo más excede tal en no más de ε.

Demostración. De la definición de costo dual Q, la derivada direccional por la derecha de Q en la dirección p_j es

$$d_j^+ = 1 - (\text{número de personas } i \text{ con } j \in A(i) \text{ y } p_j < y_{ij})$$

donde

$$y_{ij} = a_{ij} - \max_{k \in A(i),\, k \neq j} \{a_{ik} - p_k\}$$

es el nivel de p_j debajo del cual j es el mejor objeto para la persona i. Los puntos de ruptura son y_{ij} para todo i tal que $j \in A(i)$. Sea $\bar{y} = \max_{\{i | j \in A(i)\}} \{a_{ij} - p_j\}$, $\bar{i}$ una persona tal que $\bar{y} = y_{ij}$, $\hat{y} = \max_{\{i | j \in A(i), i \neq \bar{i}\}} \{a_{ij} - p_j\}$, e $\hat{i}$ una persona tal que $\hat{i} \neq \bar{i}$ y $\hat{y} = y_{ij}$. Note que el intervalo $[\hat{y}, \bar{y}]$ es el conjunto de puntos que minimizan Q en la dirección del precio coordinado p_j. Sea p_j el precio de j justo antes de la iteración en la cual j recibe una oferta y sea p_j' el precio de j después de la

iteración. Afirmamos que $\hat{y} \leq p_j' \leq \bar{y} + \varepsilon$. Es decir, si i es la persona que propone una oferta y gana j durante la iteración, entonces $p_j' = v_{ij} + \varepsilon$, implicando que $p_j' \leq \bar{y} + \varepsilon$.

Para probar que $p_j' \geq \hat{y}$, notemos que si $p_j \geq \hat{y}$, también debemos tener $p_j' \geq \hat{y}$ dado que $p_j' \geq p_j$. Por otra parte, si $p_j' < \hat{y}$ hay dos posibilidades.

1) Al inicio de la iteración, $\bar{i}$ no estaba asignado a j, en este caso $\bar{i}$ no estaba asignado a ningún objeto, por lo que i ofrecio por j cierta cantidad, de tal manera que $p_j' = \bar{y} + \varepsilon$, o $\bar{i}$ estaba asignado a un objeto $\bar{j} \neq j$, en cuyo caso por las condiciones de $\varepsilon - H.C.$

$$a_{\bar{i}j} - p_j - \varepsilon \leq a_{\bar{i}\bar{j}} - p_{\bar{j}} \leq \max_{k \in A(\bar{i}),\, k \neq j} \{a_{\bar{i}k} - p_k\} = a_{\bar{i}j} - \bar{y}.$$

Así, $p_j \geq \bar{y} - \varepsilon$, implicando que $p_j' \geq \bar{y}$ (Dado que una oferta incrementa un precio en al menos ε) En ambos casos se tiene $p_j' \geq \bar{y} \geq \hat{y}$.

2) Al inicio de la iteración, $\bar{i}$ estaba asignado a j. En este caso, $\bar{i}$ no estaba asignada a j, repitiendo el argumento del parrafo anterior, con $\bar{i}$ y $\hat{y}$ reemplazando a $\bar{i}$ y $\bar{y}$, respectivamente, obtenemos que $p_j' \geq \hat{y}$. □

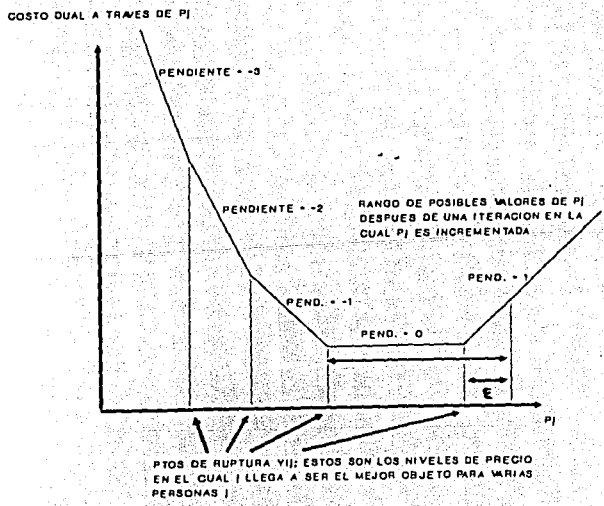


Figura 4.1 Forma del costo dual a través del precio coordinado P_j en la interpretación de las fases de oferta y asignación como un descenso coordinado.

4.4 ALGORITMO GENERICO DE SUBASTA PARA EL PROBLEMA DE FLUJO A COSTO MINIMO

En esta sección se generaliza la idea de los algoritmos subasta y se aplica al problema de flujo a costo mínimo

$$\min_{s.a.} \sum_{(i,j)\in A} a_{ij}x_{ij}$$

$$\sum_{\{j|(i,j)\in A\}} x_{ij} - \sum_{\{j|(j,i)\in A\}} x_{ji} = s_i \qquad \forall \quad i \in N \tag{4.9}$$

$$b_{ij} \leq x_{ij} \leq c_{ij} \qquad \forall\,(i,j) \in A \tag{4.10}$$

donde a_{ij}, b_{ij}, c_{ij} y s_i son enteros. Para un vector de flujo dado, la holgura de cada nodo i es denotado por.

$$g_i = \sum_{\{j|(j,i)\in A\}} x_{ji} - \sum_{\{j|(i,j)\in A\}} x_{ij} + s_i$$

El algoritmo mantiene todo el tiempo un vector de flujo de capacidad factible X y un vector de precios P que satisfacen la condición de $\varepsilon - H.C.$ Vea la figura 4.2

$$p_i - p_j \leq a_{ij} + \varepsilon \;\; \forall\,(i,j) \in A \text{ con } x_{ij} < c_{ij} \tag{4.11a}$$

$$p_i - p_j \geq a_{ij} - \varepsilon \;\; \forall\,(i,j) \in A \text{ con } b_{ij} < x_{ij} \tag{4.11b}$$

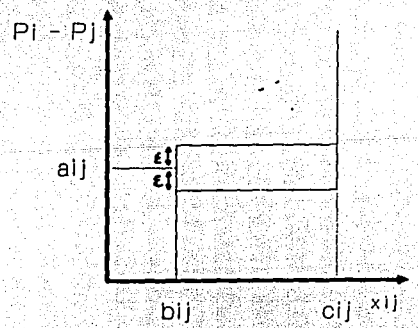


Figura 4.2 Ilustración de las condiciones de ε-Holguras Complementarias.

La utilidad de la condición de $\varepsilon - H.C.$ se debe en gran medida a la siguiente proposición:

Proposición 4.4 Si $\varepsilon < 1/N$, con N el número de nodos, X es factible y junto con P satisfacen $\varepsilon - H.C.$, entonces X es óptimo para el problema de FCM.

Demostración. Si X no es óptimo, entonces existe un ciclo simple que tiene un costo negativo

$$\sum_{(i,j)\in Y+} a_{ij} - \sum_{(i,j)\in Y-} a_{ij} < 0 \tag{4.12}$$

y que es no bloqueado con respecto a X, es decir

$$x_{ij} < c_{ij} \quad \forall\,(i,j) \in Y+$$

$$b_{ij} < x_{ij} \quad \forall\,(i,j) \in Y-$$

Por la condición de $\varepsilon - H.C.$, la relación precedente implica qe

$$p_i \leq p_j + a_{ij} + \varepsilon \;\; \forall\,(i,j) \in Y+$$

$$p_j \leq p_i - a_{ij} + \varepsilon \;\; \forall\,(i,j) \in Y-$$

Sumando estas relaciones sobre todos los arcos de Y (cuyo número no es más de N). Y usando la hipótesis $\varepsilon < 1/N$ obtenemos

$$\sum_{(i,j)\in Y+} a_{ij} - \sum_{(i,j)\in Y-} a_{ij} \geq -N\varepsilon > -1$$

pero como el costo de los arcos a_{ij} es entero, obtenemos una contradicción de la ecuación (4.12)

4.4.1 ALGUNAS OPERACIONES ALGORITMICAS BASICAS

Enseguida se definen algunos conceptos y operaciones computacionales que pueden ser usadas para construir bloques en los dos algoritmos siguientes. Cada una de estas definiciones supone que (X, P) es un par de vectores flujo-precios satisfaciendo $\varepsilon - H.C.$ y serán usadas únicamente en ese contexto

Un arco (i,j) es llamado ε^+ — *no bloqueado* si

$$p_i = p_j + a_{ij} + \varepsilon \text{ y } x_{ij} < c_{ij} \tag{4.13}$$

Un arco (j,i) es llamado ε^- — *no bloqueado* si

$$p_i = p_j - a_{ji} + \varepsilon \text{ y } b_{ji} < x_{ji} \tag{4.14}$$

La lista *incremental* de un nodo i es el (posiblemente vacío) conjunto de arcos (i,j) con vértice inicial i y que son ε^+-no bloqueados y arcos (j,i) con vértice final i que son ε^--no bloqueados.

En los algoritmos siguientes, el flujo se puede incrementar solamente a traves de arcos ε^+-no bloqueados y solamente se puede decrementar a través de arcos ε^--no bloqueados.

Las siguientes dos definiciones especifican el tipo de cambio en flujos considerados.

Para un arco (i,j) [un arco (j,i)] de la lista incremental del nodo i, sea δ un escalar tal que $0 < \delta \leq c_{ij} - x_{ij}$ [$0 < \delta \leq x_{ji} - b_{ji}$, respectivamente]. Un *$\delta$ – incremento en el nodo i por el arco (i,j)* [(j,i), respectivamente], consiste en incrementar el flujo x_{ij} en δ (decrementar el flujo x_{ji} en δ, respectivamente), dejando a los demás flujos y al vector de precios, sin cambio.

En el contexto del algoritmo subasta un δ-incremento (con $\delta = 1$) corresponde a asignar a una persona que no lo estaba a un objeto; esto resulta en un incremento del flujo en el arco correspondiente de 0 a 1. La próxima operación consiste en el incremento de precios de un subconjunto de nodos en el máximo común incremento γ que no viole ε-H.C.

Un *incremento de precio* de un conjunto de nodos I no vacío, que es un subconjunto estricto de N (es decir $I \neq \emptyset, I \neq N$) consiste en dejar el vector de flujo X y los precios de los nodos no pertenecientes a I sin cambios e incrementar los precios de los nodos I en la cantidad γ dada por

$$\gamma = \begin{cases} \min\{S^+, S^-\} & \text{si } S^+ \cup S^- \neq \emptyset \\ 0 & \text{si } S^+ \cup S^- = \emptyset \end{cases} \tag{4.15}$$

donde S^+ y S^- son los subconjuntos de escalares dados por

$$S^+ = \{p_j + a_{ij} + \varepsilon - p_i \mid (i,j) \in A \text{ tal que } i \in I,\ j \notin I,\ x_{ij} < c_{ij}\} \tag{4.16}$$

$$S^- = \{p_j - a_{ji} + \varepsilon - p_i \mid (j,i) \in A \text{ tal que } i \in I,\ j \notin I,\ b_{ji} < x_{ji}\} \tag{4.17}$$

En el caso donde el subconjunto I consiste de un solo nodo i, un incremento de precio del conjunto $\{i\}$ también es referido como un incremento de precio del nodo i. Si el incremento del precio γ de la ecuación (4.15) es positivo, se llamará *substantivo*, si $\gamma = 0$ el incremento del precio será *trivial* (Un incremento de γ trivial, que no cambia nada, es introducido para facilitar el enunciado de los dos algoritmos que se darán).

Note que todo escalar en los conjuntos S^+ y S^- de las ecuaciones (4.16) y (4.17) es no negativo por las condiciones de ε-H.C., (4.11a) y (4.11b) respectivamente, así tenemos $\gamma \geq 0$ por lo que el precio se incrementa.

El algoritmo genérico a ser descrito brevemente, consiste en una secuencia de δ-incrementos y operaciones de incrementos de precios. La siguiente proposición enumera algunas propiedades de estas operaciones, que son importantes en el contexto de este algoritmo.

Proposición 4.5 Sean (X, P) un par de vectores satisfaciendo ε-H.C.

a) El par de vectores flujo-precio obtenido después de un δ-incremento o de un incremento en precios satisface ε-H.C.

b) Sea I un subconjunto de nodos tal que $\sum_{i \in I} g_i > 0$. Entonces si el conjunto de escalares S^+ y S^- de las ecuaciones (4.16) y (4.17) son vacíos, el problema no es factible.

Demostración.

a) Por definición de ε-H.C., el flujo de un arco ε^+-no bloqueado y de un ε^--no bloqueado, puede tener cualquier valor dentro del rango de flujo factible. Como un δ-incremento únicamente cambia el flujo de un arco ε^+-no bloqueado o un ε^--no bloqueado, no puede resultar que se viole ε-H.C. Sean P y P' el vector de precios antes y después de un incremento de precios de un conjunto I, respectivamente. Para los arcos (i,j) con $i \in I$ y $j \in I$, o con $i \notin I$ y $j \notin I$, la condición de ε-H.C. (ecuación 4.11) es satisfecha con (X, P'), como también es satisfecha por (X, P) tenemos $p_i - p_j = p'_i - p'_j$. Para los arcos (i,j) con $i \in I$, $j \notin I$ y $x_{ij} < c_{ij}$ tenemos, usando las ecuaciones (4.15) y (4.16)

$$p'_i - p'_j = p_i - p_j + \gamma \leq p_i - p_j + (p_j + a_{ij} + \varepsilon - p_i) = a_{ij} + \varepsilon \tag{4.18}$$

así que la condición de ε-H.C. (4.11a) es satisfecha. Para arcos (j,i) con $i \in I$, $j \notin I$ y $x_{ji} > b_{ji}$ la condición de ε-H.C. (4.11b) se satisface de manera similar

b) Como $S^+ \cup S^-$ es vacío

$$x_{ij} = c_{ij} \quad \forall\, (i,j) \in A \text{ con } i \in I,\ j \notin I \tag{4.19}$$

$$x_{ji} = b_{ji} \quad \forall\, (i,j) \in A \text{ con } i \in I,\ j \notin I \tag{4.20}$$

Tenemos

$$0 < \sum_{i \in I} g_i = \sum_{i \in I} s_i - \sum_{\{(i,j) \in A | i \in I, j \notin I\}} x_{ij} + \sum_{\{(j,i) \in A | i \in I, j \notin I\}} x_{ji} \tag{4.21}$$

combinando las ecuaciones (4.19)-(4.21) se sigue que

$$0 < \sum_{i \in I} s_i - \sum_{\{(i,j) \in A | i \in I, j \notin I\}} c_{ij} + \sum_{\{(j,i) \in A | i \in I, j \notin I\}} b_{ji}$$

para cualquier vector factible, esta relación implica que la suma de las divergencias de los nodos en I excede la capacidad de la cortadura $[I, N - I]$, lo que es una contradicción, por lo que el problema no es factible.□

4.4.2 EL ALGORITMO GENERICO

Suponga que el problema de flujo a costo mínimo es factible y considere un par (X, P) satisfaciendo ϵ-H.C. Suponga además que para algún nodo i tenemos $g_i > 0$

Existen dos posibilidades

1. La lista incremental de i es no vacía, en cuyo caso un δ-incremento en el nodo i es posible.

2. La lista incremental de i es vacía, en cuyo caso el conjunto $S^+ \cup S^-$ correspondiente al conjunto $I = \{i\}$ es no vacío, dado que el problema es factible (ver la proposición 4.5b)). Aún más, de las ecuaciones (4.15)-(4.17) un incremento de precio del nodo i será substantivo.

Así, si $g_i > 0$ para algún i y el problema es factible, entonces un δ-incremento o un aumento substantivo de precios es posible en el nodo i.

Estas observaciones motivan un método, llamado *algoritmo genérico*, que inicia con un par (X, P) que satisfacen ϵ-H.C. y realizan una secuencia de δ-incrementos y aumentos substantivos de precios. El algoritmo mantiene ϵ en un valor positivo fijo y termina cuando $g_i \leq 0$ para todos los nodos i.

ITERACION TIPICA DEL ALGORITMO GENERICO

PASO UNICO

Realizar en secuencia y en cualquier orden un número finito de δ-incrementos y aumentos substantivos de precios; en este momento debe haber al menos un δ-incremento, pero no necesariamente al menos un aumento de precios. Cada δ-incremento debe de realizarse en algún nodo i con $g_i > 0$ y un incremento de flujo δ debe satisfacer $\delta \leq g_i$. Aún más, cada aumento de precio debe realizarse en un conjunto I con $g_i \geq 0$ para todo $i \in I$.

La siguiente proposición establece la validez del algoritmo genérico

Proposición 4.6 Suponga que el problema de flujo a costo mínimo es factible. Si el incremento δ de cada δ-incremento es entero, entonces el algoritmo genérico termina con un par (X, P) que satisface ε-H.C. El vector de flujo X es factible y es óptimo si $\varepsilon < 1/N$.

Demostración. Haremos las siguientes observaciones:

a) El algoritmo preserva ε-H.C., esto es una consecuencia de la prop. 4.5

b) Los precios de todos los nodos son monótonamente no decrecientes durante el algoritmo.

c) Una vez que un nodo tiene holgura no negativa, su holgura permanece así. La razón es que un δ-incremento en el nodo i no puede llevar la holgura de i abajo de cero (dado que $\delta \leq g_i$) y no puede decreser la holgura de los nodos vecinos.

d) Si en algún momento un nodo tiene holgura negativa, su precio nunca debió haber sido incrementado hasta ese momento, y debe ser igual a su precio inicial. Esto es una consecuencia de (c) y la suposición de que solamente nodos con holgura no negativa pueden participar en un aumento de precio.

Supongamos, para llegar a una contradicción, que el algoritmo no termina, entonces, como existe al menos un δ-incremento por iteración, un número finito de δ-incrementos deben ser realizados en algún nodo i con algún arco (i,j). Dado que para cada δ-incremento, δ es entero, un número infinito de δ-incrementos deben ser realizados en el nodo j con el arco (i,j). Esto significa que el arco (i,j) llega a ser alternadamente ε^+-no bloqueado con $g_i > 0$ y ε^--no bloqueado con $g_j > 0$ un número infinito de veces, lo que implica que p_i y p_j deben incrementarse en al menos 2ε un número infinito

de veces. Así, tenemos que $p_i \to \infty$ y $p_j \to \infty$ mientras que $g_i > 0$ o $g_j > 0$, al inicio de un número infinito de δ-incrementos.

Sea N^∞ el conjunto de nodos cuyos precios se incrementan a ∞. Para preservar ε-H.C., debemos tener, después de un número suficiente de iteraciones

$$x_{ij} = c_{ij} \quad \forall\, (i,j) \in A \text{ con } i \in N^\infty,\ j \notin N^\infty \tag{4.22}$$

$$x_{ji} = b_{ji} \quad \forall\, (j,i) \in A \text{ con } i \in N^\infty,\ j \notin N^\infty \tag{4.23}$$

Después de alguna iteración, por (d), todo nodo en N^∞ debe tener holgura no negativa, así que la suma de las holguras de los nodos en N^∞ debe de ser positiva al inicio de los δ-incrementos donde $g_i > 0$ o $g_j > 0$. Se sigue, usando el argumento de la prueba de la proposición 4.6(b) (Ver ecuaciones (4.19)-(4.22)) que

$$0 < \sum_{i \in N^\infty} s_i - \sum_{\{(i,j) \in A \mid i \in N^\infty, j \notin N^\infty\}} c_{ij} + \sum_{\{(j,i) \in A \mid i \in N^\infty, j \notin N^\infty\}} b_{ji}$$

Para cualquier vector factible, esta relación implica que la suma de las divergencias de los nodos en N^∞ exceden la capacidad de la cortadura $[N^\infty, N - N^\infty]$, lo cual es imposible. Se sigue que no hay un vector de flujo factible, contradiciendo la hipótesis. Por lo tanto, el algoritmo debe terminar. Dado que al terminar, tenemos $g_i \leq 0$ para todo i y el problema se supuso factible, se sigue que $g_i = 0$ para todo nodo i. Así, el vector final de flujo X es factible y por (a) satisface ε-H.C. junto con el vector P final. Por la proposición 4.5 si $\varepsilon < 1/N$, X es óptimo.□

La figura 4.3 muestra como el algoritmo genérico puede no terminar nunca, aún para un problema factible, si no es requerido que se realice al menos un δ-incremento por iteración

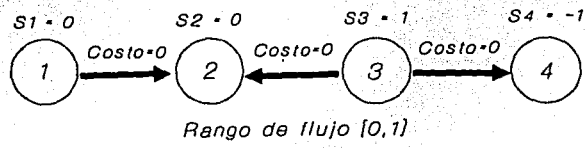


Figura 4.3 Ejemplo de como el algoritmo genérico puede no terminar para un problema factible.

Consideremos que pasa cuando el problema es no factible. Supongamos que el algoritmo genérico es utilizado, así que para cada δ-incremento, δ es entero. Entonces el algoritmo termina con $g_i \leq 0$ para todo i y $g_i < 0$ para al menos un nodo i, en cuyo caso la no factibilidad será detectada. O se realizarán un número infinito de iteraciones y, consecuentemente, un número infinito de δ-incrementos. en el último caso, de la prueba de la proposición 4.6 puede verse que los precios de los nodos involucrados en un número infinito de δ-incrementos diverge a infinito. Esto, junto a una cota en el cambio total de un precio del nodo, puede ser usado para detectar no factibilidad. También puede ser posible detectar no factibilidad descubriendo en el trascurso del algoritmo un subconjunto de nodos I tal que $\sum_{i \in I} g_i > 0$, y que el conjunto de escalares S^+ y S^- de las ecuaciones (4.16) y (4.17) sean vacíos (Ver la proposición 4.5(b)). Esto no es garantía, sin embargo, que tal conjunto sea encontrado durante la ejecución del algoritmo.

El algoritmo genérico puede aplicarse de diferentes maneras a una variedad de problemas con una estructura especial, dando lugar a una variedad de algoritmos específicos. En particular, toma como caso especial el algoritmo de subasta principal para el problema de asignación simétrico.

4.4.3 EL METODO DE ϵ-RELAJACION

El método de ϵ-relajación es un caso especial del algoritmo genérico de la sección previa, donde en cada iteración todos los δ-incrementos y aumentos de precios involucran sólo un nodo. El método de ϵ-relajación puede ser visto también como matemáticamente equivalentemente al algoritmo subasta para el problema de asignación de la sección. Además, el algoritmo subasta puede obtenerse como un caso especial de ϵ-relajación. Recíprocamente, se puede convertir el problema de flujo a costo mínimo a un problema de transporte (ver ejemplo en el primer capítulo) y convertir entonces el último problema a uno de asignación (creando lo suficientes nodos duplicados de las personas y objetos). Se puede verificar que cuando el algoritmo subasta es aplicado a este problema de asignación, y los cálculos son eficientes, se obtiene el método de ϵ-relajación, si suponemos que el problema es factible. En la practica, el método puede ser mejorado agregando mecanismos para detectar infactibilidad, como se discutió en la sección anterior.

Para esto, usamos un valor positivo fijo de ε y comenzamos con un par (X, P) que satisfacen ε-H.C. Aún más, el flujo inicial en los arcos es entero y se demostrará que la integralidad del flujo en los arcos se preserva gracias a la integralidad de la oferta de los nodos y de las capacidades de los arcos (Implementaciones que tiene una mejor complejidad en el peor caso, también requieren que todo flujo inicial en los arcos esté en su cota superior o inferior, ver [*Bet*89], esto puede hacerse fácilmente, a pesar de que no parece ser muy importante en la práctica).

Al inicio de una iteración típica, tenemos un par de vectores (X, P) satisfaciendo ε-H.C., se seleciona un nodo i con $g_i > 0$, si tal nodo no puede ser encontrado, el algoritmo termina. Durante la iteración se realizan varios δ-incrementos y operaciones de aumento del precio del tipo descrito en la sección anterior, involucrando al nodo i.

ITERACION TIPICA DEL METODO DE ε-RELAJACION

PASO 1

Si la lista incremental del nodo i es vacía ir al paso 3, de otra forma seleccionar un arco a de la lista de i e ir al paso 2.

PASO 2

Sea j el nodo final del arco a, que es adyacente a i. Sea

$$\delta = \begin{cases} \min\{g_i, c_{ij} - x_{ij}\} & \text{si } a = (i,j) \\ \min\{g_i, x_{ji} - b_{ji}\} & \text{si } a = (j,i) \end{cases} \tag{4.24}$$

realizar un δ-incremento de i para el arco a. Si como resultado de esta operación, se obtiene $g_i = 0$, ir al paso 3, de otra forma ir al paso 1.

PASO 3

Realizar un incremento de precio en el nodo i. Si $g_i = 0$, realizar la siguiente iteración, si no ir al paso 1.

Alguna información de la iteración en la ϵ-relajación puede ser obtenida notando que en el límite $\epsilon \to 0$, la iteración de relajación de la sección se lleva a cabo en un solo nodo. La figura 4.4 ilustra la secuencia de incremento en precios en una iteración de ϵ-relajación.

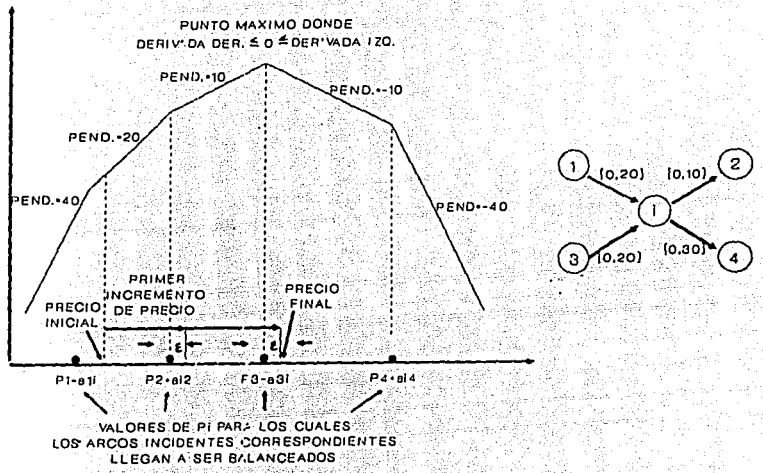


Figura 4.4 Ilustración de incrementos en precios de una iteración de ϵ-relajación. El nodo i tiene cuatro arcos incidentes $(1,i)$, $(3,i)$, $(i,2)$ e $(i,4)$ con rangos de flujo $[0,20]$, $[0,20]$, $[0,10]$, $[0,30]$, respectivamente y oferta $s_i = 0$. Los costos de los arcos y los precios actuales son tales que

$$p_1 - a_{1i} \leq p_2 + a_{i2} \leq p_3 - a_{3i} \leq p_4 + a_{i4}$$

como se muestra en la figura. Los puntos de ruptura del costo dual a través del precio p_i corresponden a los valores de p_i en los cuales uno o más arcos incidentes al nodo i llegan a ser balanceados. Para valores entre dos puntos de ruptura sucesivos, no hay arcos balanceados. Cada aumento de precios de la iteración de ϵ-relajación incrementa p_i al punto que está ϵ a la derecha del siguiente punto de ruptura mayor que p_i (suponiendo que el precio inicial del nodo i está a la izquierda del punto que maximiza en no más de ϵ). En el ejemplo de la figura, hay 2 aumentos de precios, el segundo de los cuales coloca p_i en el punto que está ϵ a la derecha del punto máximo, dando (dentro de ϵ) la interpretación aproximada de ascenso coordinado.

Enseguida demostramos la validez del método de ϵ-relajación usando el análisis de la sección precedente. En particular, decimos que la iteración consiste de un número finito (pero positivo) de δ-incrementos con δ entero y un número finito (posiblemente cero) de aumento en los nodos con holgura no negativa. Además, como los flujos iniciales, las ofertas en los nodos y las cotas en el flujo de los arcos son cantidades enteras, el incremento de flujo en todos los δ-incrementos serán enteros positivos en el transcurso del algoritmo. Aún más, de la ecuación (4.24) se ve que la condición $\delta \leq g_i$, perteneciente al algoritmo genérico se satisface. También notamos que a lo más, un δ-incremento por arco incidente al nodo i es realizado, a causa de la ecuación (4.24) donde se observa que un δ-incremento en un arco a en el paso 2 coloca el flujo del arco en su cota correspondiente, lo que provoca que el arco salga de la lista incremental de i al final de la iteración o resulta que $g_i = 0$, lo que lleva al algoritmo al paso 3 y subsecuentemente a detenerse. Sin embargo, el número de δ-incrementos por iteración es finito. Además tenemos que $g_i > 0$ al inicio y $g_i = 0$ al final de la iteración, así que al menos un δ-incremento debe ocurrir antes de que una iteración pueda detenerse.

En cuanto a los aumentos de precios, se ve que el paso 3 puede ser alcanzado bajo 2 condiciones

1. La lista incremental de i es vacía y $g_i > 0$, en cuyo caso el aumento de precio en el paso 3 será sustantivo (en vista del supuesto de que el problema es factible y de la proposición 4.5 (b))

2. $g_i = 0$, en cuyo caso la iteración se detendrá después de un aumento de precio (posiblemente trivial) en el paso 3.

Así, todos los aumentos de precios envuelven un nodo con holgura no negativa como se requiere en el algoritmo genérico. Como después de cada aumento sustantivo de precio con $g_i > 0$ en al menos un nodo un δ-incremento debe ser realizado, se sigue que el número de incrementos sustantivos de precios es finito.

De estas observaciones, vemos que si el problema es factible, el método de ϵ-relajación es un caso especial del algoritmo genérico y satisface los supuestos de la proposición 4.6, por lo tanto debe terminar con un vector de flujo factible, que es óptimo si $\epsilon < 1/N$.

4.4.4 COMPLEJIDAD DEL ALGORITMO DE ε-RELAJACION

Una parte importante de cualquier algoritmo es el análisis del tiempo que tarda en ejecutarse, por lo que en esta sección daremos un análisis del algoritmo de ε-relajación.

La cota en el precio $\beta(P)$ es una función del vector de precios P y sirve para limitar la cantidad de los incrementos en el precio. Para cualquier trayectoria H, sean $s(H)$ y $t(H)$ sus nodos inicial y final, respectivamente. Para cualquier vector de precios P y una trayectoria simple H, se define

$$d_H(P) = \max\{0, \sum_{(i,j)\in H^+}(p_i - p_j - a_{ij}) - \sum_{(i,j)\in H^-}(p_i - p_j - a_{ij})\}$$

$$= \max\{0, Ps(H) - Pt(H) - \sum_{(i,j)\in H^+} a_{ij} + \sum_{(i,j)\in H^-} a_{ij}\} \qquad (4.25)$$

Por otra parte para cualquier vector de precios P y un flujo factible X, tenemos

$$D(P,X) = \max\{d_H(P) \mid H \text{ es una trayectoria simple no bloqueada con respecto a } X\} \qquad (4.26)$$

cuando no haya una trayectoria simple no bloqueada con respecto a X (caso que se presenta excepcionalmente) definiremos $D(P,X) = 0$. En tal caso debe ocurrir que $b_{ij} = c_{ij}$ para todo (i,j).

Sea $$\beta(P) = \min\{D(P,X) \mid x \in Z^A \text{ es un flujo factible}\} \qquad (4.27)$$

Dado P, solamente hay un número finito de valores que $D(P,X)$ puede tomar, por lo que el mínimo en (4.27) se alcanza para algún X. El siguiente lema muestra que $\beta(P)$ proporciona una medida de suboptimalidad del vector de precios P. La complejidad computacional estimada es proporcional a $\beta(P^0)$ donde P^0 es el vector de precios iniciales.

Lema 4.7

(a) Si para algún $\varepsilon \geq 0$ existe un flujo factible X satisfaciendo $\varepsilon - H.C.$ junto con P entonces

$$0 \leq \beta(P) \leq (N-1)\varepsilon \qquad (4.28)$$

(b) P es solución óptima dual si y sólo si $\beta(P) = 0$

Demostración.

(a) Para cada trayectoria simple H no bloqueada con respecto a X y que tenga $\mid H \mid$ arcos, tenemos que al sumar las condiciones de $\varepsilon - H.C.$ a través de H y usando (4.25)

$$d_H(P) \leq |H|\varepsilon \leq (N-1)\varepsilon \qquad (4.29)$$

por lo tanto, el resultado se sigue de (4.26) y (4.27).

(b) Si P es óptimo entonces satisface $H.C.$ junto a un vector óptimo primal X, de (4.29), usando $\epsilon = 0$ se obtiene $\beta(P) = 0$. Recíprocamente si $\beta(P) = 0$ entonces de (4.27) sabemos que existe un flujo primal factible X tal que $D(P, X) = 0$. De aquí $d_H(P) = 0$ para todas las trayectorias simples H no bloqueadas con respecto a X. Aplicando este hecho a las trayectorias simples H y usando la definición de longitud de ciclo se tiene que X y P satisfacen $H.C.$ por lo que X y P son óptimos. □

Se ha demostrado que $\beta(P)$ es una medida de optimalidad de P y que está muy relacionado con las condiciones de $\epsilon - H.C.$ Veremos que $\beta(P)$ también impone un límite en la cantidad en que los precios pueden incrementarse en el transcurso del algoritmo de ϵ-relajación. El siguiente lema combina el análisis hecho independientemente por Bertsekas y Golbeng y servirá para dos importantes casos del algoritmo: con el uso de escalamiento y sin el uso de escalamiento.

Lema 4.8 Si el problema FCM es factible, el número de incrementos de precio en cada nodo es $O(\beta(P^0)/\epsilon + N)$.

Demostración. Sea (X, P) un par de vectores generados al termino del algoritmo y sea X^0 un vector de flujo que alcanza el mínimo en (4.27). El paso importante es considerar $Y = X - X^0$, que es un flujo (probablemente infactible en la capacidad) dando el mismo incremento a las mismas holguras $\{g_i,\ i \in N\}$ como X. Si $g_t < 0$ para algún nodo t, debe de existir un nodo s con $g_s < 0$ y una trayectoria simple H con $s(H) = s$, $t(H) = t$ y tal que $Y_{ij} > 0$ para todo $(i,j) \in H^+$ y $Y_{ij} < 0$ para todo $(i,j) \in H^-$. Por la construcción de Y, H es no bloqueada con respecto a X^0. Así, de (4.26) debemos tener $d_H(P^0) \leq D(P^0, X^0) = \beta(P^0)$ y usando (4.25) tenemos

$$p_s^0 - p_t^0 - \sum_{(i,j)\in H^+} a_{ij} + \sum_{(i,j)\in H^-} a_{ij} \leq \beta(P^0) \tag{4.30}$$

La construcción de Y también hace que la trayectoria inversa de H deba de ser no bloqueada con respecto a X. Por lo tanto, las condiciones de $\epsilon - H.C.$ nos dicen que $p_j \leq p_i - a_{ij} + \epsilon$ para todo $(i,j) \in H^+$ y $p_i \leq p_j + a_{ij} + \epsilon$ para todo $(i,j) \in H^-$. Al sumar estas condiciones a través de H, obtenemos

$$-p_s + p_t + \sum_{(i,j)\in H^+} a_{ij} - \sum_{(i,j)\in H^-} a_{ij} \leq |H|\epsilon \leq (N-1)\epsilon \tag{4.31}$$

donde $|H|$ es el número de arcos de H. Se tiene $p_s^0 = p_s$ dado que la condición $g_s < 0$ implica que el precio de s no ha sido cambiado, por lo que sumando (4.30) y (4.31) tenemos

$$p_t - p_t^0 \leq \beta(P^0) + (N-1)\varepsilon$$

a través del algoritmo para todos los nodos t con $g_t > 0$. De los supuestos y el análisis previo se concluye que todos los incrementos de los precios son al menos en ε, así que a lo más hay $\beta(P^0)/\varepsilon + N - 1$ de estos , en cada nodo. Como al final puede presentarse un incremento degenerado de precios, el número total de incrementos de precios es $\beta(P^0)/\varepsilon + N$ para cada nodo.□

Se ha establecido una cota en el número de incrementos en precios, por lo que se debe acotar la cantidad de trabajo asociado con cada incremento de precio, para esto veamos algunas definiciones y la cota del tiempo realizado en cada uno de los trabajos.

Trabajo de revisión es el trabajo realizado en el paso (3) del algoritmo, es decir calcular los nuevos precios en los nodos y construir las listas incrementales correspondientes. Se puede demostrar que este trabajo se realiza en $O(A(\beta/\varepsilon + N))$.

∂−incrementos totales son ∂−incrementos del paso 2 en la que un arco es colocado en su cota inferior o superior de flujo (es decir $\partial = c_{ij} - x_{ij}$ o $\partial = x_{ji} - b_{ji}$). Se puede demostrar que este trabajo se realiza en $O(A(\beta/\varepsilon + N))$.

∂−incrementos parciales son ∂−incrementos del paso 2 en la que un arco es colocado en un nivel de flujo entre sus cotas inferior y superior. El problema latente que se presenta es poder caer en un ciclo, y se soluciona usando un orden para revisar los nodos (llamada implementación de barrida) con lo que la complejidad de los ∂−incrementos parciales es $O(N^2(\beta/\varepsilon + N))$. Por lo que el método de ε-relajación tiene una complejidad total, en el peor de los casos de $O(N^3 + N^2(\beta/\varepsilon))$ pero depende de β y de ε, lo que puede hacerlo exponencial. Por ejemplo, una cota natural para β es NC con C el máximo valor absoluto de los costos de los arcos y $\varepsilon = 1/(N+1)$, para garantizar optimalidad a la terminación del algoritmo, así la cota de la complejidad se convierte en $O(N^4C)$. Sin embargo el uso de procedimientos de escalamiento con una estructura de datos novedosa llamada árboles dinámicos que se deben a [*GoT*90] y proporciona una complejidad $O(NA\log(N)\log(NC))$. Otra opción es usar otra forma de escalamiento conocido como escalamiento de costo que hacen que la complejidad del algoritmo sea $O(N^3 \log NC)$ y es debido a [*BeE*88].

5 RESULTADOS COMPUTACIONALES

En la primera sección de este capítulo se comenta la elección de una estructura de datos adecuada para lograr una implementación computacional eficiente de un problema de flujo a costo mínimo, particularmente para los algoritmos de relajación y subasta presentados en este trabajo. En la segunda sección se presentan los comentarios de algunos de los resultados obtenidos, en la comparación computacional de estos algoritmos.

5.1 ESTRUCTURA DE DATOS DE LA IMPLEMENTACION

Para implementar eficientemente un algoritmo de optimización en redes es esencial aprovechar la naturaleza de la gráfica del problema usando estructuras de datos apropiadas. Existen dos decisiones principales a este respecto

1. Representar el problema en una forma que facilite la aplicación del algoritmo
2. Utilizar estructuras de datos adicionales que realicen eficientemente las operaciones del algoritmo.

Para todos los algoritmos de este trabajo la representación apropiada del problema es muy sencilla, por ejemplo supongamos que tenemos el siguiente problema de flujo a costo mínimo con cotas inferiores iguales a cero, que es la forma estándar utilizada en los códigos de los programas para resolver el problema de flujo a costo mínimo.

$$\begin{aligned} \min_{s.a.} \quad & \sum_{(i,j)\in A} a_{ij}x_{ij} \\ & \sum_{\{j|(i,j)\in A\}} x_{ij} - \sum_{\{j|(j,i)\in A\}} x_{ji} = s_i \qquad \forall \quad i \in N \\ & 0 \le x_{ij} \le c_{ij} \qquad \forall\,(i,j) \in A \end{aligned}$$

Una manera de representar este problema es utilizar cuatro arreglo de tamaño A y un arreglo de tamaño N, que almacenan lo siguiente

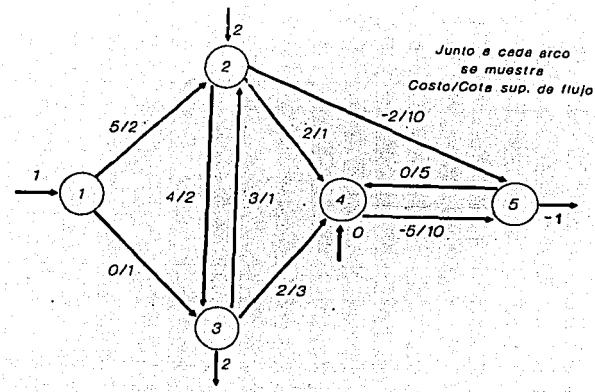
$INICIO(a)$	El nodo inicial del arco a
$FINAL(a)$	El nodo final del arco a
$COSTO(a)$	El coeficiente de costo del arco a
$CAPACIDAD(a)$	La cota superior de flujo del arco a
$OFERTA(i)$	La oferta del nodo i

Una representación alternativa es almacenar los costos a_{ij} y la cota superior c_{ij} en un arreglo de dos dimensiones de tamaño $N \times N$ (o en un arreglo unidimensional de longitud N^2, con los elementos de cada renglón almacenados contiguamente). Esta representación desperdicia memoria y requiere mucho trabajo adicional cuando el problema es poco denso ($A \ll N^2$) pero puede ser una buena elección para problemas densos porque evita el almacenamiento de los nodos inicial y final de cada arco. Para los métodos de relajación (Código Relaxqc) además de los cinco arreglos mencionados, se necesitan estructuras de datos adicionales que faciliten las operaciones de etiquetado y el descubrimiento de direcciones de ascenso. En particular, es necesario tener fácil acceso al conjunto de arcos incidentes a cada nodo. Esto puede ser hecho con la ayuda de los siguientes cuatro arreglos adicionales

$PRIMERINT(i)$	El primer arco interior del nodo i (= 0 si i no tiene arcos interiores).
$PRIMEREXT(i)$	El primer arco exterior del nodo i (= 0 si i no tiene arcos exteriores).
$PROXINT(a)$	El siguiente arco después de a con el mismo nodo final de a (= 0 si a es el último arco interior del nodo final de a).
$PROXEXT(a)$	El siguiente arco después de a con el mismo nodo inicial de a (= 0 si a es el último arco interior del nodo inicial de a).

La figura 5.1 ilustra la utilización de estos arreglos con un problema de flujo a costo mínimo

Como un ejemplo de su uso, supongamos que queremos revisar todos los arcos interiores del nodo i. Primero obtenemos el arco a_1 =PRIMERINT(i), entonces el arco a_2 =PROXINT(a_1), el arco a_3 =PROXINT(a_2), etc, hasta el arco a_k para el cual PROXINT(a_k)=0.



ARCO	INICIO	FIN	COSTO	CAPACIDAD	PROXINT	PROXEXT
1	1	2	5	2	4	2
2	1	3	0	1	3	0
3	2	3	4	2	0	5
4	3	2	3	1	0	7
5	2	5	−2	10	0	6
6	2	4	2	1	7	0
7	3	4	2	3	8	0
8	5	4	0	5	0	0
9	4	5	5	10	5	0

NODO	OFERTA	PRIMERINT	PRIMEREXT
1	1	0	1
2	2	1	3
3	−2	2	4
4	0	6	9
5	−1	9	8

Figura 5.1 Representación de los datos de un problema de flujo a costo mínimo en términos de los nueve arreglos INICIO, FIN, COSTO, CAPACIDAD, PROXINT, PROXEXT, OFERTA, PRIMERINT, PRIMEREXT.

Es posible no utilizar el arreglo PROXEXT si los arcos son almacenados en el orden de su nodo inicial, esto es, los arcos externos de cada nodo i son los arcos de PRIMEREXT(i) a PRIMEREXT($i+1$)-1, por lo que el arreglo PRIMEREXT es suficiente para generar todos los arcos externos de cualquier nodo. Haciendo esto se ahorra la memoria de un arreglo y usualmente unos calculos. La desventaja de esta idea es que complica el análisis de sensibilidad. En particular, si los datos del problema son cambiados para añadir o suprimir algunos arcos, todos los arreglos que describen el problema, excepto OFERTA, deben ser recompilados.

La implementación de los algoritmos subasta y sus principales operaciones que involucra revisar los arcos incidentes a los nodos, son análogas a las realizadas por el método de relajación, por lo que la estructura de datos y la discusión precedente también se aplican a los algoritmos subasta. Por último, un conjunto similar de arreglos a los denominados PRIMERINT, PRIEMREXT pueden ser usados para almacenar los arcos de las listas incrementales en el método de ϵ−relajación.

Los programas estan escritos en FORTRAN y tienen el siguiente formato común

a) Leen el problema en una forma estándar común

b) Convierten el problema a una forma especial según el algoritmo utilizado

c) Llama el algoritmo que resuelve el problema

d) Proporciona resultados

5.2 RESULTADOS COMPUTACIONALES

Es importante hacer notar que las comparaciones presentadas son de implementaciones de los algoritmos y no de los algoritmos en si. Además, el desempeño de los códigos depende de muchos factores como son los ejemplos seleccionados, la computadora y hasta el compilador utilizado.

La computadora empleada para compilar y correr los programas fue una Compaq 486 a 33 Mhz utilizando el compilador de FORTRAN de Microsoft versión 5.1.

Los problemas empleados fueron generados usando el programa llamado GRIDGEN que construye problemas aleatorios no bipartitos y al parecer este generador no tiene ninguna característica

especial, es decir los problemas que genera son muy parecidos a los que se pueden obtener con otros generadores. El programa GRIDGEN genera solamente problemas factibles, utilizando para esto una estructura de malla.

Debemos mencionar además, que las comparaciones que se presentan a continuación son limitadas, dado que algunos tipos especiales de problemas prácticos tienen estructuras que no se pueden obtener utilizando generadores aleatorios. Sin embargo, los resultados presentados en este trabajo, se asemejan a los proporcionados por Bertsekas en [$BeT88a$] y a los de Glover en [$GKP92$]. Estos autores concluyen que la familia de algoritmos de relajación, conocidos como RELAX son unos de los más rápidos para resolver problemas de flujo a costo mínimo.

En la figura 5.2 se presentan los parámetros más importantes utilizados en la generación de los problemas empleados para la comparación de las implementaciones computacionales.

PROBLEMA NUMERO	NUMERO DE NODOS	NUMERO DE ARCOS	NUMERO DE FUENTES Y SUMIDEROS	OFERTA	SOLUCION
1	10 × 10	600	20,20	5,000	690,513
2	10 × 10	1000	20,20	5,000	443,428
3	10 × 20	1500	50,50	10,000	1,149,228
4	20 × 10	3000	50,50	10,000	600,305
5	10 × 40	2400	100,100	20,000	2,704,184
6	20 × 20	4000	100,100	20,000	1,813,627
7	40 × 10	6000	100,100	20,000	1,093,335
8	20 × 20	9000	100,100	200,000	26,613,401
9	25 × 20	3000	150,150	25,000	3,077,039
10	25 × 20	5000	150,150	240,000	34,487,136
11	25 × 20	9000	150,150	450,000	62,123,410
12	25 × 20	10000	150,150	450,000	64,294,077

Figura 5.2 Características de los problemas resueltos.

En la figura 5.3 se presenta el tiempo requerido para resolver doce problemas de FCM obtenidos empleando el generador GRIDGEN, utilizando las siguientes implementaciones:

- **RELAXQC:** Es una implementación del algoritmo de relajación que mantiene en una cola el conjunto de nodos con holgura distinta de cero.
- **ϵ-RELAX:** Es una implementación del algoritmo de ϵ-relajación. Itera solamente a partir de nodos con holgura positiva.
- **ϵ-RELAXN:** Es una implementación del algoritmo de ϵ-relajación que itera tanto de nodos con holguras positivas como negativas.
- **NETFLO:** Es una implementación del método primal simplex de la gran M, debida a Kennigton y Helgason. Un listado de este código se encuentra en [*KeH*80].

PROBLEMA NUMERO	NETFLO	RELAXQC	ϵ-RELAX	ϵ-RELAXN (min,min)	ϵ-RELAXN (max,max)
1	.72	.17	.33	.66	.88
2	1.27	.22	.55	.50	.60
3	2.91	.38	1.37	1.70	1.59
4	5.61	.61	2.36	2.15	2.25
5	7.64	.87	2.75	3.40	3.63
6	12.80	1.21	4.17	4.1	5.05
7	18.73	1.26	5.54	4.67	5.11
8	38.56	5.02	10.82	17.73	18.83
9	10.49	1.10	4.45	6.10	6.43
10	29.17	3.51	8.84	11.75	12.36
11	45.37	5.98	11.37	21.31	48.89
12	58.89	7.14	13.51	28.14	54.63

Figura 5.3 Tiempo necesario en segundos para resolver los problemas con las implementaciones utilizadas.

El algoritmo de ε-relaxn necesita dos parámetros, uno es el factor de escalamiento que se recomienda estar entre 4 y 10. El otro parámetro se basa en el costo máximo de los arcos, costomax=$\max_{(i,j)\in A}|c_{ij}|$ y se recomienda estar en el rango $\frac{costomax}{20}$ y $\frac{costomax}{2}$. Los resultados presentados en la tabla de ε-relaxn (min,min) corresponden a la utilización de un factor de escalamiento de 4 y una epsilon de $\frac{costomax}{20}$ mientras que ε-relaxn (max,max) corresponden a un factor de escalamiento de 4 y una epsilon de $\frac{costomax}{2}$. Se probaron otras combinaciones, en las cuales el tiempo de solución no varia mucho, pero son mayores que las presentadas en la tabla.

Otra ventaja de los métodos de relajación y subasta con respecto a los algoritmos simplex primales es que son mejores para el análisis de sensibilidad. Por ejemplo, supongamos que se resuelve un problema y se modifica al cambiar las capacidades de algunos arcos y/o las ofertas de los nodos. Para resolver el problema modificado usando relajación o subasta, si las variables duales obtenidas en la solución previa son las variables duales iniciales generalmente esta solución inicial esta muy cerca del óptimo y la solución al problema modificado se obtiene rápidamente. En contraste, para resolver el problema modificado usando simplex primal, se debe contar con una base inicial y la base obtenida en la solución previa generalmente no es una base para el problema modificado. como consecuencia, una nueva base inicial debe construirse y no hay maneras ismples de escoger una base cercana a la óptima.

Una desventaja aparente de los algoritmos relaxqc y ε-relaxn es que necesitan guardar en memoria 6 arreglos de tamaño $|A|$ y 7 arreglos de tamaño $|N|$, mientras que algunos códigos simplex primales pueden ser implementados utilizando 4 arreglos de tamaño $|A|$ y 4 de tamaño $|N|$.

6 COMENTARIOS FINALES

El problema de flujo a costo mínimo es de gran interés en la Investigación de Operaciones, debido a su amplia aplicabilidad en problemas de diversos campo de estudio, así como por su estructura al plantearlo como problema de Programación Lineal.

En este trabajo se desarrollan algoritmos teóricamente eficientes para resolver el problema de flujo a costo mínimo, presentando y analizando las bases teóricas de dos nuevos algoritmos que han venido a aportar nuevas líneas de investigación en el desarrollo de la Programación Matemática y de la Optimización Combinatoria.

Uno de ellos es el algoritmo de relajación, que adopta ideas de la Programación no Lineal y es un método de ascenso dual que calcula fácilmente las direcciones de mejoramiento de la función dual . La implementación presentada es muy eficiente para una gran cantidad de problemas generados aleatoriamente y mejor portada que los métodos simplex primales para el análisis de sensibilidad. Una desventaja de esta implementación es que requiere un poco más de memoria computacional que la utilizada en los métodos simplex primales. Aunque los resultados computacionales aquí presentados son limitados proporcionan una idea del excelente rendimiento práctico del método de relajación y son parecidos a los reportados por Bertsekas [*BeT*88*a*].

El otro algoritmo, llamado de ϵ-relajación, es una especialización del algoritmo de subasta que surge del problema de asignación con la motivación de encontrar una solución en paralelo de los problemas de flujo en redes. Usa la idea de ϵ-Holguras Complementarias, que es esencial para determinar que el algoritmo no termine antes de alcanzar la optimalidad, además que estas condiciones son útiles en la construcción de algoritmos de escalamiento y pueden ser vistas como un relajamiento de las condiciones clásicas de Holguras Complementarias. La implementación presentada utiliza una estructura de datos sencilla combinada con conceptos de escalamiento, lo que hace que su complejidad computacional sea favorable, aunque su comportamiento práctico no es tan bueno como el del algoritmo de relajación.

En la parte teórica de este trabajo, se proporcionan las bases adecuadas para que utilizando estos enfoques se puedan elaborar e implementar algoritmos específicos para otros problemas de flujo en redes.

La importancia de estos métodos se debe a que pueden aplicarse a una amplio rango de problemas más alla de los tratados en este trabajo, incluyendo problemas generales de Programación Lineal. También existe esfuerzos dirigidos a la implementación en paralelo de este tipo de algoritmos lo que se cree mejorara tanto la complejidad como el aspecto práctico de los algoritmos.

7 BIBLIOGRAFIA

[AMO93] Ahuja, R. K., Magnanti, T. L. y Orlin, J.B., 1993. Network Flows. Prentice Hall, Englewood Cliffs, N. J.

[BHT87] Bertsekas, D.P., Hossein, P. y Tseng, P., 1987. "Relaxation Methods for Network Flows Problems with Convex Arc Costs" SIAM, J. on Control and Optimization, Vol. 25 pags 1291-1243.

[BJS90] Bazaraa, M. S., Jarvis, J.J. y Sherali, H.D., 1990. Linear Programming and Network Flows (2a. Edición), Wiley, N.Y.

[Bal86] Balinski, M. L., 1986. "A Competitive (Dual) Simplex Method for the Assignment Problem" Math. Programming, Vol. 34 pags 125-141.

[BeC89] Bertsekas, D. P. y Castañon, D. A., 1989. "The auction Algorithm for Transportation Problems" Annals of Operations Research, Vol 20 pags 67-96.

[BeE87a] Bertsekas, D. P. y El Baz, D., 1987. "Distributed Asynchronous Relaxation Methods for Convex Network Flow Problems" SIAM, J. on Control and Optimization, Vol 25 pags 74-85.

[BeE87b] Bertsekas, D. P. y Eckstein, J., 1987. "Distributed Asynchronous Relaxation Methods for Linear Network Flow Problems" Proc. of IFAC'87. Munich, Alemania.

[BeE88b] Bertsekas, D. P. y Eckstein, J., 1988. "Dual Coordinate Step Methods for Linear Network Flow Ploblems" Math. Programming, Series B. Vol 42 pags 203-243.

[BeG87] Bertsekas, D. P. y Gallager, R. G., 1987. Data Networks. Prentice Hall, Englewood Cliffs, N. J.

[BeT88a] Bertsekas, D. P. y Tseng, P., 1988. "Relaxation Methods for Minimum Cost Ordinary and Generalized Network Flow Problems." Operations Research, Vol 36 pags 93-114.

[BeT88b] Bertsekas, D. P. y Tseng, P., 1988. "RELAX: A Computer Code for Minimum Cost Network Flow Problems." Annals of Operations Research, Vol 13 pags 127-190.

[BeT89] Bertsekas, D. P. y Tsitsiklis, D., 1989. Parallel and Distributed Computation: Numerical Methods, Prentice Hall, Englewood Cliffs, N, J.

[Ber81] Bertsekas, D. P., 1981. "A New Algorithm for the Assignment Problem", Math. Programming, Vol 21 pags. 152-171.

[BeT82] Bertsekas, D. P., 1982. Constrained Optimization and Lagrange Multiplier Methods, Academic Press, N.Y.

[Ber85a] Bertsekas, D. P., 1985. "A Unified Framework for Minimum Network Flow Problems." Math. Programming, Vol 32 pags 125-145.

[Ber85b] Bertsekas, D. P., 1985. "A Distibuted Relaxation Algorithm for the Assigment Problem" Proc. 24th IEEE Conference on Decision and Control, Ft, Lauderdale, Fla., pags 1703-1704.

[Ber86] Bertsekas, D. P., 1986. "Distibuted Relaxation Methods for Linear Network Flow Problems" Proc. 25th IEEE Conference on Decision and Control, Atenas, Grecia, pags 2101-2106.

[Ber88] Bertsekas, D. P., 1988. "The Auction Algorithm: A Distibuted Relaxation Method for the Assigment Problem" Annals of Operations Research, Vol 14, pags 105-123.

[Ber92] Bertsekas, D. P., 1992. Linear Network Optimization: Algorithms and Codes. MIT Press, Cambridge, Masachusetts.

[Chr75] Christofides, N., 1975. Graph Theory: An Algorithm Approach. Academic Press, N. Y.

[Cun76] Cunningham, W. H., 1979. "A Network Simplex Method" Math. Programming, Vol 4, pags 105-116.

[Dan63] Dantzig, G. B., 1963. Linear Programming and Extensions, Pricenton Univ. Press, Pricenton, N. J.

[EdK72] Edmonds, J. y Karp, R. M., 1972. "Theorical Improvements in Algorithmic Efficiency for Network Flow Problems" Journal of the ACM, Vol. 19, pags 248-264.

[FoF62] Ford, L. R., y Fulkerson, D. R., 1962. Flow in Networks, Princeton Univ. Press, Princeton, N. J.

[GKK74] Glover, F., Karney, D. y Klingman, D., 1974. "Implementation and Computational Comparasions of Primal, Dual, and Primal-Dual Computer Codes for Minimum Cost Network Flow Problem" Networks, Vol. 4, pags 191-212.

[GKP92] Glover, F., Klingman, D. y Phillips, 1992. Network Models in Optimization and their Applications in Practice. Wiley Interscience, N.Y.

[GoT90] Goldberg, A. V. y Tarjan, R. E., 1990. "Solving Minimum Cost Flow Problems by Successive Aproximation" Math. of Operations Research, Vol. 15, pags 430-466.

[Gri86] Grigoriadis, M. D., 1986. "An Efficient Implementation of the Network Simplex Method". Mathematical Programming, Study 26, pags 83-111.

[JeB80] Jensen, P. A. y Barnes, J. W., 1980. Netwok Flow Programming, Wiley, N.Y.

[KNS74] Klingman, D., Kenningotn, J. y Stutz, J., 1974. "NETGEN - A Program for Generating Large Scale (Un)Capacited Assignment, Transportation, and Minimum Cost flow Network Problems" Management Science, Vol. 20, pags 814-822.

[KeH80] Kennington, J. y Helgarson, R., 1980. Algorithms for Network Programming, Wiley, N. Y.

[Law76] Lawler, E., 1976. Combinatorial Optimization: Networks and Matroids, Holt, Reinhart, and Wiston, N.Y.

[PaS82] Papadimitriou, C. H. y Steiglitz, K., 1982. Combinatorial Optimization: Algoritms and Complexity, Prentice Hall, Englewood Cliffs, N. J.

[Roc84] Rockafellar, R. T., 1984. Network Flows and Monotropic Programming, Wiley Interscience, N. Y.

[TsB87] Tseng, P. y Bertsekas, D. P., 1987. "Relaxation Methods for Linear Programs" Math. of Operations Research, Vol. 12, pags 569-596.

[TsB90] Tseng, P. y Bertsekas, D. P., 1990. "Relaxation Methods for Monotropic Programs" Math. of Operations Research, Vol. 46, pags 127-151.

[Zad79] Zadeh, N., 1979. "Near Equivalence of Network Flow Algorithms" Technical Report No. 26, Dept. of Operations Research, Stanford University, CA.