UNIVERSIDAD NACIONAL AUTÓNOMA

DIVISION DE ESTUDIOS DE POSGRADO
FACULTAD DE INGENIERIA

ALGORITMOS PARA RESOLVER PROBLEMAS DE REDES DE FLUJO CON COSTOS LINEALES Y SIN GANANCIA.

T E S I S

QUE PARA OBTENER EL GRADO
DE MAESTRO EN INGENIERIA

presenta:

ALBERTO CADENA LANDETA

Octubre 1982

México, D. F.

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

CONTENIDO

CONTENIDO

CONTENIDO

CAPITULO I

GENERALIDADES:

1.1 Introducción

1.2 Métodos

1.3 Desarrollo del Trabajo

1.1 INTRODUCCION

Algunos de los problemas que se presentan en la vida diaria tienden a ser resueltos en forma más eficiente y rápida. Justamente "La programación de redes de flujo" es una de las herramientas más recientes, la misma, que está empezando a ser usada con mayor frecuencia; debido a que los avances en los métodos computacionales, orientados a resolver este tipo de problemas son continuos y constantes. Los modelos de redes de flujo están siendo aplicados en muchos problemas, tales como, sistemas de inventarios, sistemas de aprovechamientos hidráulicos, sitemas de redes eléctricas, sistemas de transporte, sistemas de distribución, - sistemas de producción, etc.

El objetivo del presente trabajo es describir, analizar y proporcionar los programas en FORTRAN de redes de flujo que permiten resolver los problemas de ruta más corta, flujo máximo y - flujo a costo mínimo, considerando que el flujo se mantiene constante. Además se ilustrará una serie de aplicaciones de cada uno de estos modelos.

1.2 METODOS

Los modelos de redes son casos particulares de la programación lineal que disponen de métodos de solución propios y resultan más eficientes que el método Simplex.

Cada uno de estos problemas tienen su método de solución.

En el problema de ruta más corta se consideran tres casos:

1. Todos los arcos tienen longitud positiva
2. Algunos arcos tienen longitud negativa pero no forman ciclos negativos.
3. Algunos arcos tienen longitud negativa y forman uno o - más ciclos negativos.

Para representar el problema de ruta más corta, el parámetro longitud de cada arco se considera como el costo del mismo.

El primer caso se resuelve aplicando el método de Dijkstra. El segundo caso se resuelve aplicando el método de Dijkstra y - luego un método primal que iterativamente adiciona y remueve los

arcos desde un árbol primal básico, El tercer caso, se resuelve combinando el método Dijkstra y el método primal para obtener el algoritmo SHORT, este algoritmo termina cuando encuentra en ci-- clo negativo o encuentra la solución óptima. Para resolver este problema, también se emplean el método no básico, que no usa la base de el árbol y finalmente el método dual.

El problema de flujo máximo se resuelve mediante el uso de métodos básicos y no básicos. Los métodos no básicos son procedimientos iterativos. Los métodos básicos en cambio mantienen a lo largo de su proceso un árbol básico y su procedimiento es más - complejo.

El problema de flujo a costo mínimo se resuelve mediante - el uso de algoritmos primales. Dentro de los cuales se conside-- ran los métodos de flujo máximo y del arco artificial para obtener una solución primal factible. Finalmente los métodos primal no básico y primal básico para obtener la solución óptima.

1.3 DESARROLLO DEL TRABAJO

Al considerar que lo más importante de este trabajo es dar las facilidades necesarias al interesado para que pueda utilizar los progrmas en FORTRAN en la solución de los tres tipos de problemas. Se pone más enfásis en la presentación del Manual del - Usuario de los programas, a tal punto que pueda emplearlos efi-- cazmente, sin necesidad de profundizar en los fundamentos matemáticos de redes y sin conocer la codificación de los algoritmos.

El Capítulo II define los conceptos básicos sobre redes y el modelo matemático de redes. El Capítulo III inicia describiendo los algoritmos básicos que leen los datos, representan la red transforman los parámetros de los nodos y los arcos y determinan los apuntadores tanto de la red como del árbol, luego se plantea cada uno de los problemas, sus métodos de solución, haciendo un análisis de cada uno de los algoritmos, con sus respectivos ejemplos de aplicación. Finalmente se adjuntarán los anexos que lo - constituyen el Manual de uso de los programas y los listados de codificación de los problemas, programas en FORTRAN.

CAPITULO II

CONCEPTOS BASICOS DE REDES

2.1 Introducción
2.2 Gráfica dirigida y su caracterización
2.3 Cadena, Trayectoria, circuito y ciclo
2.4 Concepto y tipos de redes
2.5 Arbol y sus caracterizaciones
2.6 Notación de redes
2.7 Modelo matemático de redes.
2.8 Redes expandida y marginal.

2.1 INTRODUCCION

En este capítulo se presentan algunas definciones básicas de la teoría de redes, con el propósito de unificarlas y evitar confusiones al usuario. Las mismas que serán usadas posteriormente en este trabajo. En la primera parte de este capítulo se describen algunos conceptos elementales de teoría de gráficas, como la forma de caracterizar una gráfica dirigida a través de su matriz de incidencia nodos-nodos y nodos-arcos. Se definen los conceptos de cadena, trayectoria, circuito y ciclo; con algunos ejemplos ilustrativos.

Se introduce de manera natural el concepto de red, que es la gráfica dirigida más importante. Se presentan algunos tipos de redes tales como la red bipantita, la red simple y la red circulatoria. Se menciona la transformación de toda red reducida en red circulatoria, mostrando en forma gráfica esta transformación. Se describen el concepto de árbol y sus caracterizaciones. Se mencionan algunos otros conceptos sobre notación de redes, el modelo matemático de redes y finalmente las redes expandida y marginal.

2.2 GRAFICA DIRIGIDA Y SU CARACTERIZACION

Es denotada por $G=(N,M)$, consiste de un conjunto finito $N=(1,2,\ldots,i,\ldots,n)$ cuyos elementos se denominan nodos y un conjunto $M=(1,2,\ldots,k,\ldots,m)$ cuyos elementos se denominan arcos que estan formados por pares ordenados de nodos, asi un arco se define como el arco $k(i,j)$ o (i,j), el mismo que se origina en el nodo i (llamado nodo origen) y termina en el nodo j (llamado nodo terminal). Puede presentarse el arco $k(i,i)$ el cual se origina y termina en el nodo i y el arco $k(j,i)$ el cual se origina en el nodo j y termina en el nodo i, en todos los casos anteriores la dirección de los arcos son diferentes. La forma clásica de construir una gráfica dirigida es dibujando circulos pequeños que no se interceptan, cuyos números son los nodos $i,j \in N$ y las líneas o flechas son los arcos $(i,j) \in M$. Para su mejor entendimiento se ilustra la figura 2.1. cuya gráfica dirigida $G=(N,M)$ esta formada por cuatro nodos $N=(1,2,3,4)$ y seis arcos $M=\{(1,2), (2,2), (2,4), (3,2), (4,3), (3,1)\}$.

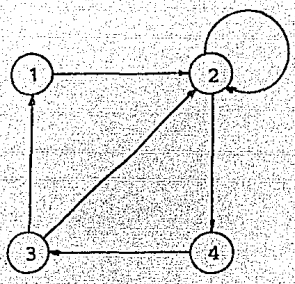


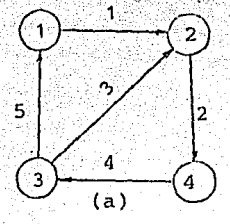
Fig. 2.1
Gráfica dirigida

Entre las diferentes formas de caracterizar una gráfica dirigida es por medio del denominado concepto de la matriz de adyacencia nodos-nodos, que consiste en una matriz A de orden n x n, donde n es el número de nodos. En esta matriz $A=(a_{ij})$ se tiene que $a_{ij}=1$, si existe un arco que va del nodo i al nodo j; en el caso contrario $a_{ij}=0$. La matriz de adyacencia asociada a la gráfica dirigida de la figura 2.1 es:

$$A = \begin{array}{c} \begin{matrix} \text{NODOS} & 1 & 2 & 3 & 4 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} \text{NODOS} \\ 1 \\ 2 \\ 3 \\ 4 \end{matrix}$$

Otra forma de caracterizar una gráfica dirigida es por medio de la matriz de incidencia nodos-arcos, que consiste en una matriz D de orden nxm, donde n es el número de nodos y m es el número de arcos, los que han sido previamente enumerados. En la matriz $D=(d_{ij})$ se tiene que $d_{ij}=1$, si del nodo i parte el arco con el número j; $d_{ij}=-1$, si el nodo i llega al arco con el número j;

en otros casos, $d_{ij}=0$. Note que esta caracterización es solo aplicable a una gráfica dirigida que no tiene arcos de la forma (i,i) la figura 2.2 muestra un ejemplo.



(a)

ARCOS	1	2	3	4	5	NODOS
	1	0	0	0	-1	1
D =	-1	1	-1	0	0	2
	0	0	1	-1	1	3
	0	-1	0	1	0	4

(b)

Fig. 2.2

2.3 CADENA, TRAYECTORIA, CIRCUITO Y CICLO

En una gráfica dirigida se define una cadena del nodo i al nodo j como la sucesión de nodos distintos, pertenecientes a N, denotados por $i=i_1, i_2,\ldots,i_r=j$ y arcos de M denotados por $a_1,a_2\ldots a_r$ tales que $a_t=(i_t,i_t+1)$, donde $t = 1,\ldots,r-1$. Si no hay ambiguedad, solo se especifican los nodos que forman la cadena. Trayectoria del nodo i al nodo j, tiene la misma definición de cadena, con la diferencia de que en la trayectoria cada arco puede tener la forma $a_t=(i_t,i_t+1)$ ó bien $a_t=(i_{t+1},i_t)$, donde t= ,...,r-1.

Es una gráfica dirigida G=(N,M) se define un circuito como una cadena, en el cual el nodo inicial es igual al nodo final. En cambio un ciclo es una trayectoria con el mismo nodo inicial y final. En las cadenas y circuitos los arcos tienen el mismo sentido, además toda cadena es una trayectoria y todo circuito es un ciclo, pero reciprocamente no es cierto.

De la figura 2.1, obtenemos un ejemplo de cada uno de los conceptos anteriores:

1. Cadena del nodo 2 al 4: nodos 2,1,3,4; Arcos (2,1),(1,3)y (3,4)
2. Trayectoria de 4 a 2; nodos 4,2; arco (2,4)
3. Circuito de 1 a 1, nodos 1,2,4,3,1; arcos (1,2),(2,4),(4,3) y (3,1)
4. Ciclo de 3 a 3; nodos 3,4,2,3; arcos (4,3),(2,4) y (3,2)

2.4 CONCEPTO Y TIPOS DE REDES

La red se define como una gráfica dirigida G=(N.M) en donde no existen arcos de la forma $(i,i)\in M$ y dentro de las gráficas diri

gidas la red es la más importante. Además, dado el nodo $i \in N$, se denota por d_i a la "disponibilidad" en este nodo y se dice que es nodo de depósito ($d_i > 0$), destino ($d_i < 0$) o traspaso ($d_i = 0$), si la disponibilidad es positiva, negativa o cero, respectivamente.

Existen diferentes tipos de redes. Aqui se mencionan las - más comunes: Una red, $G=(N,A)$, es bipantita si el conjunto de nodos N puede dividirse en dos subconjuntos N_1, N_2, tales que si - $(i,j) \in M$ entonces, $i \in N$, y $j \in N_2$. Una red, $G=(N,M)$, es simple si - tiene un solo depósito s y un solo sumidero r, y no existen arcos de la forma (i,s) o (r,j) donde $i, j \in N$. Red circulatoria si todos sus nodos son de traspaso. Estos tres tipos de redes se ilustran en la figura 2.3

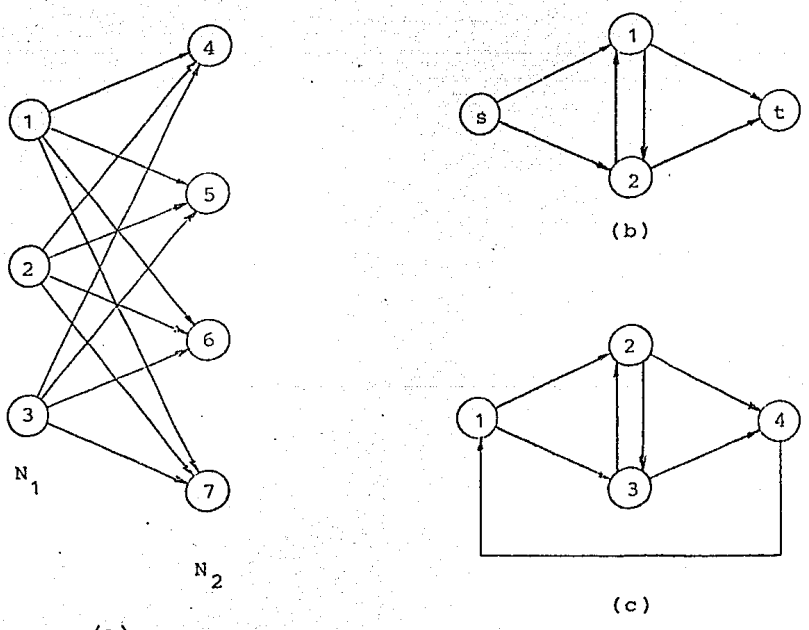


Figura 2.3
(a) Red bipartita (b) Red simple (c) Red circulatoria

En una red es común asociar a cada arco los parámetros correspondientes a los flujos mínimo y máximo permitidos, y al costo unitario por unidad de flujo que pasa por ese arco. En cada arco k(i,j), estos parámetros se denotan por c_k, $\bar{c}_k$ y h_k respectivamente. Además es fácil convertir una red que tiene uno o más nodos fuente y uno o más nodos sumidero a una red circulatoria. Para lo cual se añade a la red original un nodo inicial s que se conecta a cada uno de los nodos fuente, y cada uno de los nodos sumidero se conectan a un nodo terminal t. Finalmente se conectan los nuevos nodos t y s mediante un arco, para obtener la red circulatoria. Esto se ilustra en la figura 2.4, en la que además se muestra la transformación de los respectivos parámetros de los nodos a parámetros de los arcos (las ofertas y demandas en los nodos se transofrman a flujos mínimo y máximos permitidos, incluyendo los costos con valor cero en los arcos). En esta forma las dos redes son equivalentes; como se puede observar en la Figura 2.4

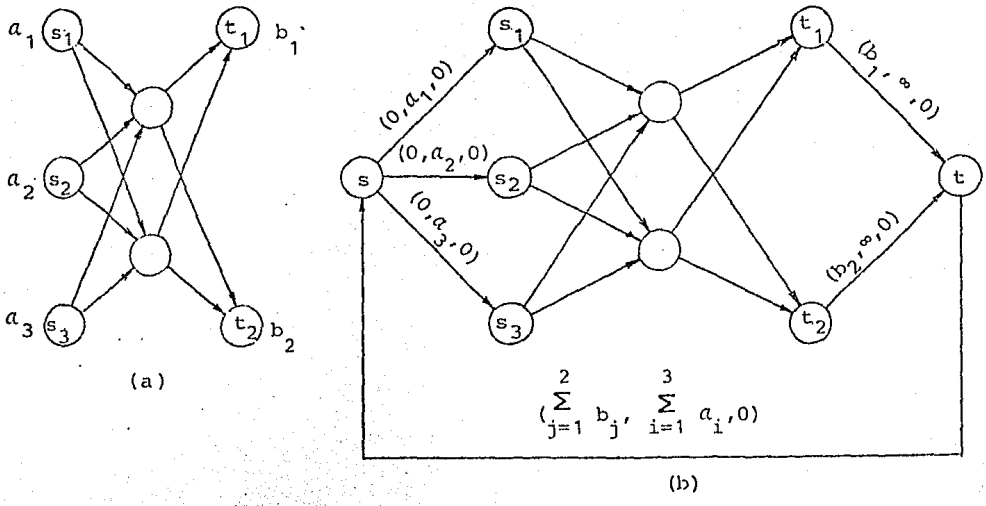


Figura 2.4

(a) Red con fuentes y destinos multiples (b) Red circulatoria equivalente.

2.5 ARBOL Y SUS CARACTERIZACIONES

Un Arbol es una gráfica G=(N,M) que se caracteriza por ser conectada y no tener ciclos, existe una trayectoria única entre cada par de nodos, no tiene ciclos pero exactamente un ciclo se forma al añadir un arco, es conectada pero deja de serlo si algún arco es quitado. Además si G es un árbol, G tiene n-1 arcos y no tiene ciclos y G tiene n-1 arcos y es conectada (siendo n el número de nodos). Sea G=(N,M) una gráfica. Se define el grado de un nodo como el número de arcos que inciden en i. Se dice que un nodo es terminal si tiene grado uno o sea un solo arco - sale o entra en ese nodo. Además la suma de los grados de todos los nodos de una red es igual a dos veces el número de arcos - existentes, ya que cada arco contribuye con dos unidades en la suma de los grados.

2.6 NOTACION DE REDES

En base a los conceptos anteriores se definen las listas - de nodos origen y de nodos terminales, como:

$$O = (O_1, O_2, \ldots, O_m)$$

$$T = (t_1, t_2, \ldots, t_m)$$

Adicionalmente se definen las listas de arcos que se originan en el nodo i y de arcos que terminan en el nodo i, como:

$$M_{oi} = (k \mid O_k = i)$$

$$M_{ti} = (k \mid t_k = i)$$

El flujo f_k o f(i,j), es una variable que especifica la - cantidad de flujo en el arco (flujo de personas, flujo de fluidos, flujo de monedas, etc.). Una característica principal es - la conservación de flujo en los nodos o sea que el flujo que entra a un nodo es igual al flujo que sale del mismo nodo. Además se analizarán los problemas en los cuales el flujo en los arcos no cambian. El flujo $f=(f_1, f_2, \ldots, f_n)^t$, es el vector fila transpuesto.

El costo $h_k(f_k)$, esta asociado con el flujo en el arco y - es función del mismo, siendo independiente del flujo en los - -

otros arcos. Aquí se estudia el caso en el cual la función del costo es lineal, o sea $h_k(f_k) = h_k f_k$, donde h_k esta dado por el vector de costos $h = (h_1, h_2, \ldots, h_m)$. Se pretende minimizar el costo en la red, el cual esta dado por la suma de los costos en los arcos.

$$H(f) = \sum_{k=1}^{m} h_k f_k = hf$$

La capacidad máxima c_k es un parámetro que determina la máxima cantidad de flujo en el arco k ($f_k \leq c_k$), cuyo vector fila transpuesto es $c = (c_1, c_2, \ldots, c_k)^t$. Asimismo habrá situaciones en que se debe especificar la capacidad mínima $\underline{c}_k$ que es la mínima cantidad de flujo en cada arco k, ($f_k \geq c_k$), cuyo vector fila transpuesto es $\underline{c} = (\underline{c}_1, \underline{c}_2, \ldots, \underline{c}m)^t$. Si no se especifica la capacidad mínima en el arco se lo toma como cero. Además para facilitar el proceso de solución del problema es conveniente transformar las capacidades mínimas diferentes de cero en cero.

Los flujos externos que entran o salen en los nodos, conectan con la parte exterior del sistema. Hay dos tipos de flujo externos, el flujo fijo y el flujo de holgura. El flujo fijo b_i, entra en el nodo i (oferta) si $b_i > 0$ y sale del nodo i (demanda) si $b_i < 0$. El flujo de holgura f_{si} en el nodo i es determinado como parte del procedimiento de optimización y su valor b_{si} es la capacidad de holgura del flujo externo. El flujo de holgura b_{si}, entra en el nodo i, si $b_{si} > 0$ y sale del nodo i si $b_{si} < 0$. Considerando estos nuevos parámetros, el nuevo costo en la red será:

$$H(f) = \sum_{k=1}^{m} h_k f_k + \sum_{i=1}^{n} h_{si} f_{si}$$

Para representar en mejor forma una red es conveniente usar parámetros en los arcos, en lugar de los parámetros en los nodos. En esta nueva representación, se crea un nodo de holgura (auxiliar), que no requiere conservación de flujo y es designado con el número siguiente al nodo con mayor numeración. Cuando el flujo de holgura $f_{si} > 0$, se construye un arco que va desde el nodo de holgura al nodo i, y cuando $f_{si} < 0$ se construye un arco que va desde el nodo i al nodo de holgura. La capacidad del nuevo

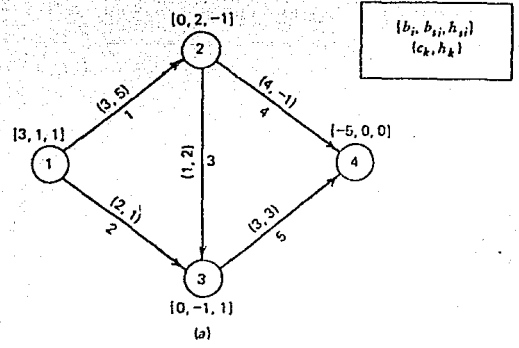
arco es $|b_{si}|$ y su costo es h_{si}. Para su mejor entendimiento se ilustra un ejemplo en la figura 2.5

Un flujo factible f satisface la conservación de flujo en todos los nodos de la red, excepto en el nodo de holgura. En tal forma que la conservación de flujo en cada nodo esta dada por:

Flujo total de los arcos que salen del nodo-flujo total - de los arcos que entran en el nodo=Flujo externo fijo en el nodo.

Matemáticamente, la conservación de flujo, en general, para un nodo i es:

$$\sum_{k \epsilon M_{oi}} f_k - \sum_{k \epsilon M_{ti}} f_k = b_i$$



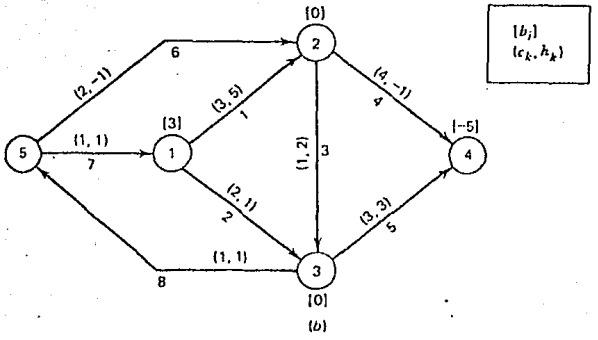


Figura 2.5

Representación de los flujos externos de holgura (a) con parámetros en los nodos (b) con un nodo de holgura

Para el ejemplo de la figura 2.5b, el conjunto de restricciones de conservación de flujo está dado por:

	Arcos								
Nodos	1	2	3	4	5	6	7	8	
1	f_1	$+f_2$					$-f_7$		$= 3$
2	$-f_1$		$+f_3$	$+f_4$		$-f_6$			$= 0$
3		$-f_2$	$-f_3$		$+f_5$			$+f_8$	$= 0$
4				$-f_4$	$-f_5$				$=-5$

Note que las columnas que no representan arcos de holgura tienen exactamente dos elementos diferentes de cero. Cuando la columna representa uno de estos arcos k, al flujo variable f_k - (i,j) apareçe con un coeficiente +1 en la fila i y un coeficiente -1 en la fila j. Los arcos de holgura en cambio tienen solamente un elemento diferente de cero, ya que no hay restricción de conservación de flujo en el nodo de holgura. Si A es la matriz de coeficientes asociada con las ecuaciones de conservación de flujo y b como el vector columna de los flujos externos fijos, las ecuaciones de conservación de flujo pueden escribirse como $Af = b$

La transformación de la capacidad mínima $\underline{c}_k$ a cero, es importante para minimizar el número de parámetros usados al definir una red. Especialmente se reduce el espacio de memoria en el proceso computacional. Este proceso de transformación para cada arco k(i,j), en problemas con capacidades mínimas $\underline{c}_k$ diferentes de cero, se obtiene ejecutando los siguientes reemplazos:

1. $\underline{c}_k$ por cero
2. c_k por $c'_k = c_k - \underline{c}_k$
3. f_k por $f'_k = f_k - \underline{c}_k$
4. b_i por $b'_i = b_i - \underline{c}_k$
5. b_j por $b'_j = b_j + a_k\underline{c}_k$

Cada transformación usa los parámetros actualizados que -

resultaron de la transformación anterior, Se ilustra un ejemplo en la figura 2.6

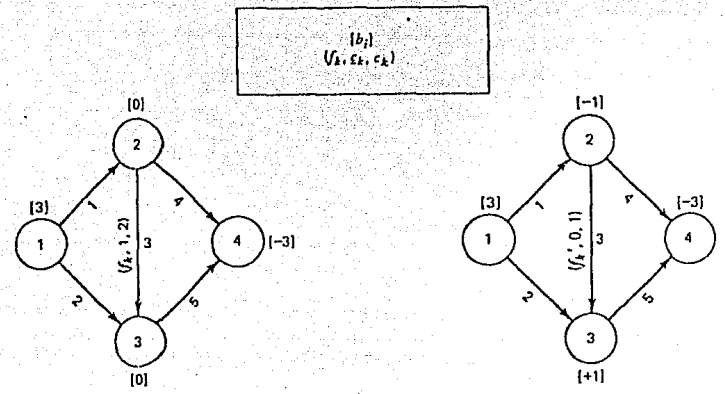


Figura 2.6

La transformación de los parámetros capacidad y costo de un nodo, a parámetros de un arco, se puede hacer fácilmente reemplazando tal nodo, por dos nodos conectados mediante un arco, como se puede ver en la figura 2.7

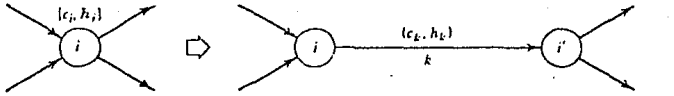


Figura 2.7

2.7 MODELO MATEMATICO DE REDES

Para definir un modelo de redes de flujo es suficiente la representación gráfica junto con los parámetros de los nodos y los arcos. La representación matemática es importante para el

desarrollo teórico. Las características del nodo de holgura, la función objetivo y la conservación de flujo están implícitas en la definición. Es una ventaja poder expresar el problema simple de redes de flujo a costo mínimo, como un problema de programación lineal (PL), en la siguiente forma:

$$\text{Min.} \quad \sum_{k=1}^{m} h_k f_k$$

$$\text{s.a.} \quad \sum_{k \in M_{oi}} f_k - \sum_{k \in M_{ti}} f_k = b_i \qquad i=1,\ldots,n-1$$

$$f_k \leq c_k \qquad k=1,\ldots,m$$

$$f_k \geq 0 \qquad k=1,\ldots,m$$

donde n es el nodo de holgura.
La notación matricial del problema de programación lineal es:

$$\text{Min.} \quad h_f \qquad (1)$$

$$\text{s.a.} \quad Af = b \qquad (2)$$

$$f \leq c \qquad (3)$$

$$f \geq 0 \qquad (4)$$

Se resumen algunos resultados importantes de la teoría de programación lineal.

Considere la partición de A tal que A=(B,R). Siendo B una matriz formada por un conjunto de n-1 columnas linealmente independientes de A, y es denominada una base; las variables correspondientes a estas columnas son las variables básicas. El vector de variables básicas implicitamente escrito en el mismo orden de las columnnas en B, es f_B, y el conjunto de variables no básicas asociadas con R es f_R. Con estas consideraciones el problema de programación lineal (PL), se escribe de la siguiente manera.

$$[B,R] \begin{bmatrix} f_B \\ f_R \end{bmatrix} = b \qquad (5)$$

Resolviendo para f_B se tiene

$$f_B = B^{-1} [b - Rf_R] \qquad (6)$$

Para definir una solución básica, primero se asigna a cada variable no básica el valor de cero o C_k, y luego la ecuación (6) es la solución básica f_B.

El dual asociado es:

$$\text{Min.} \quad \sum_{i=1}^{n-1} \Pi_i b_i + \sum_{k=1}^{m} \delta_k C_k$$

$$\text{s.a.} \quad \Pi_i - \Pi_j + \delta_k \geq -h_k \qquad \begin{matrix} k=1,\ldots,m \\ i=0(k),\ j=t(k) \end{matrix}$$

$$\Pi_i \text{ irrestricto} \qquad i=1,\ldots,n-1$$

$$\delta_k \geq 0$$

Su respectiva notación matricial es:

$$\text{Min.} \quad \Pi b + \delta_c \qquad (7)$$

$$\text{s.a.} \quad \Pi A + \delta I \geq -h \qquad (8)$$

$$\Pi \text{ irrestricto} \qquad (9)$$

$$\delta \geq 0 \qquad (10)$$

Las condiciones primal-dual para obtener la solución óptima del PL han sido resumidas para el problema de redes de flujo a costo mínimo, en los siguientes tres teoremas.

TEOREMA 1

Dada una solución $f=(f_B, f_R)^T$ para el problema primal y una solución (Π, δ) para el problema dual, las soluciones son óptimas si y solo si:

1. f es factible para el problema primal; esto es, se satisfacen (2), (3) y (4)
2. (Π, δ) es factible para el problema dual, esto es se satisfacen (8) y (10)
3. Se satisface la holgura complementaria, esto es,

(a) Si $f_k(i,j)>0$, $\Pi i - \Pi j + \delta_k = -h_k$

(b) Si $f_k < c_k$, $\delta_k = 0$.

(c) Si $\Pi i - \Pi j + \delta_k > -h_k$, $f_k(i,j)=0$

(d) Si $\delta_k > 0$, $f_k = c_k$

TEOREMA 2

Si (Π,δ) es una solución óptima del problema dual, entonces δ_k = máx $(0, -h_k-\Pi_i + \Pi_j)$.

El Teorema 2 permite que la optimalidad para el problema sea escrito como sigue:

TEOREMA 3

Dada una solución f para el problema primal y la solución parcial Π para el problema dual, las soluciones son óptimas para sus respectivos problemas si y solo si, se satisfacen las siguientes consideraciones:

1. Factibilidad primal
2. δ_k= Máx $(0, -h_k - \Pi_i + \Pi_j)$ (factibilidad restringida del dual)
3. Holgura complementaria.

(a) $\Pi_i - \Pi_j = -h_k$ para $0 < f_k < c_k$.

(b) $f_k = 0$ para $\Pi_i - \Pi_j > -h_k$

(c) $f_k = c_k$ para $\Pi_i - \Pi_j < -h_k$ donde i=0(k) y j=t(k)

2.8 REDES EXPANDIDA Y MARGINAL

Las redes expandida y marginal son definiciones que se utilizarán con mucha frecuencia a lo largo de este trabajo.

La red expandida es obtenida directamente desde la red original, empleando la definición del arco reflejado, -k, que simplemente es el mismo arco hacia adelante, pero con dirección opuesta. Luego la red expandida, D_E= (N,M_E), tiene el mismo conjunto de nodos que la red original, pero el conjunto de arcos M_E contiene los arcos originales y los arcos reflejados, en tal forma que M_E= $(1,2,\ldots,m,-1,-2,\ldots,-m)$. Como se puede observar en la figura 2.8

Se debe notar que si la red original es conectada, existe una trayectoria dirigida entre cada par de nodos en la red expandida. Además el arco reflejado de un arco reflejado es un arco - hacia adelante.

La red marginal, $D_M = (N, M_M)$, tiene el mismo conjunto de nodos que la red original, pero el conjunto de arcos M_M que es un subconjunto de M_E contiene solo arcos admisibles. Considerándose que un arco hacia adelante es admisible si su flujo no alcanzado su capacidad máxima, esto es si es posible incrementar el flujo en la red original, Mientras un arco reflejado es admisible si - el flujo en el correspondiente arco hacia adelante es igual a cero (esta en su limite inferior); esto es si es posible disminuir el flujo en su correspondiente arco asociado de la red original.

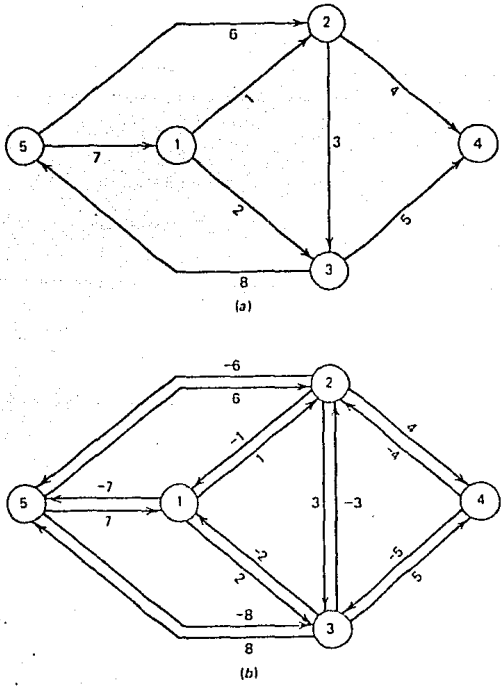


Figura 2.8
(a) Red dirigida original (b) Red expandida.

Note que el costo h_k en el arco reflejado es $-h_k$. La figura 2.9 ilustra un ejemplo de la red original y su correspondiente red marginal asociada.

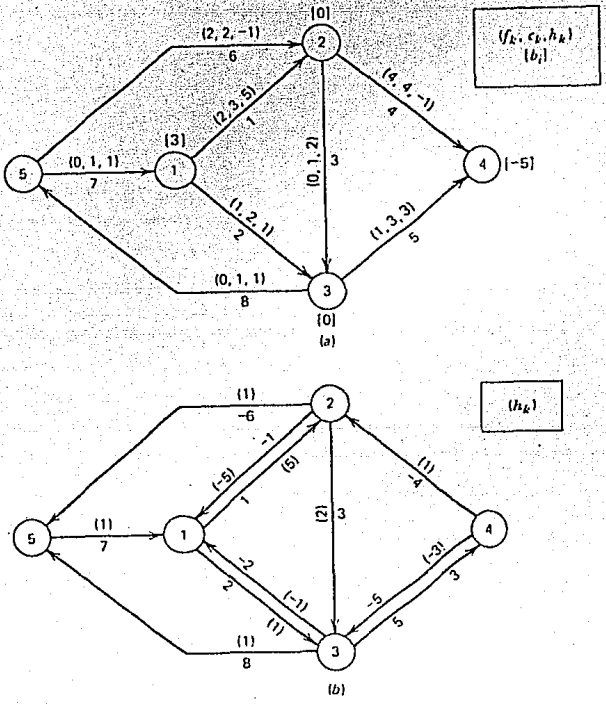


Figura 2.9

(a) Red original (b) Red marginal asociada.

CAPITULO III

ALGORITMOS BASICOS Y SU MANIPULACION.

3.1 INTRODUCCION.

En este capítulo se consideran en forma general los algoritmos que ayudan a resolver los problemas de redes de flujo.

Estos algoritmos se usan después en los capítulos siguientes en la solución de los problemas propuestos, los cuales almacenan, modifican y manipulan las representaciones de redes y subredes.

3.2 REPRESENTACION DE REDES.

La forma de representar una red es importante al considerar el tiempo de proceso y el espacio para el almacenamiento de los datos en una computadora. Se puede usar la representación del arco-orientado o la representación del nodo-orientado.

La representación de un arco-orientado se realiza mediante las listas de arcos $O = [o_k]$ y $T = [t_k]$, donde O es el nodo origen y T es el nodo terminal de un arco k. Sus parámetros se representan en forma similar $f = [f_k]$, $c = [c_k]$ y $h = [h_k]$, - que significan el flujo, la capacidad y el costo del arco k, -- respectivamente. Como se puede observar en la figura 3.1.

Arc (k)	1	2	3	4	5	6	7
o_k	2	5	1	1	3	2	5
t_k	4	2	3	2	4	3	1
f_k	2	1	3	1	3	0	1
c_k	2	1	3	3	5	1	2
h_k	−1	−1	1	5	3	2	1

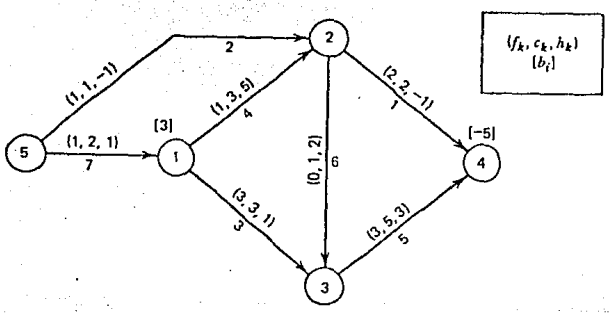


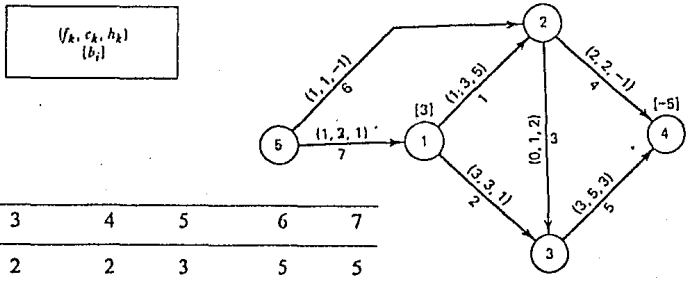
Figura 3.1
La red y su correspondiente lista de arcos.

La representación de nodo-orientado se realiza mediante la matriz origen-terminal, en la cual se almacenan las variables y parámetros tal que la entrada en la fila i, columna j, corresponde al arco (i,j). Esta representación incluye las matrices F = [f(i,j)] y C = [c(i,j)] , como se muestra en la figura 3.2. Este tipo de representación generalmente se emplea en problemas de transporte, donde sus matrices son pequeñas y con un alto --

Origen \ Terminal	1	2	3	4	5
1	--	1	3	--	--
2	--	--	0	2	--
3	--	--	--	3	--
4	--	--	--	--	--
5	1	1	--	--	--

Figura 3.2

grado de densidad. Es necesario realizar una primera modificación para almacenar la lista de arcos, ordenando de acuerdo al crecimiento del nodo origen. En la figura 3.3 se indican los resultados de esta modificación luego de reordenar la numeración de los arcos de la figura 3.1.



Arc (k)	1	2	3	4	5	6	7
o_k	1	1	2	2	3	5	5
t_k	2	3	3	4	4	2	1
f_k	1	3	0	2	3	1	1
c_k	3	3	1	2	5	1	2
h_k	5	1	2	−1	3	−1	1

Figura 3.3
La red y su correspondiente lista de arcos reordenada.

Además se manejarán ciertas listas de apuntadores tales como la lista $Po = [p_o(i)]$, la cual contiene el arco con el menor número índice que se origina en al nodo i. Si no se originan arcos en el nodo i, se hace $p_o(i) = P_o(i + 1)$. Esta lista de arcos es ordenada en tal forma que:

$$o(k_1) < o(k_2) \quad \text{si } k_1 < k_2$$
$$o(k_1) = o(k_2) \quad \text{si el orden de } k_1 \text{ y } k_2 \text{ es arbitrario.} \qquad (1)$$

Posteriormente, para $i \leq m$.

$$p_o(1) = 1$$
$$p_o(i) = \{k \mid o(k) \geq i,\ o(k-1) < i\} \quad 1 < i \leq n \qquad (2)$$

y $$p_o(n+1) = m + 1$$

El conjunto de arcos originados en el nodo i, Mo_i, es

$$Mo_i = \{k \mid p_o(i) \leq k < p_o(i+1)\} \qquad (3)$$

Aplicando estos conceptos a la red de la figura 3.3 tenemos la siguiente lista del apuntador origen P_o.

Nodo	1	2	3	4	5	6
p_o	1	3	5	6	6	8

La lista auxiliar $L_T = \{ l_T(i) \}$, la cual ordena los arcos de acuerdo al crecimiento del nodo terminal. Asi, si K'_w es el índice del arco K_w en L_T, la lista de arcos es ordenada tal que:

$$t(k_w) < t(k_y) \quad \text{si } k'_w < k'_y$$
$$t(k_w) = t(k_y) \quad \text{si el orden de } k_w \text{ y } k_y \text{ es arbitrario.} \qquad (4)$$

La lista $p_T = \{p_T(i)\}$, la que contiene el número del arco con el menor valor que termina en el nodo i, pero en base a la lista L_T. En la siguiente forma:

$$p_T(1) = 1$$

$$\text{para } i \leqslant n, \quad p_T(i) = \{k' \mid t(l_T(k') \geqslant i,\ t(l_T(k'-1)) < i\} \quad (5)$$

$$\text{y} \qquad p_T(n+1) = m+1$$

El conjunto de arcos que terminan en el nodo i, M_{T_i}, es

$$M_{T_i} = \{l_T(k') \mid p_T(i) \leqslant k' < p_T(i+1)\} \quad (6)$$

Aplicando otros conceptos a la red de la figura 3.3, tenemos la lista auxiliar L_T y el apuntador terminal p_T

k'	1	2	3	4	5	6	7
L_T	7	1	6	2	3	4	5

Nodo	1	2	3	4	5	6
p_T	1	2	4	6	8	8

3.3 LECTURA Y ALMACENAMIENTO DE LOS DATOS DE LA RED.

Es importante la lectura y el almacenamiento de los datos, al iniciar el proceso de cálculo en una computadora. Primeramente lee los datos de cada uno de los nodos que son: el número del nodo, el flujo externo fijo, la capacidad del flujo de holgura externo y el costo del flujo de holgura externo. Luego se deja una fila en blanco y se continúa con los datos de los arcos, los cuales son leídos por cada uno de los arcos en secuencia y estos son: el nodo origen, nodo terminal, capacidad míni-

ma, capacidad máxima y costo. Luego se deja una fila en blanco y se puede continuar con otros datos que serán necesarios.

La entrada de datos puede ser diferente para cada tipo de problema pero la lógica de la entrada en forma general se mantiene. En la figura 3.4 se puede observar las listas de datos de los nodos y de los arcos, correspondientes a la red de la figura 3.1.

Datos de los nodos:

Nodo	Flujo externo	Capacidad de holgura	costo de holgura.
1	3	2	1
2	0	1	-1
3	0	0	0
4	-5	0	0

Datos de los arcos:

Nodo origen	Nodo terminal	Capacidad inferior	Capacidad superior	costo
1	2	0	3	5
1	3	0	3	1
2	3	0	1	2
2	4	0	2	-1
3	4	0	5	3

Figura 3.4
Listas de datos de entrada.

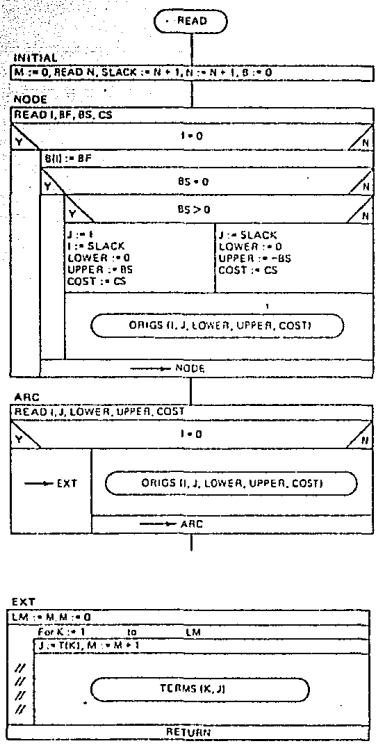
3.4 ALGORITMOS BASICOS.

En esta sección se presentan los algoritmos básicos que se utilizan para la solución de los tres problemas mencionados en

el capítulo I, sección 1.1. Estos algoritmos son: READ lee los datos de la red y usa el nodo de holgura para modificar la red, teniendo solamente flujos externos fijos; ORIGS ordena todos los arcos de acuerdo al incremento del nodo origen, determina la lista de apuntadores P_o, y transforma la capacidad mínima si se requiere; ORIG encuentra la lista de arcos originados en el nodo I (Mo_i); TERMS elabora la lista auxiliar L_T de los arcos ordenados de acuerdo al incremento del nodo terminal, y la lista de los apuntadores p_T; TERM encuentra la lista de arcos que terminan en el nodo I (M_{T_I}), en base a la lista auxiliar L_T.

ALGORITMO READ.

Propósito: leer y almacenar los datos de los nodos y los arcos para el problema de flujo a costo mínimo.



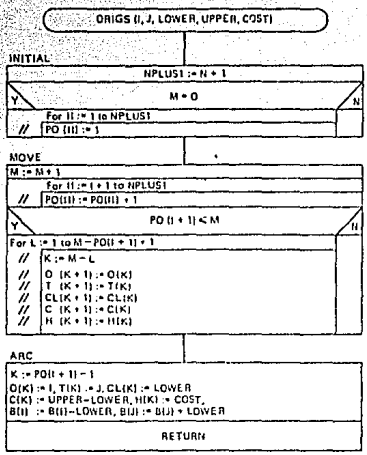
1. (INICIAR) Inicializa el número del arco a cero. Lee el número de nodos, crea el nodo de holgura SLACK : = N + 1, N : = N + 1, fija todos los flujos externos a cero, $B_I = 0$.
2. (NODO) Lee los datos del nodo I (flujo externo B, capacidad superior BS y el costo de holgura CS). Si el renglón de datos está en blanco, va al paso 3. De lo contrario almacena los flujos fijos. Si el flujo externo es cero, repite el paso 2. De otra manera crea un arco de holgura y almacena los datos del arco en la posición correcta en la lista de arcos ORIGS (I,J,LOWER, UPPER, COST).
3. (ARCO) Lee los datos del arco (I,J) (la capacidad mínima LOWER, la capacidad máxima UPPER y el costo por unidad de flujo COST). Si el renglón de datos está en blanco, va al paso

4. De lo contrario almacena los datos del arco en la posición correcta en la lista de arcos ORIGS (I, J, LOWER, UPPER, COST). Repite el paso 3.

4. (EXT) Coloca cada dato de los arcos en la posición correcta en la lista terminal correspondiente, TERMS (K,J).

ALGORITMO ORIGS.

Propósito: Acepta el conjunto de datos correspondientes a un arco y los almacena en una lista ordenada incrementando el índice del nodo origen.



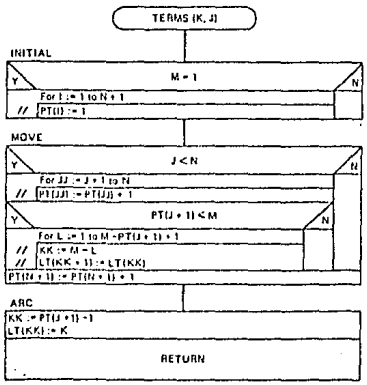
1. (INICIAR) Si primero llama a ORIGS, asigna al apuntador PO sobre todos los nodos, el valor de uno, o sea PO : = 1. De lo contrario, va al paso 2.

2. (MOVER) Incrementa el número de arcos M en 1, o sea M: = M+1. Incrementa el apuntador PO sobre todos los nodos mayores que I, en 1. Mueve todos los arcos sobre la nueva entrada a un índice mayor en la lista. k + 1: = k.

3. (ARCO) Inserta un arco en la última posición asignada al nodo I. Modifica la capacidad máxima del arco y el flujo externo fijo considerando la capacidad mínima del arco.

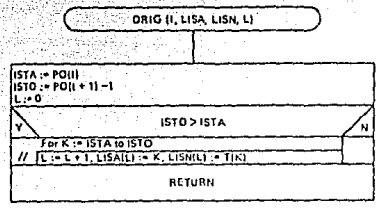
ALGORITMO TERMS.

Propósito: Elabora la lista LT de los índices de los arcos en orden de acuerdo al incremento del nodo terminal. También produce la lista de apuntadores PT hacia los nodos terminales, tal que la lista LT puede ser referenciada rápidamente. Este algoritmo es llamado una vez por cada arco en la red.



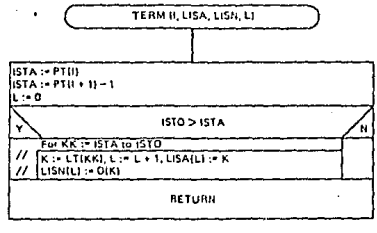
1. (INICIAR) La primera vez que este algoritmo es llamado, asigna al apuntador PT sobre todos los nodos terminales el valor de 1.
2. (MOVER) Incrementa el valor del apuntador PT en 1 para todos aquellos nodos con índice mayor que J. Mueve todos los arcos referenciados con el nodo terminal mayor que J, un índice mayor en la lista.
3. (ARCO) Inserta la nueva entrada en la lista en la última posición permitida al nodo J.

ALGORITMO ORIG.



Propósito: Determina la lista de arcos LISA originados en el nodo I --- (Mo_i) y la lista de sus respectivos nodos terminales LISN. Encuentra -- los apuntadores P_o al inicio con -- ISTA y al final con ISTO, de los arcos originados en el nodo I. Si no hay tales arcos termina. De otra manera almacena los arcos originados en I en la lista de arcos LISA. Encuentra el nodo terminal de cada uno de estos arcos y los almacena en una lista de nodos LISN.

ALGORITMO TERM

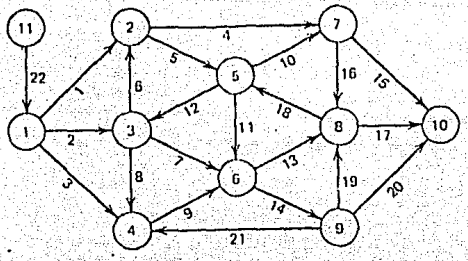


Propósito: Determina la lista de arcos LISA que terminan en el nodo -- (M_{T_i}) y la lista de sus respectivos nodos origen LISN. Encuentra los apuntadores PT al inicio con ISTA y - al final con ISTO, de los arcos que terminan en el nodo I. Si no hay tales arcos termina. De otra manera encuentra estos arcos en la lista auxiliar LT y los almacena en la lista de arcos LISA. Encuentra el nodo origen de cada uno de estos arcos y los almacena en una lista de nodos LISN.

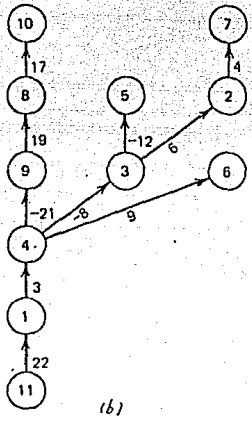
3.5 CONSTRUCCION DEL ARBOL.

En esta sección se presentan los algoritmos para la construcción de un árbol dirigido. TPATH encuentra la única ruta desde i a j; ROOT encuentra el subárbol enraizado a un nodo dado; DELTRE quita un arco desde un árbol básico; ADDTRE adiciona un arco a un bosque; TRECHG cambia de base, y TREINT inicializa un árbol. Varios de estos algoritmos usan una base (el árbol dirigido enraizado en el nodo de holgura), la misma que debe estar almacenada en tal forma que sea fácil de localizarla y modificarla. En la figura 3.5 se puede ver la red y sus correspondientes dos árboles expandidos. Como se puede notar cada árbol representa una única ruta desde el nodo raiz a cada uno de los nodos donde termina. Los arcos que tienen signo negativo están invertidos, con la finalidad de que todos los arcos del árbol estén orientados hacia arriba (árbol dirigido). El árbol $D_T = [N_T, M_T]$, es una subred del árbol expandido y puede ser representado usando algunas etiquetas, que son los métodos más usados. Se considera primeramente el método de la triple etiqueta, en el cual, cada nodo tiene tres etiquetas. Estas etiquetas para el nodo i son el apuntador hacia atrás, $p_B(i)$, el apuntador hacia adelante, $p_F(i)$, y el apuntador hacia la derecha $p_R(i)$. El apuntador hacia atrás es el único arco que termina en el nodo i. El apuntador hacia adelante es el nodo terminal más a la izquierda de un arco que se origina en el nodo i. El apuntador hacia la derecha es el nodo que aparece directamente a la derecha y a la misma altura del nodo i. $p_D(i)$ indica el nivel al que se encuentran cada uno de los nodos y $p_P(i)$ es el "preordenador transversal" etiqueta para cada nodo. p_p será utilizado cuando se discuta un segundo método de representación del árbol. Note que a cada nodo sólo se le puede asignar un sólo apuntador hacia atrás p_B, pero la asignación de p_F y p_R no necesariamente es única.

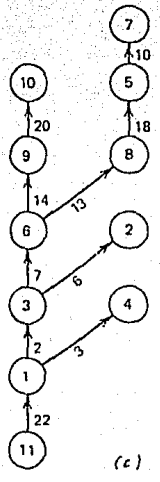
Dado un par de nodos i y j, existe una única ruta desde i a j definida por los arcos del árbol. Una ruta puede tener un nodo unión ℓ y además una ruta hacia adelante o una ruta inver-



(a)



(b)



(c)

Nodo	1	2	3	4	5	6	7	8	9	10	11
P_B	22	6	−8	3	−12	9	4	19	−21	17	0
P_F	4	7	5	9	0	0	0	10	8	0	1
P_R	0	0	6	0	2	0	0	0	3	0	0
P_D	1	4	3	2	4	3	5	4	3	5	0
P_P	4	7	5	9	2	11	6	10	8	3	1

(d)

Nodo	1	2	3	4	5	6	7	8	9	10	11
P_B	22	6	2	3	18	7	10	13	14	20	0
P_F	3	0	6	0	7	9	0	5	10	0	1
P_R	0	0	4	0	0	2	0	0	8	0	0
P_D	1	3	2	2	5	3	6	4	4	5	0
P_P	3	4	6	11	7	9	2	5	10	8	1

(e)

Figura 3.5
(a) Red, (b) y (c) árboles, (d) y (e) apuntadores de (b) y (c), respectivamente.

sa o ambas.

El nodo unión es el último nodo común en las dos únicas rutas desde los nodos raíz i y j, respectivamente. Si i está en la ruta desde la raiz a j, el nodo unión es el nodo i y no hay ruta inversa. Si j está en la ruta desde la raiz al nodo i,j es el nodo unión y no hay ruta hacia adelante. De otra manera, la ruta inversa procede desde el nodo i al nodo ℓ atravesando arcos en la dirección inversa a su orientación en el árbol, mientras la ruta hacia adelante procede desde el nodo ℓ al nodo j siguiendo los arcos en la misma dirección a su orientación en el árbol.

Sean Mrj y Mri los conjuntos de arcos para la ruta desde el nodo raiz a j e i, respectivamente. El conjunto de arcos común a las dos rutas es Mrℓ. Así los arcos en la ruta hacia adelante son

$$M_F = Mrj - Mr\ell \tag{7}$$

y los arcos en la ruta inversa son

$$M_R = Mri - Mr\ell \tag{8}$$

Si a $M_{\bar{R}}$ se le define como el conjunto de arcos hacia atrás de M_R, se puede escribir el conjunto de arcos en la ruta desde i a j como

$$Mij = M_F \cup M_{\bar{R}} \tag{9}$$

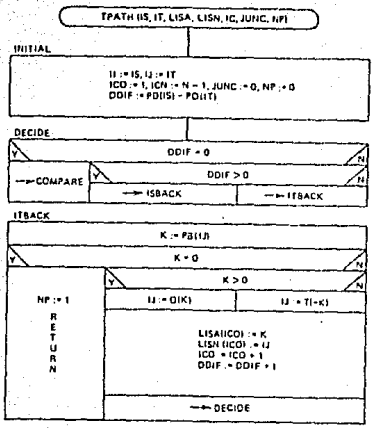
Conceptualmente simple, pero computacionalmente no conviene. El método de determinar Mij puede ser: 1ro. usar los apuntadores hacia atrás desde el nodo j para encontrar Mrj, y 2do., los apuntadores hacia atrás desde el nodo i pueden ser trazados hasta el primer nodo común a Mrj, el nodo ℓ, es encontrado. -- Las ecuaciones (7), (8) y (9) pueden ser aplicadas directamente.

En redes pequeñas no es conveniente usar este método.

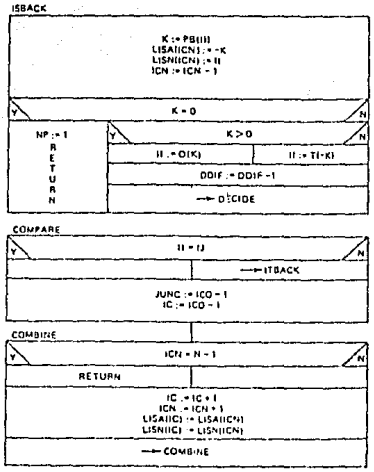
Este problema puede facilitarse asociando una etiqueta adicional $p_D(i)$ con cada nodo. El algoritmo TPATH implementa los conceptos anteriores.

ALGORITMO TPATH.

Propósito: Encuentra la ruta en el árbol desde el nodo (IS) al nodo (IT). Se proporciona la lista de arcos en la ruta (LISA), la lista de nodos (LISN), el número de nodos en la ruta (IC), el nodo unión (JUNC) entre arcos inversos y arcos hacia adelante que forman la ruta. Los arcos y nodos se listan de (IT) a (IS). Si NP = 1, no existe ruta desde IS a IT.



1. (INICIAR) Inicializa las variables y calcula las diferencias en las alturas (niveles) de los nodos IS e IT.
2. (DECIDIR) Decide si el rastro adicional hacia atrás es necesario para igualar alturas en los nodos. Si no va a COMPARE. De otra manera determina cuál tiene mayor altura y realiza una iteración hacia atrás a partir de este nodo.
3. (BAJAR-IT) Ejecuta una iteración hacia atrás en la ruta hacia atrás del nodo IT. Guarda el arco y el nodo encontrado. Si el nodo al que se llega no tiene apuntador hacia atrás, no existe trayectoria entre IS a IT.
4. (BAJAR-IS). Ejecuta una iteración hacia atrás en la trayectoria hacia atrás del nodo IS. Guarda el arco y el nodo encontrado. Si el nodo al que se llega no tiene apuntador hacia



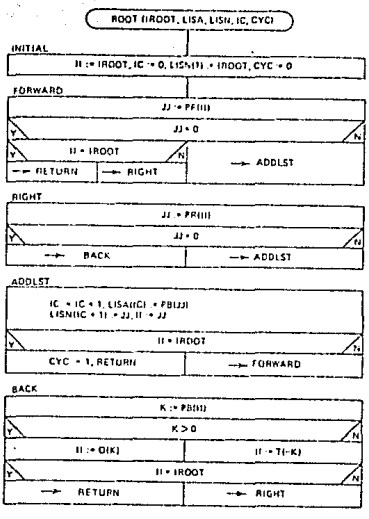
atrás, no existe trayectoria entre IS e IT.

5. (COMPARAR). Ya que los nodos visitados en el momento tienen la misma altura, bien puede ser el mismo nodo. De otra manera ir a BAJAR-IT.
6. (COMBINAR) Une la lista de nodos y arcos guardados en las dos rutas hacia atrás en una sola lista.

Para encontrar un árbol enraizado a un nodo dado se usan las tres etiquetas. Se inicia dando un árbol y uno de sus nodos. Hay un único subárbol en el árbol enraizado a un nodo dado. La subrutina ROOT encuentra los arcos y nodos que forman este subárbol.

ALGORITMO ROOT.

Propósito: Encuentra la lista de arcos (LISA) y la lista de nodos (LISN) que están en el árbol dirigido enraizado en el nodo IROOT. Si los apuntadores detectan un ciclo hace CYC=1.



1. (INICIAR) Hace II = IROOT, almacena IROOT en LISN.
2. (ADELANTE) Si el nodo II tiene su apuntador hacia adelante distinto de cero, va al paso 4. De lo contrario, si el nodo II no es igual a IROOT, va al paso 3. El subárbol consiste de un sólo nodo, -- IROOT, retorna.
3. (DERECHA) Si el nodo II no tiene apuntador hacia la derecha, hace un rastreo hacia atrás; va al paso 5. De otra manera va al paso 4.
4. (SUMA LST) Almacena el arco y el nodo encontrados en la última búsqueda. Si el nodo II es igual a IROOT existe un ciclo que regresa a IROOT. De otra manera va al paso 2.

5. (ATRAS) Rastrea hacia atrás del nodo II. Si el nuevo nodo no es igual a IROOT, va al paso 3.

Para quitar un arco desde la base del árbol se usan los apuntadores de las tres etiquetas. La salida de un arco k_L (i_L, j_L) desde la base de un árbol es complejo, debido a la modificación de los apuntadores para la representación del árbol. El arco que sale del árbol puede presentarse en dos formas:

1. Como arco apuntando hacia delante, $p_B(j_L)=k_L$ y $p_F(i_L)=j_L$

2. Como arco apuntando hacia la derecha $p_B(j_L)=k_L$, $p_R(\ell)=j_L$

En la figura 3.5b, los arcos -21 y -8 son los que apuntan hacia adelante y hacia la derecha respectivamente.

En uno u otro caso, el nodo j_L se vuelve la raíz de un subárbol cuando el arco $k_L(i_L, j_L)$ sale del árbol original. La estructura de las tres etiquetas para cambiar del árbol original a los subárboles, después quitar el arco en cada uno de los dos casos anteriores:

1. $p_F(i_L) := p_R(j_L)$

2. $p_R(\ell) := p_R(j_L)$

y luego se implementa con $p_B(j_L) = 0$ y $p_R(j_L) = 0$

En la figura 3.6a se ilustra el primer caso, al salir el arco -21(4,9) y en la figura 3.6b se ilustra el segundo caso al salir el arco -8(4,3); a partir de la red original de la figura 3.5b. Además se puede notar que al salir el arco se forman dos subárboles, y cada uno mantiene todas las propiedades de un árbol.

Las operaciones anteriores lo realiza el algoritmo DELTRE.

Node	1	2	3	4	5	6	7	8	9	10	11
P_B	22	6	−8	3	−12	9	4	19	0	17	0
P_F	4	7	5	3	0	0	0	10	8	0	1
P_R	0	0	6	0	2	0	0	0	0	0	0
P_D	1	4	3	2	4	3	5	4	3	5	0
P_P	4	7	5	3	2	11	6	10	8	9	1

Node	1	2	3	4	5	6	7	8	9	10	11
P_B	22	6	0	3	−12	9	4	19	−21	17	0
P_F	4	7	5	9	0	0	0	10	8	0	1
P_R	0	0	0	0	2	0	0	0	6	0	0
P_D	1	3	2	2	5	3	6	4	3	5	0
P_P	4	7	5	9	2	11	3	10	8	6	1

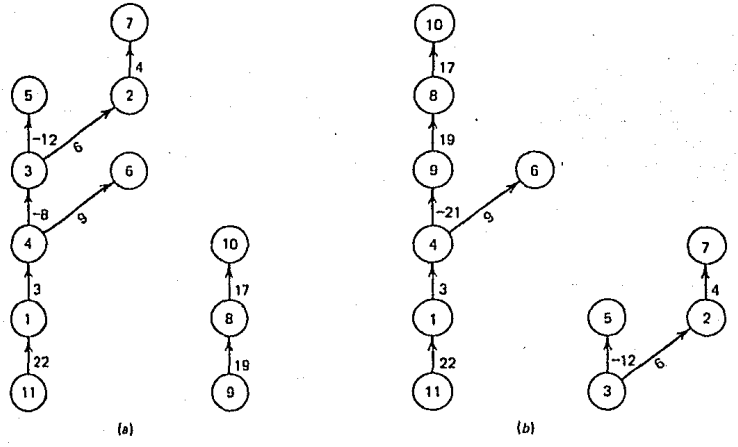
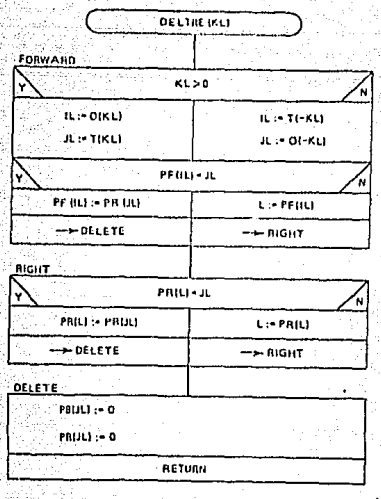


Figura 3.6
(a) Subárboles al salir del arco -21(4.9) del árbol de la figura 3.5b.
(b) Subárboles al salir del arco -8(4.3) del árbol de la figura 3.5b.

ALGORITMO DELTRE.

Propósito: Quita un arco del árbol y actualiza su representación en términos de los apuntadores de las tres etiquetas.



1. (ADELANTE) El arco $k_L(i_L,j_L)$ sale del árbol. Si $p_F(i_L)=j_L$, entonces hace $p_F(i_L):=p_R(j_L)$ y va al paso 3. Si $p_F(i_L) \neq j_L$, va al paso 2.
2. (DERECHA) Encuentra el nodo l para el cual $p_R(l):=j_L$. Hace $p_R(l):=p_R(j_L)$ va al paso 3.
3. (QUITAR) Hace al nodo j_L un nodo raiz; esto es, $p_B(j_L):=0$, $p_R(j_L):=0$.

Para adicionar un arco a un bosque usando las tres etiquetas. Se define primeramente un bosque como dos o más árboles dirigidos, D_{T_1} , D_{T_2} , ... Cuando se añade un arco $k_E(i_E,j_E)$ a un bosque, tal que el nodo i_E se encuentre en un árbol y el nodo j_E sea la raiz del otro, estos dos árboles se conectan y forman uno solo. Para lo cual se realizan dos modificaciones en los apuntadores. En la primera se consideran dos casos:

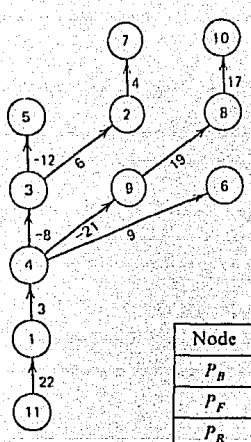
1. Si $p_F(i_E):=0$, entonces $p_F(i_E):=j_E$

2. Si $p_F(i_E):=l$, entonces $p_R(j_E):=l$

En la segunda se hace $p_B(j_E):=k_E$

Esto es posible ya que el nodo j_E es la raiz del árbol.

En la figura 3.7 se puede observar un árbol al añadir el arco -8(4.3) partiendo del árbol de la figura 3.6b. Las operaciones anteriores lo realiza el algoritmo ADDTRE.

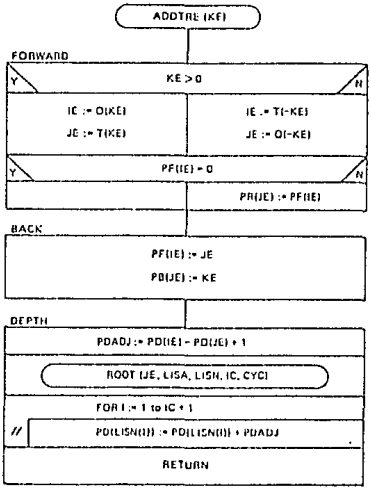


Node	1	2	3	4	5	6	7	8	9	10	11
P_B	22	6	8	3	−12	9	4	19	−21	17	0
P_F	4	7	5	3	0	0	0	10	8	0	1
P_R	0	0	9	0	2	0	0	0	6	0	0
P_D	1	4	3	2	4	3	5	4	3	5	0
P_P	4	7	5	3	2	11	9	10	8	6	1

Figura 3.7
Arco -8(4.3) añadido al bosque de la fig. 3.6b.

ALGORITMO ADDTRE.

Propósito: Añade el arco $k_E(i_E, j_E)$ a un bosque. Los nodos i_E y j_E deberán estar en diferentes árboles y el nodo j_E deberá ser raiz de un árbol.



1. (ADELANTE) Si el apuntador hacia adelante de i_E es cero, va al paso 2. De otra manera, iguala el apuntador a la derecha de j_E con el apuntador hacia adelante de i_E y va al paso 2.
2. (ATRAS) Asigna al apuntador hacia atrás de j_E el valor de k_E y el apuntador hacia adelante de i_E el valor de j_E.
3. (ALTURA) Actualiza las alturas de los nodos en el subárbol -

enraizado en el nodo j_E.

Al cambiar la base cuando un arco $k_L(i_L,j_L)$ sale del árbol básico $D_T = [N,M_T]$, se forman dos conjuntos $D_1 = [N_1,M_1]$ y $D_2 = [N_2,M_2]$ en tal forma que:

$$j_L \in N_2 \qquad i_L \in N_1$$

$$N_1 \cup N_2 = N$$

$$M_1 \cup M_2 = M_T - k_L$$

La red D_1 es un árbol dirigido enraizado en el nodo n. La red D_2 es un árbol dirigido enraizado en el nodo j_L. Al añadir el arco $k_E(i_E,j_E)$; la nueva base del árbol $D_T = [N,M'_T]$ forma un árbol enraizado en el nodo n que se compone de la combinación de D_1, el arco k_E, y D_2. Al añadir D_2 para formar el árbol D'_T, se requiere que algunos de sus arcos cambien el sentido desde j_L hasta j_E. Siendo $D'_2 = [N_2,M'_2]$ la modificación de D_2. La nueva base está dada por: $M'_T = M_1 \cup k_E \cup M'_2$.

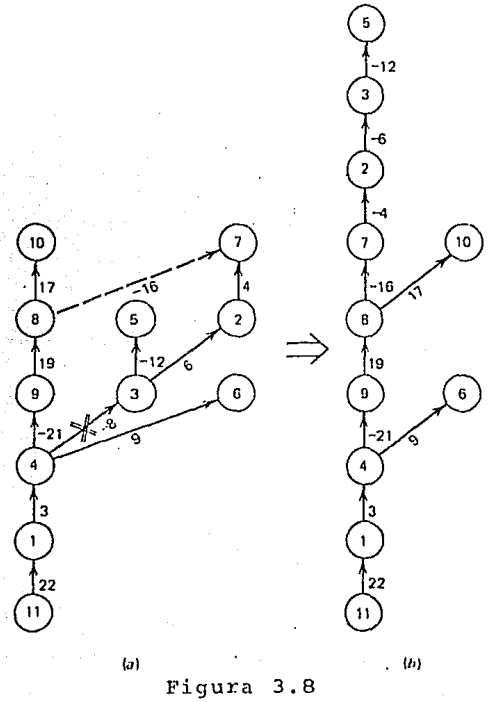
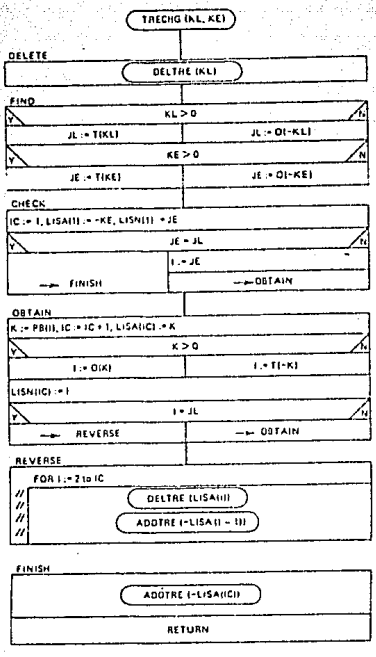


Figura 3.8
(a) Base original. (b) Base luego de sacar el arco -8(4.3) y añadir el arco -16(8,7).

La figura 3.8 ilustra este cambio de base. Al quitar el arco -8(4.3) y añadir el arco -16(8.7). Aquí se puede notar que los arcos 4, 6 cambian su sentido en tal forma que la nueva base sea un árbol dirigido. El cambio de base lo realiza el algoritmo TRECHG, con la ayuda de los algoritmos DELTRE y ADDTRE.

ALGORITMO TRECHG.

Propósito: Quita un arco (k_L) del árbol base, inserta otro arco (k_E) en el árbol base y orienta ciertos arcos en el árbol para mentener el árbol dirigido. El algoritmo asume que el nodo terminal del arco entrante está en N_2.



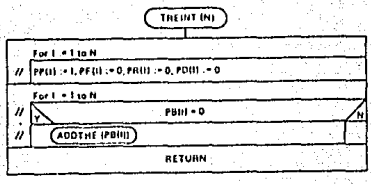
1. (QUITAR) Quita el arco k_L de la base.
2. (ENCONTRAR) Encuentra los nodos terminales para k_E y k_L.
3. (CHEQUEAR) Inicializa el índice IC y la lista de arcos y nodos. Si el nodo terminal de los arcos que entra y sale es el mismo, va al paso 6. De otra manera va al paso 4.
4. (OBTENER) Obtiene la lista de arcos y nodos que permanecen; usando los arcos apuntadores hacia atrás en la ruta en el árbol de j_L a j_E.
5. (INVERTIR) Invierte los arcos en la trayectoria de j_L a j_E, excepto el último arco.
6. (TERMINAR) Suma el inverso del último arco, en la lista de arcos y retorna.

Un algoritmo adicional es usado para inicializar la representación de un árbol por medio de apuntadores, este es el algo

ritmo TREINT. Para lo cual se requiere de la lista de apuntadores hacia atrás (P_B) del árbol. Para la inicialización de los apuntadores P_p, P_F, P_R y P_D; TREINT LLAMA a ADDTRE para cada arco P_B.

ALGORITMO TREINT.

Propósito: Representa un árbol por medio de apuntadores, conociendo previamente la lista de apuntadores hacia atrás P_B.



1. Inicializa las listas P_p, P_F, P_R y P_D.
2. Llama a ADDTRE para cada arco apuntador hacia atrás y retorna.

3.6 CAMBIOS DE FLUJO.

Muchos de los algoritmos descritos requieren cambiar el flujo en la ruta o ciclo de la red original. La trayectoria (o ciclo) es descrita por los índices positivos o negativos de los arcos. Por ejemplo, en la figura 3.9, un ciclo está formado por el conjunto de arcos $M_p = (1, 3, -2)$. Cambiando el flujo en este ciclo por una cantidad Δ correspondiente al incremento del flujo en los arcos con índices positivos y una misma cantidad Δ correspondiente al decremento del flujo en los arcos con índices negativos. Si f_k y f'_k son los flujos en el arco k antes y después del cambio de flujo, respectivamente, tenemos:

$$f'_k = f_k + \Delta \quad \text{para} \quad k \in M_p \quad \text{y} \quad k > 0$$

$$f'_{-k} = f_{-k} - \Delta \quad \text{para} \quad k \in M_p \quad \text{y} \quad k < 0$$

Esta operación es realizada por el algoritmo FLOCHG.

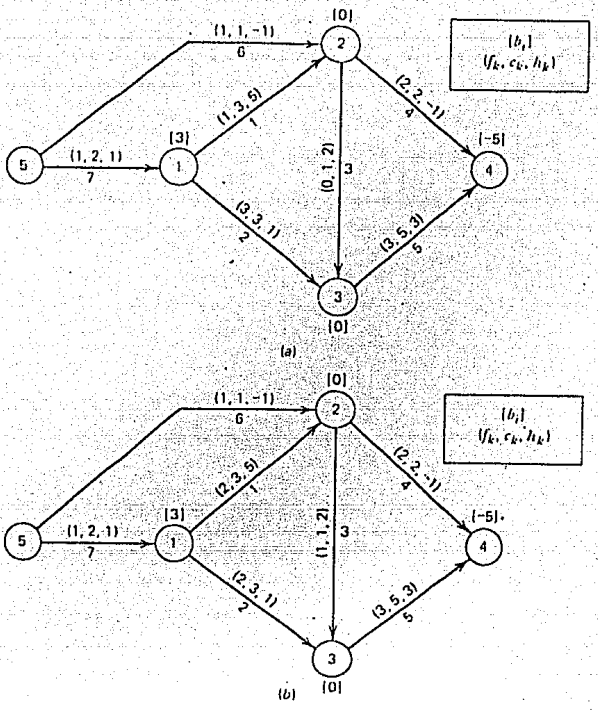
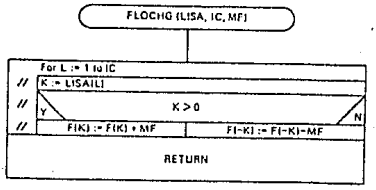


Figura 3.9
(a) Flujos antes del cambio. (b) Flujos des-- pués del cambio en el ciclo, $M_p=(1,3,-2)$, por $\Delta=1$.

ALGORITMO FLOCHG.

Propósito: Cambiar los flujos en una trayectoria (LISA). Para cada arco en la trayectoria, si $k>0$, incrementa el flujo en el arco k por MF, si $k<0$, decrementa el flujo en el arco -k en MF.



Frecuentemente después del cambio de flujo anterior, se -- procede a determinar el máximo cambio de flujo en los arcos, en forma secuencial. El máximo cambio de flujo es el máximo valor de Δ, tal que, los nuevos flujos sean factibles o:

$$f'_k = f_k + \Delta \leq C_k \quad \text{para} \quad k \in M_p \quad y \quad k>0$$

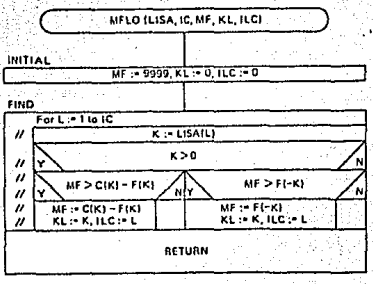
$$f'_{-k} = f_{-k} - \Delta \geq 0 \quad \text{para} \quad k \in M_p \quad y \quad k<0$$

luego $$\Delta_m = \text{Min.} \left[\underset{k>0}{\text{Min.}} \; C_k - f_k \, , \; \underset{k<0}{\text{Min.}} \; f_{-k} \right]$$

Esta operación es realizada por el algoritmo MFLO.

ALGORITMO MFLO.

Propósito: Determinar el máximo cambio de flujo (MF) en una trayectoria (LISA). El algoritmo también determina el arco k_L para el cual se satura su capacidad, guardando esta información en la variable ILC.



1. (INICIAR) Sea MF = R (Siendo R un número grande).
2. (ENCONTRAR) Va a través de la lista de arcos. Encuentra el arco k_L para el que se obtiene el mínimo

$$M_F = \text{Min.}\ [\underset{k>0}{\text{Min.}}\ C_k - f_k\ ,\ \underset{k<0}{\text{Min.}}\ f_{-k}].$$

Para ilustrar en forma más objetiva lo que hacen los algoritmos: READ, ORIGS, TERMS, TERM, TREINT, ADDTRE, ROOT, DELTRE y TRECHG; con los datos de la red de la figura 3.10 se obtienen algunos resultados que imprime el programa RFCLSG (ver anexo A-2). Estos resultados son:

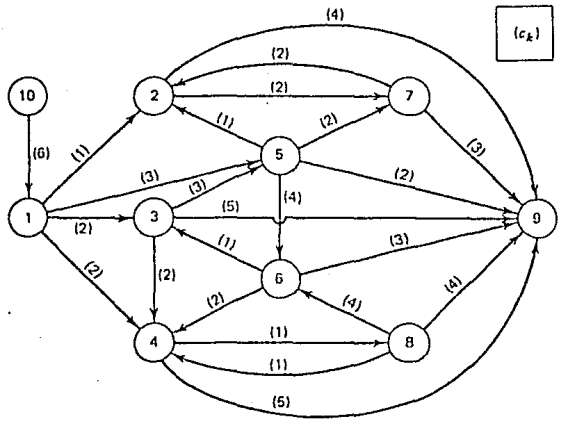


Figura 3.10

```
100     10
200      1
300      2
400      3
500      4
600      5
700      6
800      7
900      8
1000      9
1100     10
1200
1300      1     2                    1
1400      1     5                    3
1500      1     3                    2
1600      1     4                    2
1700      2     9                    4
1800      2     7                    2
1900      3     5                    3
2000      3     9                    5
2100      3     4                    2
2200      4     8                    1
2300      4     9                    5
2400      5     7                    2
2500      5     9                    2
2600      5     6                    4
2700      5     2                    1
2800      6     9                    3
2900      6     4                    2
3000      6     3                    1
3100      7     9                    3
3200      7     2                    2
3300      8     9                    4
3400      8     4                    1
3500      8     6                    4
3600     10     1                    6
3700
3800    24   15   -9    4    7 -17   12 -23   16    0
3900    24
4000   -17
4100   -17
4200    24
4300    10 -23
*
```

Cuadro de datos correspondientes a la figura 3.10

1. La subrutina READ, con la ayuda de las subrutinas ORIGS y -- TERMS lee los datos correspondientes a los nodos y a los arcos de la red y la transforma en parámetros de los nodos y de los arcos. Además obtiene los apuntadores PO, PT y LT; como se puede ver a continuación:

REPRESENTACION DE LA RED--

NRO.DE NODOS__ 11 NRO.DE ARCOS__ 24

PARAMETROS TRASFORMADOS DE LOS NODOS

NODO I	B(I)
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	O(K)	T(K)	C(K)	H(K)
1	1	2	1	0
2	1	5	3	0
3	1	3	2	0
4	1	4	2	0
5	2	9	4	0
6	2	7	2	0
7	3	5	3	0
8	3	9	5	0
9	3	4	2	0
10	4	8	1	0
11	4	9	5	0
12	5	7	2	0
13	5	9	2	0
14	5	6	4	0
15	5	2	1	0
16	6	9	3	0
17	6	4	2	0
18	6	3	1	0
19	7	9	3	0
20	7	2	2	0
21	8	9	4	0
22	8	4	1	0
23	8	6	4	0
24	10	1	6	0

INDICE I/L	PO(I)	PT(I)	LT(L)
1	1	1	24
2	5	2	1
3	7	5	15
4	10	7	20
5	12	11	3
6	16	13	18
7	19	15	4
8	21	17	9
9	24	18	17
10	24	25	22
11	25	25	2
12	25	25	7
13	0	0	14
14	0	0	23
15	0	0	6
16	0	0	12
17	0	0	10
18	0	0	5
19	0	0	8
20	0	0	11
21	0	0	13
22	0	0	16
23	0	0	19
24	0	0	21

2. La subrutina ORIG obtiene la lista de arcos originados en el nodo I y su respectivo nodo terminal para cada arco. En la siguiente forma:

ARCOS QUE SE ORIGINAN EN EL NODO I

NODO NRO.	ARC ORIGINADO NRO.	NODO TERMINNAL NRO.
1	1	2
1	2	5
1	3	3
1	4	4
2	5	9
2	6	7
3	7	5
3	8	9
3	9	4
4	10	8
4	11	9
5	12	7
5	13	9
5	14	6
5	15	2
6	16	9
6	17	4
6	18	3
7	19	9
7	20	2
8	21	9
8	22	4
8	23	6
9		
10	24	1
11		

3. La subrutina TERM obtiene la lista de los arcos que terminan en el nodo I, con su respectivo nodo origen para cada arco. En la siguiente forma:

ARCOS QUE TERMINAN EN EL NODO I

NODO NRO.	ARCO TERMINAL NRO.	NODO ORIGEN NRO.
1	24	10
2	1	1
2	15	5
2	20	7
3	3	1
3	18	6
4	4	1
4	9	3
4	17	6
4	22	8
5	2	1
5	7	3
6	14	5
6	23	8
7	6	2
7	12	5
8	10	4
9	5	2
9	8	3
9	11	4
9	13	5
9	16	6
9	19	7
9	21	8
10		
11		

4. La subrutina TREIN con la ayuda de las subrutinas ADDTRE y - ROOT, representa el árbol mediante los apuntadores PF, PB, - PR y PD, para lo cual hay que proporcionar los datos iniciales del árbol, representado mediante los apuntadores hacia - atrás PB (como se puede observar en la lista de datos, en el renglón 3800 se especifican los apuntadores PB). Las subrutinas ADDTRE y ROOT son llamadas para cada arco que se va adicionando en el árbol, hasta cuando se incluye el último arco. En cada paso se va determinando los apuntadores PF, PB, PR y PD para cada nodo, hasta la entrada del último arco, momento en el cual la subrutina TREINT finaliza su proceso. A continuación se puede observar la impresión de los apuntadores de acuerdo a los aroos entrantes, y la gráfica del árbol asociado con la última lista de apuntadores.

```
                CONSTRUCCION DEL ARBOL

SUBRUTINA TREINT

ARBOL REPRESENTADO POR LOS APUNTADORES PB

 24  15  -9   4   7 -17  12 -23  16   0

SUBRUTINA ADDTRE

EL ARCO QUE ENTRA ES EL :  24

RESULTADOS

   I   PF   PB   PR   PD

   1    0   24    0    1
   2    0   15    0    0
   3    0   -9    0    0
   4    0    4    0    0
   5    0    7    0    0
   6    0  -17    0    0
   7    0   12    0    0
   8    0  -23    0    0
   9    0   16    0    0
  10    1    0    0    0

FIN DE ADDTRE

SUBRUTINA ADDTRE

EL ARCO QUE ENTRA ES EL :  15

RESULTADOS

   I   PF   PB   PR   PD

   1    0   24    0    1
   2    0   15    0    1
   3    0   -9    0    0
   4    0    4    0    0
   5    2    7    0    0
   6    0  -17    0    0
   7    0   12    0    0
   8    0  -23    0    0
   9    0   16    0    0
  10    1    0    0    0

FIN DE ADDTRE
```

SUBRUTINA ADDTRE

EL ARCO QUE ENTRA ES EL : -9

RESULTADOS

I	PF	PB	PR	PD
1	0	24	0	1
2	0	15	0	1
3	0	-9	0	1
4	3	4	0	0
5	2	7	0	0
6	0	-17	0	0
7	0	12	0	0
8	0	-23	0	0
9	0	16	0	0
10	1	0	0	0

FIN DE ADDTRE

SUBRUTINA ADDTRE

EL ARCO QUE ENTRA ES EL : 4

RESULTADOS

I	PF	PB	PR	PD
1	4	24	0	1
2	0	15	0	1
3	0	-9	0	3
4	3	4	0	2
5	2	7	0	0
6	0	-17	0	0
7	0	12	0	0
8	0	-23	0	0
9	0	16	0	0
10	1	0	0	0

FIN DE ADDTRE

SUBRUTINA ADDTRE

EL ARCO QUE ENTRA ES EL : 7

RESULTADOS

I	PF	PB	PR	PD
1	4	24	0	1
2	0	15	0	5
3	5	-9	0	3
4	3	4	0	2
5	2	7	0	4
6	0	-17	0	0
7	0	12	0	0
8	0	-23	0	0
9	0	16	0	0
10	1	0	0	0

FIN DE ADDTRE

SUBRUTINA ADDTRE

EL ARCO QUE ENTRA ES EL : -17

RESULTADOS

I	PF	PB	PR	PD
1	4	24	0	1
2	0	15	0	5
3	5	-9	0	3
4	6	4	0	2
5	2	7	0	4
6	0	-17	3	3
7	0	12	0	0
8	0	-23	0	0
9	0	16	0	0
10	1	0	0	0

FIN DE ADDTRE

SUBRUTINA ADDTRE

EL ARCO QUE ENTRA ES EL : 12

RESULTADOS

I	PF	PB	PR	PD
1	4	24	0	1
2	0	15	0	5
3	5	-9	0	3
4	6	4	0	2
5	7	7	0	4
6	0	-17	3	3
7	0	12	2	5
8	0	-23	0	0
9	0	16	0	0
10	1	0	0	0

FIN DE ADDTRE

SUBRUTINA ADDTRE

EL ARCO QUE ENTRA ES EL : -23

RESULTADOS

I	PF	PB	PR	PD
1	4	24	0	1
2	0	15	0	5
3	5	-9	0	3
4	6	4	0	2
5	7	7	0	4
6	8	-17	3	3
7	0	12	2	5
8	0	-23	0	4
9	0	16	0	0
10	1	0	0	0

FIN DE ADDTRE

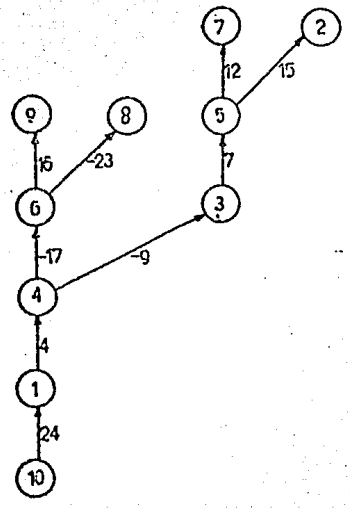
EL ARCO QUE ENTRA ES EL : 16

RESULTADOS

I	PF	PB	PR	PD
1	4	24	0	1
2	0	15	0	5
3	5	-9	0	3
4	6	4	0	2
5	7	7	0	4
6	9	-17	3	3
7	0	12	2	5
8	0	-23	0	4
9	0	16	8	4
10	1	0	0	0

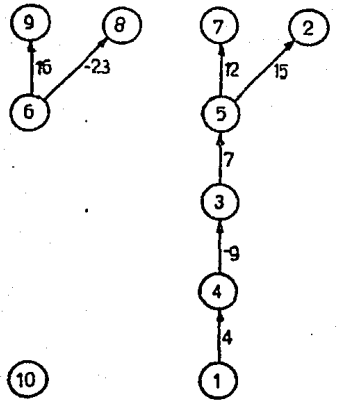
FIN DE ADDTRE

FIN DE TREINT



5. La subrutina DELTRE quita un arco desde la red y los subárbo les que se forman se representan mediante los apuntadores -- PF, PB y PR. En este caso, de la representación del árbol en el numeral 4, salen los arcos 24(10,1) y -17(4,6). Obtenien dose tres subárboles, como se puede observar a continuación:

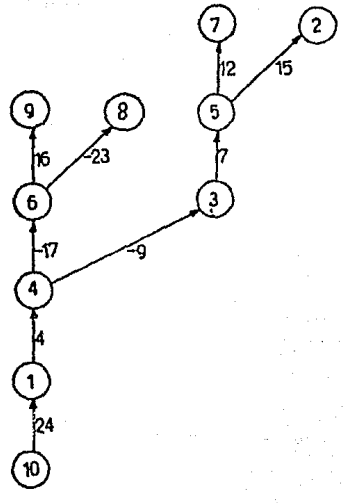
```
SUBRUTINA DELTRE

EL ARCO QUE SALE ES EL :  24

RESULTADS

   I PF(I) PB(I) PR(I)

   1    4     0     0
   2    0    15     0
   3    5    -9     0
   4    6     4     0
   5    7     7     0
   6    9   -17     3
   7    0    12     2
   8    0   -23     0
   9    0    16     8
  10    0     0     0

FIN DE DELTRE

SUBRUTINA DELTRE

EL ARCO QUE SALE ES EL : -17

RESULTADS

   I PF(I) PB(I) PR(I)

   1    4     0     0
   2    0    15     0
   3    5    -9     0
   4    3     4     0
   5    7     7     0
   6    9     0     0
   7    0    12     2
   8    0   -23     0
   9    0    16     8
  10    0     0     0

FIN DE DELTRE
```



6. La subrutina ADDTRE sirve para añadir un arco en la red y se forma un árbol representado nuevamente por los apuntadores -

PF, PB, PR y PD. A los subárboles representados anteriormente en el numeral 5, les añadimos los arcos 24(10,1) y -17(4,6). Obteniendose nuevamente el mismo árbol, el cual se puede observar en las listas de apuntadores que el programa imprime a continuación:

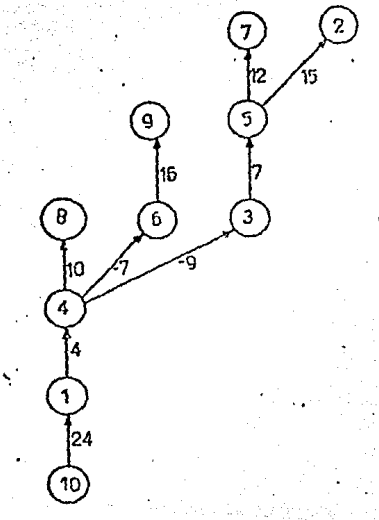
```
SUBRUTINA ADDTRE

EL ARCO QUE ENTRA ES EL : -17

RESULTADOS

   I   PF    PB    PR    PD

   1    4     0     0     1
   2    0    15     0     5
   3    5    -9     0     3
   4    6     4     0     2
   5    7     7     0     4
   6    9   -17     3     3
   7    0    12     2     5
   8    0   -23     0     4
   9    0    16     8     4
  10    0     0     0     0

FIN DE ADDTRE

SUBRUTINA ADDTRE

EL ARCO QUE ENTRA ES EL :  24

RESULTADOS

   I   PF    PB    PR    PD

   1    4    24     0     1
   2    0    15     0     5
   3    5    -9     0     3
   4    6     4     0     2
   5    7     7     0     4
   6    9   -17     3     3
   7    0    12     2     5
   8    0   -23     0     4
   9    0    16     8     4
  10    1     0     0     0

FIN DE ADDTRE
```



7. Para quitar un arco y añadir otro o el mismo se usa la subrutina TRECHG con la ayuda de las subrutinas DELTRE y ADDTRE. Partiendo de la red obtenida anteriormente en el numeral 6, se quita el arco -23(6,8) y entra el arco 10(4,8). Obtenién

dose un nuevo árbol, el cual se puede observar a continuación:

```
UBRUTINA TRECHG

EL ARCO QUE SALE ES EL :  -23  EL ARCO QUE ENTRA ES  EL :   10

SUBRUTINA DELTRE

EL ARCO QUE SALE ES EL : -23

RESULTADS

   I PF(I) PB(I) FR(I)

   1    4    24     0
   2    0    15     0
   3    5    -9     0
   4    6     4     0
   5    7     7     0
   6    9   -17     3
   7    0    12     2
   8    0     0     0
   9    0    16     0
  10    1     0     0

FIN DE DELTRE

SUBRUTINA ADDTRE

EL ARCO QUE ENTRA ES EL :  10

RESULTADOS

   I   PF   PB   FR   PB

   1    4   24    0    1
   2    0   15    0    5
   3    5   -9    0    3
   4    8    4    0    2
   5    7    7    0    4
   6    9  -17    3    3
   7    0   12    2    5
   8    0   10    6    3
   9    0   16    0    4
  10    1    0    0    0

FIN DE ADDTRE

FIN DE TRECHG
```



CAPITULO IV

EL PROBLEMA DE RUTA MAS CORTA:

4.1 Introducción.

4.2 Representación matemática como un problema de flujo a costo mínimo.

4.3 Solución del problema cuando todos los arcos admisibles tienen costos positivos, usando el algoritmo de Dijkstra.

4.4 Solución del problema cuando algunos arcos tienen costos negativos con ciclos positivos, usando el algoritmo primal.

4.5 Empleo del algoritmo combinado (Dijkstra y primal) para problemas con ciclos negativos.

4.6 Otro método de solución del problema cuando algunos arcos tienen costos negativos, usando un algoritmo no básico.

4.7 Otro método de solución del problema cuando algunos arcos tienen costos negativos, usando un algoritmo dual.

4.1 INTRODUCCION.

En este capítulo se resolverá el problema de redes de flujo, para obtener la ruta más corta desde un nodo fuente s a un nodo sumidero t, en el cual cada arco de la red tiene asociado el parámetro costo que define la longitud de los arcos, por tanto la longitud de una trayectoria se define como la suma de las longitudes de sus arcos. Al resolver el problema pueden presentarse tres casos:

1. Todos los arcos tienen longitudes positivas.
2. Algunos arcos tienen longitudes negativas pero no presentan ciclos negativos.
3. Algunos arcos tienen longitudes negativas y presentan uno o más ciclos negativos.

En este último caso no es posible resolver el problema de ruta más corta, pero los algoritmos conducen a descubrir e identificar el ciclo negativo.

En el primer caso se empleará el método Dijkstra, mediante el algoritmo DSHORT. En el segundo caso se empleará un algoritmo primal PSHORT que inicia con un árbol básico factible. En el tercer caso se usará el algoritmo SHORT, que es una combinación de los algoritmos DSHORT y PSHORT. Finalmente se mencionarán otros dos métodos de solución cuando existen arcos con costos negativos, y estos son el algoritmo no básico NBSHORT y el algoritmo dual DUALSP. Estos algoritmos encuentran la ruta más corta desde el nodo s al nodo t o a todos los otros nodos. Se identifica el caso en el cual las rutas más cortas a todos los nodos es determinada definiendo t = o.

4.2 REPRESENTACION MATEMATICA COMO UN PROBLEMA DE FLUJO A COSTO MINIMO.

Para representar el problema de ruta más corta en forma de flujo a costo mínimo, se identifica un parámetro costo que define la longitud del arco. Sea $b_s = n-1$ y $b_i = -1$ para todos los otros nodos, el problema de ruta más corta en la red $D = [N,M]$ puede establecerse como un problema primal de programación lineal en la siguiente forma:

$$\text{Min.} \quad \sum_{K \in M} h_k f_k \tag{1}$$

$$\text{S.a.} \quad \sum_{k \in M_{o_i}} f_k - \sum_{k \in M_{T_i}} f_k = -1 \qquad i \neq s \tag{2}$$

$$\sum_{k \in M_{o_s}} f_k - \sum_{k \in M_{T_s}} f_k = n-1 \tag{3}$$

$$f_k \geq 0 \qquad k \in M$$

$$f_k \leq n \qquad k \in M$$

La total unimodularidad de las ecuaciones de conservación de flujo asegura que todos los flujos óptimos sean cantidades enteras. El problema dual es:

$$\text{Min.} \quad - \sum_{j \in N-s} \Pi_i + (n-1)\,\Pi_s + n \sum \delta_k \tag{5}$$

$$\text{s.a.} \quad \Pi_i - \Pi_j + \delta_k \geq - h_k \quad \text{para} \quad k(i,j) \in M \tag{4}$$

$$\Pi_i \quad \text{no restringida para toda } i$$

$$\delta_k \geq 0 \qquad k \in M$$

Considerando el Teorema 3, se puede escribir las condiciones de optimalidad para los problemas dual y primal, de la siguiente manera:

1. Factibilidad primal.
   (a) Conservación de flujo.
   (b) $0 \leq f_k \leq n$
2. Factibilidad restringida del dual.
   $\delta_k = \text{Máx.} \left[0, -d_k\right] \qquad (d_k = \Pi_i - \Pi_j + h_k)$
3. Holgura complementaria.
   (a) Si $0 < f_k < n$, $d_k = 0$
   (b) Si $d_k > 0$, $f_k = 0$
   (c) Si $d_k < 0$, $f_k = n$

Note que f_k nunca puede excederse de n-1 unidades de flujo para algún arco. Esto implica que δ_k debe ser siempre mayor o igual a cero para todo arco k, por lo que cada δ_k debe ser igual a cero, lo cual permite simplificar las condiciones de optimalidad en la siguiente forma:

1. Factibilidad primal.
   (a) Conservación de flujo.
   (b) $f_k \geq 0$
2. Factibilidad restringida del dual.
   $\delta_k = 0$ para todo k
3. Holgura complementaria.
   (a) Si $f_k > 0$, $d_k = 0$
   (b) Si $d_k > 0$, $f_k = 0$
   (c) $d_k \geq 0$

Substituyendo $\delta_k = 0$ en las ecuaciones (4) y (5), tenemos:

$$\text{Máx.} \quad \sum_{j \in N-s} \Pi_i - (n-1)\,\Pi_s = \text{Máx} \sum_{j \in N-s} (\Pi_i - \Pi_s) \qquad (4a)$$

$$\text{s.a.} \quad \Pi_j - \Pi_i \leq h_k \quad \text{para} \quad k(i,j) \in M \qquad (5a)$$

Π_i no restringida.

La variable dual Π_j representa la distancia desde algún nodo origen arbitrario a su asociado nodo j. Si el interés es encontrar la distancia desde el nodo fuente a todos los otros nodos ($i \in N-s$), entonces la distancia para el mismo nodo fuente es, $\Pi_s = 0$. Esto permite reducir las ecuaciones anteriores a:

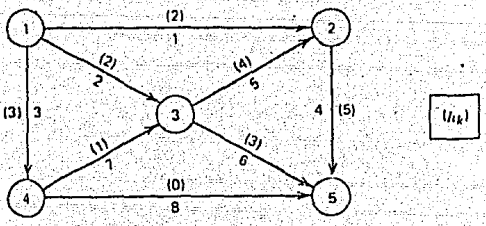
$$\text{Máx.} \quad \sum_{i \in N-s} \Pi_i \qquad (4b)$$

$$\text{s.a.} \quad \Pi_j - \Pi_i \leq h_k \quad \text{para} \quad k(i,j) \in M \qquad (5b)$$

Π_i no restringido $\quad i \in M - s$

$$\Pi_s = 0$$

El ejemplo de la figura 4.1 ilustra la red original, y las formulaciones primal y dual, de acuerdo a las ecuaciones (1), (2), (3), (4b) y (5b).



Primal:

$$\begin{aligned}
\text{Min. } & 2f_1+2f_2+3f_3+5f_4+4f_5+3f_6+f_7+0f_8 \\
\text{s.t. } & f_1+f_2+f_3 = 4 \\
& -f_1+f_4-f_5 = -1 \\
& -f_2+f_5+f_6-f_7 = -1 \\
& -f_3+f_7+f_8 = -1 \\
& -f_4-f_6-f_8 = -1 \\
& \text{all } f_k \geq 0
\end{aligned}$$

Dual:

$$\begin{aligned}
\text{Max. } & \pi_2+\pi_3+\pi_4+\pi_5 \\
\text{s.t. } & \pi_2 \leq 2 \\
& \pi_3 \leq 2 \\
& \pi_4 \leq 3 \\
& -\pi_2+\pi_5 \leq 5 \\
& \pi_2-\pi_3 \leq 4 \\
& -\pi_3+\pi_5 \leq 3 \\
& \pi_3-\pi_4 \leq 1 \\
& -\pi_4+\pi_5 \leq 0
\end{aligned}$$

π_i irrestricto $i=2,3,4,5$

Figura 4.1.

(a) Red original. (b) Formulación primal. (c) Formulación dual.

Aquí se puede observar que el problema primal busca mínimizar la distancia total recorrida por n-1 unidades de flujo, requeriendo que cada uno de los nodos i ($i \in N-s$), reciba exactamente una unidad de flujo. En cambio el problema dual procura máximizar la distancia total desde el nodo s a todos los otros nodos, requeriendo que la diferencia entre las distancias desde el nodo s a los nodos i y j no sea mayor que h(i,j) si el arco k(i,j) existe.

Los algoritmos que se verán en este capítulo para la solución del problema, se basan en la interpretación dual.

Deben considerarse tres aspectos para la solución del problema:

1. La solución del problema primal está definida por n-1 arcos, los cuales forman un árbol dirigido $D_T = [N, M_T]$ donde $M_T \subset M$.

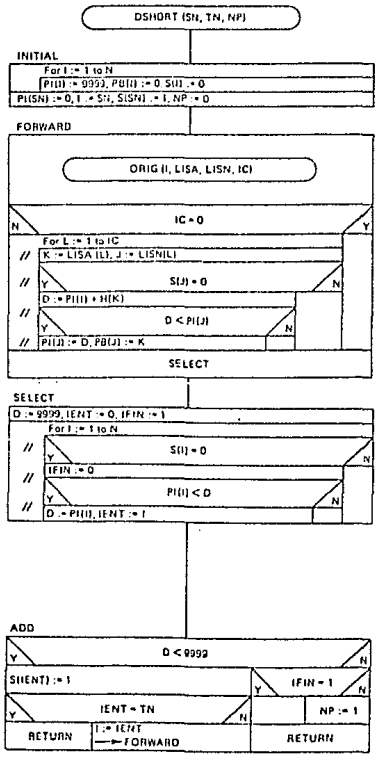
2. Los arcos no básicos tienen flujo cero.
3. Para el problema de ruta más corta, los flujos en la red sirven solamente para definir el árbol que está representado por los apuntadores hacia atrás p_B (i), a cada nodo.

4.3 SOLUCION DEL PROBLEMA CUANDO TODOS LOS ARCOS ADMISIBLES TIENEN COSTOS POSITIVOS, USANDO EL ALGORITMO DIJKSTRA.

Este problema se resuelve por medio del algoritmo Dijkstra, en el cual todos los arcos admisibles son positivos y cumplen extrictamente con la inecuación (5a). Inicia estableciendo el valor de $\Pi_s = 0$ y encuentra las rutas más cortas desde el nodo fuente s a los nodos siguientes; adicionando un nodo en cada iteración. El algoritmo que realiza este proceso es el algoritmo DSHORT.

ALGORITMO DSHORT.

Propósito: Encuentra la trayectoria más corta del nodo s al nodo t, cuando todos los arcos admisibles tienen costos positivos. Si t = 0, el algoritmo da el árbol de trayectoria más corta.



1. (INICIAR) Hace $\Pi_i = R$ (siendo R un número muy grande) para todos los nodos excepto para s.
   Inicializa los apuntadores hacia atrás con el valor de cero. Define el conjunto de nodos S = {s} . Hace $\Pi_s := 0$
2. (ADELANTE) Para cada arco hacia adelante originado en el nodo i, k(i,j), tal que $j \notin s$, calcula la longitud de la trayectoria al nodo j a través del nodo i, usando $\Pi_i + h_k$. Si este valor es menor que Π_j , reemplaza Π_j por $\Pi_i + h_k$. Reemplaza el apuntador hacia atrás de j con i.
3. (SELECCIONAR) Encuentra D = Min. $\{\Pi_j : j \in N-s\}$. i_E es el nodo para el cual se obtiene -

el mínimo. IFIN es igual a cero si algún nodo no está incluido en S.

4. (SUMA) Si D<R incluye i_E en S. Si i_E = t, termina con la trayectoria más corta de s a t. Si $i_E \neq t$ hace i = i_E y va al paso 2. Si D = R y todos los nodos están en el conjunto S, termina con el árbol de trayectoria más corta. Si todos los nodos no están en S, no hay trayectoria de s a t, y hace NP = 1.

La figura 4.2a presenta la red original, la figura 4.2b indica una parte de la red luego de la cuarta iteración, cuando S = { 1, 3, 4, 2, 6 }. Los valores de Π_i son calculados en forma secuencial y se encuentran entre paréntesis rectangulares junto a cada nodo. Finalmente la figura 4.2c presenta el árbol óptimo, luego de la última iteración.

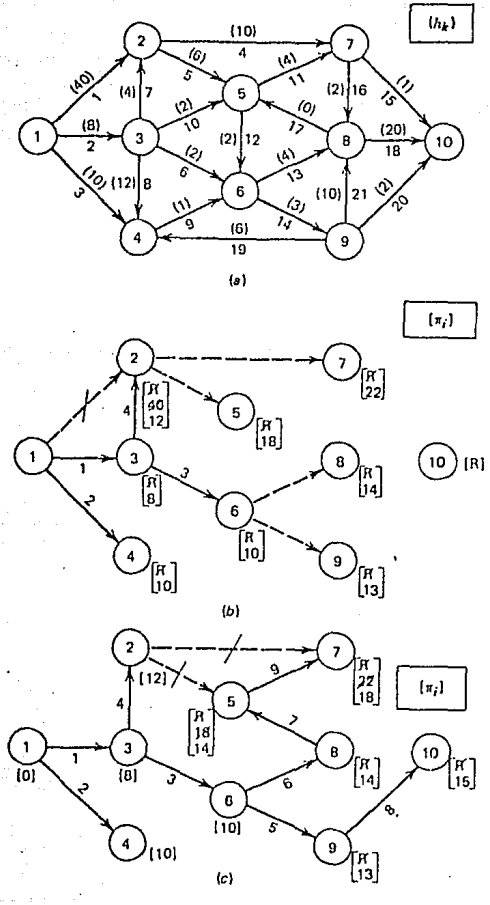
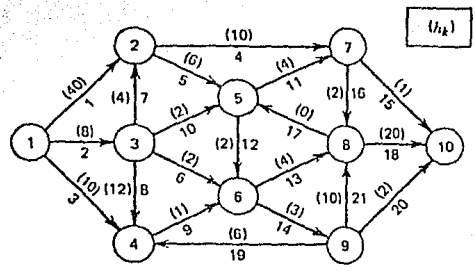


Figura 4.2
(a) Red original. (b) Cuarta iteración. (c) Arbol de ruta más corta.

En el anexo A-2 se encuentra la subrutina principal COMI43, codificada en FORTRAN, la misma que utiliza el método Dijkstra (algoritmo DSHORT) para resolver la red de la figura 4.2, cuyos datos de entrada y resultados se pueden observar en las figuras 4.3 y 4.4, respectivamente.



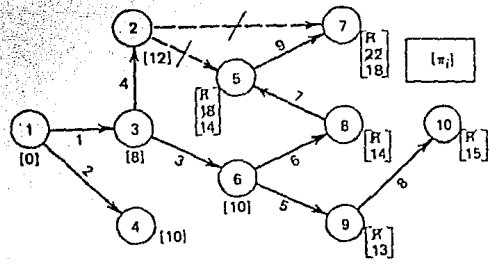

```
100     43
200      1
300     10
400      1          9
500      2         -1
600      3         -1
700      4         -1
800      5         -1
900      6         -1
1000     7          -1
1100     8          -1
1200     9          -1
1300    10          -1
1400
1500    01   02              10.          40.
1600    01   03               8.           8.
1700    01   04               6.          10.
1800    02   07              20.           10.
1900    02   05               6.           6.
2000    05   03               8.           2.
2100    03   06               4.           2.
2200    04   06               8.           1.
2300    09   04               2.           6.
2400    05   07              12.           4.
2500    08   05              12.           0.
2600    06   08               4.           4.
2700    06   09               8.           3.
2800    07   10               8.           1.
2900    08   10               2.          20.
3000    09   10               4.           2.
3100    03   02               4.           4.
3200    03   04              10.          12.
3300    05   06               2.           2.
3400    07   08               5.           2.
3500    09   08               6.           10.
3600
*
```

Fig. 4.3

```
EL NRO. CLAVE DE ESTA SUBRUTINA ES :   43

PROGRAMA CONI43
DIJKSTRA, COSTO MINIMO CON COSTOS POSITIVOS

EL NODO    1  ES EL NODO RAIZ DE LA RUTA MAS CORTA

          REPRESENTACION DE LA RED--
               NRO. DE NODOS--   11          NRO. DE ARCOS--   21
```

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	9
2	-1
3	-1
4	-1
5	-1
6	-1
7	-1
8	-1
9	-1
10	-1
11	0



PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	O(K)	T(K)	C(K)	H(K)
1	1	2	10	40
2	1	3	8	8
3	1	4	6	10
4	2	7	20	10
5	2	5	6	6
6	3	6	4	2
7	3	2	4	4
8	3	4	10	12
9	4	6	8	1
10	5	3	8	2
11	5	7	12	4
12	5	6	2	2
13	6	8	4	4
14	6	9	8	3
15	7	10	8	1
16	7	8	5	2
17	8	5	12	0
18	8	10	2	20
19	9	4	2	6
20	9	10	4	2
21	9	8	6	10

```
LA SOLUCION OPTIMA DEL PROBLEMA ES :
```

I	P(I)	PB(I)	O(I)	T(I)
1	0	0	0	0
2	12	7	3	2
3	8	2	1	3
4	10	3	1	4
5	14	17	8	5
6	10	6	3	6
7	18	11	5	7
8	14	13	6	8
9	13	14	6	9
10	15	20	9	10

```
FIN DEL PROGRAMA RFCLSG

EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES:  0.183333

#ET=1:59.3 PT=0.3 IO=0.2
```

Fig. 4.4

4.4 SOLUCION DEL PROBLEMA CUANDO ALGUNOS ARCOS TIENEN COSTOS NEGATIVOS CON CICLOS POSITIVOS, USANDO UN ALGORITMO PRIMAL.

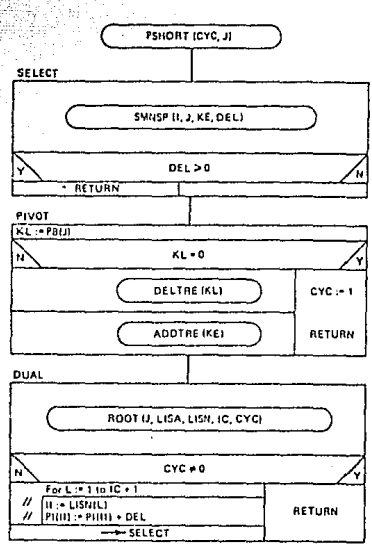
Cuando algunos arcos tienen costos negativos, el algoritmo - Dijkstra no es eficiente. Pueden presentarse dos alternativas al aplicar el algoritmo Dijkstra a este problema. Encontrar una solución óptima o uno o más arcos violan las condiciones de holgura complementaria para los arcos básicos. Las variables duales para cada arco básico k(i,j) deben cumplir la condición:

$$\Pi_i + h_k = \Pi_j \qquad (9)$$

Dado un árbol básico factible el algoritmo STARTD determina los potenciales iniciales asociados a cada nodo, y el algoritmo - SMNSP encuentra el arco con el valor más negativo de $\Pi_i + h_k - \Pi_j$

ALGORITMO PSHORT.

Propósito: Inicia con un árbol asociado a una solución básica factible y - los potenciales en los nodos que violan la condición de factibilidad del problema dual para algunos arcos no básicos. Si no se satisface, entonces procede iterativamente a modificar -- las bases del árbol y los potenciales en los nodos hasta obtener una solu-- ción óptima.

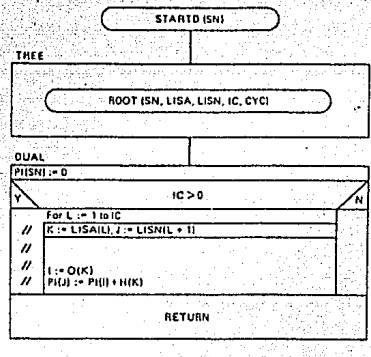


1. (SELECCIONAR) Encuentra un arco no básico $k_E(i,j)$ para el cual $\Pi_i + h_k < \Pi_j$. Si no existe tal arco, finaliza con la solución óptima. De otra manera, hace $d_E = \Pi_i + h_k - \Pi_j$
2. (PIVOTE) Sea $k_L(i',j)$ el arco básico que termina en el nodo j. Quita este arco de la base del árbol y añade al arco (no-básico) k_E a la base del árbol.
3. (DUAL) Encuentra los nodos en el árbol enraizado al nodo j. Si se localiza un ciclo negativo, termina con CYC = 1 y J un nodo en el ciclo. No hay solución del problema. De otra manera su

ma d_E a los potenciales de todos los nodos en el subárbol.

ALGORITMO STRATD.

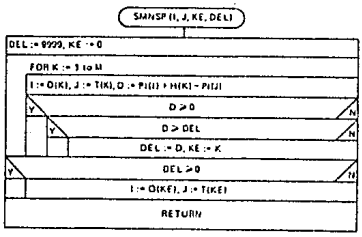
Propósito: Dado un árbol básico, éste algoritmo determina los potenciales en los nodos tal que $\Pi_i + h_k = \Pi_j$ para cada arco básico k(i,j); admitiendo solamente arcos hacia adelante.



1. (ARBOL) Lista los nodos y arcos del árbol en orden tal que cada nodo (arco) aparece en la lista después de todos sus nodos (arcos) predecesores. La lista de nodos (LISN) y la lista de arcos (LISA) son construídos por ROOT tal que el nodo terminal de la i-écima entrada en la lista de arcos es la i + 1 entrada en la lista de nodos. Hace $\ell = 1$ y $\Pi_s = 0$.
2. (DUAL) Sea k la ℓ-écima entrada en la lista de arcos. j es la (ℓ + 1)-écima entrada en la lista de nodos. Encuentra el origen del arco k. Sea $\Pi_j = \Pi_i + h_k$. Si el final de la lista de arcos es encontrado, termina. De otra manera, incrementa ℓ en 1 y repite el paso 2.

ALGORITMO SMNSP.

Propósito: Encuentra el arco con el valor más negativo de $\Pi_i + \Pi_k - \Pi_j$.



1. Hace D=9999 y $k_E = 0$.
2. Para cada arco k(i,j) calcula $d_k = \Pi_i + h_k - \Pi_j$. Si $d_k \geqslant 0$ va al siguiente arco. Si $d_k < 0$ y $d_k < D$, hace $k_E = k$ y $D = d_k$ y va al siguiente arco.
3. Después completa la búsqueda de los arcos, si un $d_k < 0$ ha sido encontrado, hace I = O(KE) y J = T(KE), y retorna.

Para ilustrar el uso de estos algoritmos se resuelve la red de la figura 4.5a, la cual presenta algunos arcos con costos negativos. En la figura 4.5b se observa el árbol básico inicial. La figura 4.5c indica el árbol con los arcos no básicos en líneas punteadas; el valor d_k se menciona entre paréntesis sobre cada uno de los arcos. Puesto que $d_k < 0$.

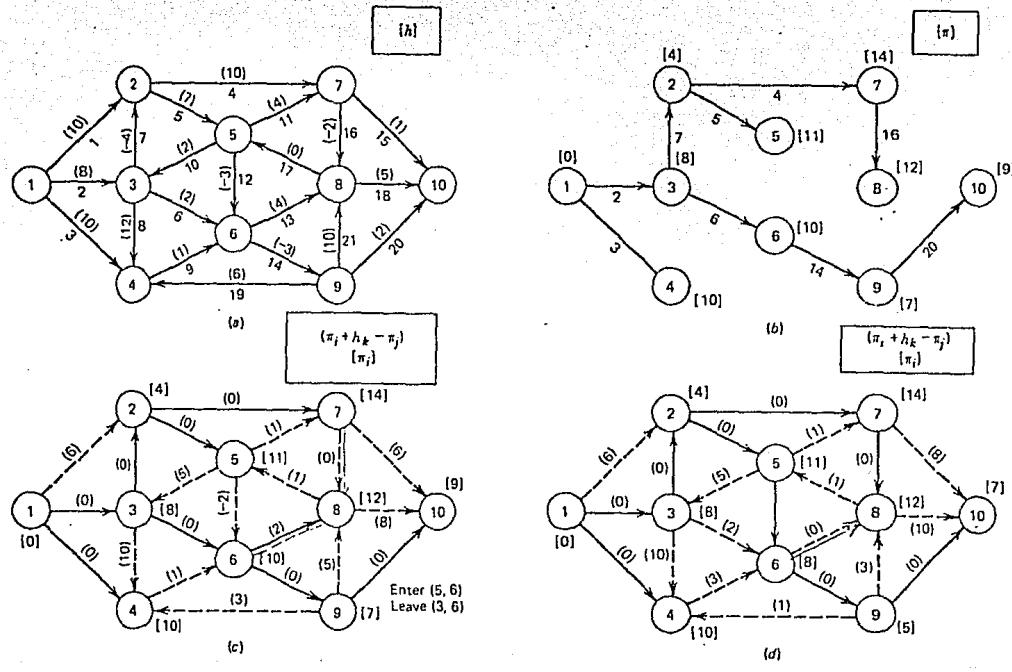
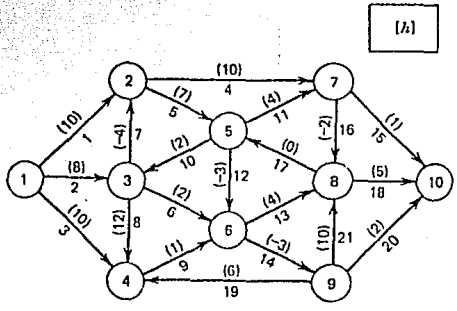


Figura 4.5
(a) Red original. (b) Arbol básico determinado por el algoritmo Dijkstra. (c) Arbol con los arcos no básicos en líneas punteadas. (d) Solución óptima.

para algunos arcos, la solución no es óptima. Luego el arco 12 (5,6) que tiene el menor valor d_k es elegido para entrar a la base, y el arco 6 (3,6) sale de la base. En la figura 4.5d se obtiene la solución óptima.

En el anexo A-2 se encuentra la subrutina principal COMI44, codificada en FORTRAN, la misma que utiliza el algoritmo primal PSHORT con la ayuda del algoritmo STARTD, para resolver la red de la figura 4.5a, cuyos datos de entrada y resultados se pueden observar en las figuras 4.6 y 4.7, respectivamente.

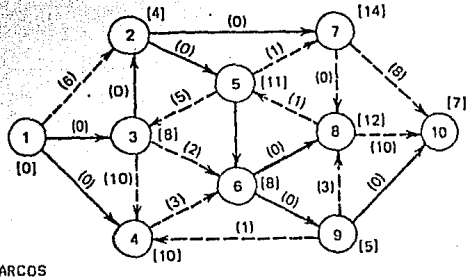



```
100     44
200      1
300     10
400      1          9
500      2         -1
600      3         -1
700      4         -1
800      5         -1
900      6         -1
1000     7         -1
1100     8         -1
1200     9         -1
1300    10         -1
1400
1500    01   02              10.          10.
1600    01   03               8.           8.
1700    01   04               6.          10.
1800    02   07              20.           10.
1900    02   05               6.           7.
2000    05   03               8.           2.
2100    03   06               4.           2.
2200    04   06               8.           1.
2300    09   04               2.           6.
2400    05   07              12.           4.
2500    08   05              12.           0.
2600    06   08               4.           4.
2700    06   09               8.          -3.
2800    07   10               8.           1.
2900    08   10               2.           5.
3000    09   10               4.           2.
3100    03   02               4.          -4.
3200    03   04              10.          12.
3300    05   06               2.          -3.
3400    07   08               5.          -2.
3500    09   08               6.          10.
3600
3700    0   1   2  19   5  12   4  21  14  18
3800
$
```

Fig. 4.6

```
EL NRO. CLAVE DE ESTA SUBRUTINA ES :   44

PROGRAMA COMI44
PRIMAL DE RUTA MAS CORTA

EL USUARIO DEBE PROPORCIONAR UN ARBOL BASICO INICIAL

EL NODO RAIZ ES    1

                REPRESENTACION DE LA RED--

                NRO. DE NODOS--   11          NRO. DE ARCOS--   21
```

$(\pi_i + h_k - \pi_j)$
(π_i)



PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	9
2	-1
3	-1
4	-1
5	-1
6	-1
7	-1
8	-1
9	-1
10	-1
11	0

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	O(K)	T(K)	C(K)	H(K)
1	1	2	10	10
2	1	3	8	8
3	1	4	6	10
4	2	7	20	10
5	2	5	6	7
6	3	6	4	2
7	3	2	4	-4
8	3	4	10	12
9	4	6	8	1
10	5	3	8	2
11	5	7	12	4
12	5	6	2	-3
13	6	8	4	4
14	6	9	8	-3
15	7	10	8	1
16	7	8	5	-2
17	8	5	12	0
18	8	10	2	5
19	9	4	2	6
20	9	10	4	2
21	9	8	6	10

RUTA MAS CORTA DESDE EL NODO RAIZ 1

I	PI(I)	PB(I)	O(I)	T(I)
1	0	0	0	0
2	4	7	3	2
3	8	2	1	3
4	10	3	1	4
5	11	5	2	5
6	8	12	5	6
7	14	4	2	7
8	12	13	6	8
9	5	14	6	9
10	7	20	9	10

```
FIN DEL PROGRAMA RFCLSG

EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES:  0.200000

#ET=1150.6 PT=0.3 IO=0.2
```

Fig. 4.7

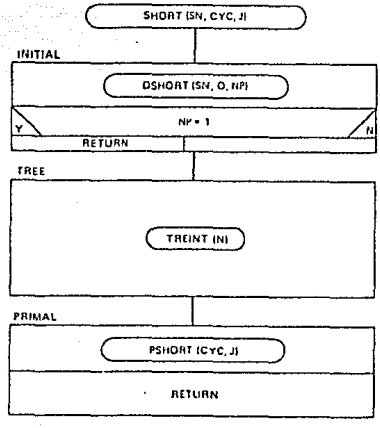
4.5 EMPLEO DEL ALGORITMO COMBINADO (DIJKSTRA Y PRIMAL) PARA PROBLEMAS CON CICLOS NEGATIVOS.

Si la red tiene un ciclo con una longitud total negativa, el problema primal de ruta más corta se vuelve ilimitado, cuando esto sucede el problema dual no tiene solución factible, por lo que el algoritmo primal de la sección anterior no puede obtener la solución óptima. Si se encuentra un ciclo negativo, éste es representado por medio de apuntadores.

Si en una red se presentan arcos con costos negativos, y además inicialmente no se dispone de un árbol básico factible, el problema se resuelve usando el algoritmo SHORT que es una combinación del algoritmo DSHORT (obtiene un árbol básico factible mediante el método Dijstra) y el algoritmo PSHORT (obtiene la solución óptima o un ciclo negativo).

ALGORITMO SHORT.

Propósito: Encuentra el árbol con trayectorias más cortas en una red que puede tener arcos con costos negativos. El algoritmo termina con el árbol de trayectoria más corto (desde SN a los demás nodos) o bien, encuentra un ciclo negativo (CYC = 1) y proporciona un nodo J en dicho ciclo.



1. (INICIA) Usa el algoritmo Dijkstra para encontrar una solución básica factible para el árbol de trayectoria más corta. Si no hay árbol generador, indica por NP = 1 y retorna.
2. (ARBOL) Obtiene el árbol básico representándolo mediante apuntadores, usando DSHORT.
3. (PRIMAL) Usa el algoritmo primal para verificar la optimalidad de la solución inicial y efectúa los cambios necesarios para obtener el árbol de trayectoria más corta. Si se encuentra un ciclo negativo, lo indica con CYC = 1 y J es un nodo en ese ciclo.

La solución de este tipo de problemas se ilustra mediante un ejemplo en la figura 4.8, en la cual se puede ver claramente la formación de un ciclo - negativo Mc = {12, 13, 17} . El ciclo es encontrado en el algoritmo ROOT y el algoritmo primal termina indicando el ciclo negativo.

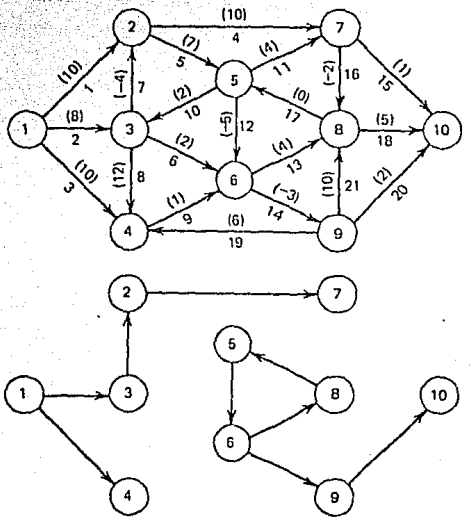
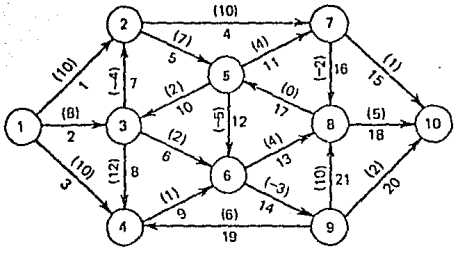


Figura 4.8
(a) Red original. (b) Se encuentra el ciclo negativo $M_c = \{12, 13, 17\}$.

En el anexo A-2 se encuentra la subrutina principal COMI45, codificada en FORTRAN, la misma que utiliza el algoritmo SHORT (combinación de los algoritmos DSHORT y PSHORT), para resolver la red de la figura 4.8, cuyos datos de entrada y resultados se pueden observar en la figura 4.9 y 4.10, respectivamente.




```
100     45
200      1
300     10
400      1          9
500      2         -1
600      3         -1
700      4         -1
800      5         -1
900      6         -1
1000     7         -1
1100     8         -1
1200     9         -1
1300    10         -1
1400
1500     01   02              10.          10.
1600     01   03               8.           8.
1700     01   04               6.          10.
1800     02   07              20.          10.
1900     02   05               6.           7.
2000     05   03               8.           2.
2100     03   06               4.           2.
2200     04   06               8.           1.
2300     09   04               2.           6.
2400     05   07              12.           4.
2500     08   05              12.           0.
2600     06   08               4.           4.
2700     06   09               8.          -3.
2800     07   10               8.           1.
2900     08   10               2.           5.
3000     09   10               4.           2.
3100     03   02               4.          -4.
3200     03   04              10.          12.
3300     05   06               2.          -5.
3400     07   08               5.          -2.
3500     09   08               6.          10.
3600
3700
$
```

Fig. 4.9

EL NRO. CLAVE DE ESTA SUBRUTINA ES : 45

PROGRAMA COMI45
COMBINADO DE RUTA MAS CORTA
CON UN ARBOL INICIAL DADO POR EL ALGORITMO DIJKSTRA

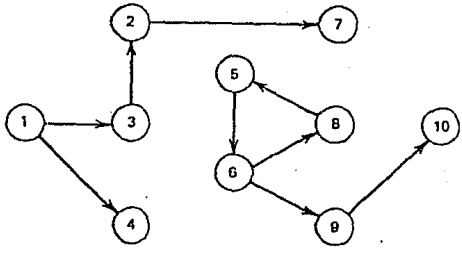
EL NODO RAIZ ES 1

REPRESENTACION DE LA RED--

NRO. DE NODOS-- 11 NRO. DE ARCOS-- 21

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	9
2	-1
3	-1
4	-1
5	-1
6	-1
7	-1
8	-1
9	-1
10	-1
11	0



PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	O(K)	T(K)	C(K)	H(K)
1	1	2	10	10
2	1	3	8	8
3	1	4	6	10
4	2	7	20	10
5	2	5	6	7
6	3	6	4	2
7	3	2	4	-4
8	3	4	10	12
9	4	6	8	1
10	5	3	8	2
11	5	7	12	4
12	5	6	2	-5
13	6	8	4	4
14	6	9	8	-3
15	7	10	8	1
16	7	8	5	-2
17	8	5	12	0
18	8	10	2	5
19	9	4	2	6
20	9	10	4	2
21	9	8	6	10

UN CICLO NEGATIVO FUE ENCONTRADO
SE LOCALIZA POR LOS APUNTADORES PB(I) DESDE EL NODO 5

I	PI(I)	PB(I)	O(I)	T(I)
1	0	0	0	0
2	4	7	3	2
3	8	2	1	3
4	10	3	1	4
5	11	17	8	5
6	6	12	5	6
7	14	4	2	7
8	10	13	6	8
9	3	14	6	9
10	5	20	9	10

FIN DEL PROGRAMA RFCLSG

EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES: 0.200000

Fig. 4.10

4.6 OTRO METODO DE SOLUCION DEL PROBLEMA CUANDO ALGUNOS ARCOS TIENEN COSTOS NEGATIVOS, USANDO UN ALGORITMO NO BASICO.

Cuando algunos arcos tienen costos negativos, otra forma de resolver el problema es utilizando el algoritmo no básico NBSHORT, el cual no requiere de la manipulación de un árbol básico y usa la condición $d_k \geq 0$. En este algoritmo, la red puede estar representada por una simple lista de arcos. La ventaja de aplicar este método es que ocupa poco espacio de memoria en la computadora, pero su desventaja es el tiempo de proceso que generalmente es grande.

El algoritmo inicialmente establece las trayectorias más cortas (Π_i) con un número grande para todos los nodos, excepto para el nodo fuente que lo asigna con $\Pi_s = 0$. Luego chequea la condición $\Pi_i + h_k \geq \Pi_j$ para cada arco k (i,j). Obviamente la condición es inicialmente violada en algún arco que se origina en el nodo fuente. Siempre que se encuentra un arco k (i,j) que viola esta condición, se hace $\Pi_j = \Pi_i + h_k$. En esta forma, los valores de Π_j decrecen hasta que se integran en la trayectoria más corta a cada nodo. El proceso del algoritmo asegura la existencia de al menos un arco k (i,j) que termina en un nodo j que no sea el nodo fuente, (asumiento que la red es conectada) tal que:

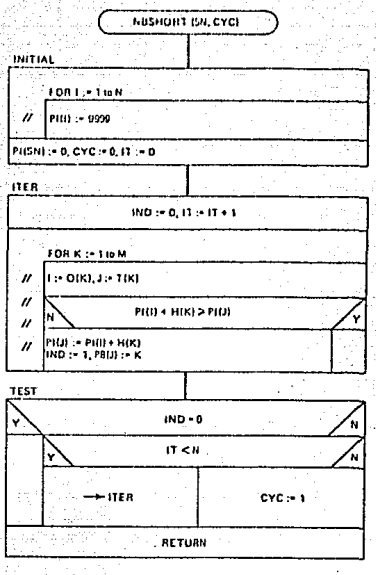
$$\Pi_i + h_k = \Pi_j$$

La selección de un arco para cada nodo (excepto el nodo fuente) crea un árbol generador enraizado en el nodo fuente. Este árbol es usado para encontrar la solución primal factible, por tanto, las condiciones de holgura complementaria se satisfacen para las soluciones primal y dual, este árbol es el árbol de ruta más corta. Se debe notar que para la optimalidad usa una base primal al terminar, pero durante su proceso no mantiene una representación básica y solamente opera en las variables duales. Este algoritmo es similar al algoritmo primal PSHORT en que ambos van desde una solución dual que viola la condición $d_k \geq 0$ a una solución óptima. El algoritmo primal, sin embargo, mantiene y usa un árbol básico primal a lo largo del proceso de solución.

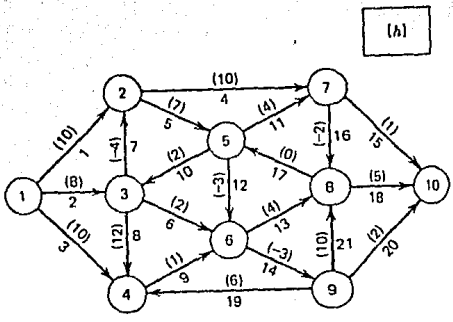
ALGORITMO NBSHORT.

Propósito: Encuentra la longitud de la trayectoria más corta desde el nodo fuente a todos los demás, a través de los arcos. El algoritmo usa una representación no básica.

1. (INICIA) Sea $\Pi_i := 9999$, para cada nodo i y $\Pi_s := 0$, donde el nodo S es el nodo fuente.
2. (ITER) Para cada arco k (i,j), si $\Pi_i + h_k < \Pi_j$, reemplaza Π_j por $\Pi_i + h_k$ y registra su apuntador hacia atrás. De otra forma no hace nada.
3. (PRUEBA) Si ningún potencial fue cambiado en el paso 2, termina con la solución óptima. De otra forma regresa al paso 2.



En el anexo A-2 se encuentra la subrutina principal COMI46, codificada en FORTRAN, la misma que utiliza el algoritmo no básico NBSHORT, para resolver la red de la figura 4.5a, cuyos datos de entrada y resultados se pueden observar en la figura 4.11 y 4.12, respectivamente.




```
100     46
200      1
300     10
400      1
500      1          9
600      2         -1
700      3         -1
800      4         -1
900      5         -1
1000     6         -1
1100     7         -1
1200     8         -1
1300     9         -1
1400    10         -1
1500
1600    01   02              10.          10.
1700    01   03               8.           8.
1800    01   04               6.          10.
1900    02   07              20.           10.
2000    02   05               6.           7.
2100    05   03               8.           2.
2200    03   06               4.           2.
2300    04   06               8.           1.
2400    09   04               2.           6.
2500    05   07              12.           4.
2600    08   05              12.           0.
2700    06   08               4.           4.
2800    06   09               8.          -3.
2900    07   10               8.           1.
3000    08   10               2.           5.
3100    09   10               4.           2.
3200    03   02               4.          -4.
3300    03   04              10.          12.
3400    05   06               2.          -3.
3500    07   08               5.          -2.
3600    09   08               6.          10.
3700
3800
*
```

Fig. 4.11

EL NRO. CLAVE DE ESTA SUBRUTINA ES : 46

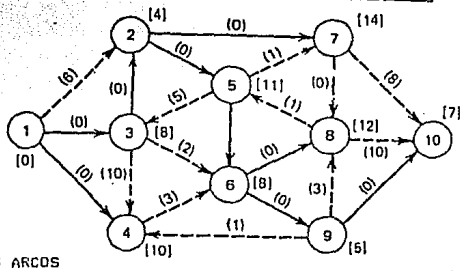
PROGRAMA COMI44
NO BASICO DE RUTA MAS CORTA

EL NODO RAIZ ES 1

REPRESENTACION DE LA RED--

NRO. DE NODOS-- 11 NRO. DE ARCOS-- 21

$(\pi_i + h_k - \pi_j)$
$[\pi_i]$



PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	9
2	-1
3	-1
4	-1
5	-1
6	-1
7	-1
8	-1
9	-1
10	-1
11	0

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	O(K)	T(K)	C(K)	H(K)
1	1	2	10	10
2	1	3	8	8
3	1	4	6	10
4	2	7	20	10
5	2	5	6	7
6	3	6	4	2
7	3	2	4	-4
8	3	4	10	12
9	4	6	8	1
10	5	3	8	2
11	5	7	12	4
12	5	6	2	-3
13	6	8	4	4
14	6	9	8	-3
15	7	10	8	1
16	7	8	5	-2
17	8	5	12	0
18	8	10	2	5
19	9	4	2	6
20	9	10	4	2
21	9	8	6	10

RUTA MAS CORTA DESDE EL NODO RAIZ 1 A TODOS LOS OTROS NODOS

I	PI(I)	PB(I)	O(I)	T(I)
1	0	0	0	0
2	4	7	3	2
3	8	2	1	3
4	10	3	1	4
5	11	5	2	5
6	8	12	5	6
7	14	4	2	7
8	12	13	6	8
9	5	14	6	9
10	7	20	9	10

FIN DEL PROGRAMA RFCLSG

EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES: 0.183333

ET=1150.0 PT=0.3 IO=0.3

Fig. 4.12

4.7 OTRO METODO DE SOLUCION DEL PROBLEMA CUANDO ALGUNOS ARCOS TIENEN COSTOS NEGATIVOS, USANDO UN ALGORITMO DUAL.

Se supone que para alguna red se obtiene el árbol de trayectoria más corta óptima, pero uno o más arcos de este árbol se hacen no admisibles. La solución satisface las condiciones de holgura complementaria pero no es completamente factible. Este algoritmo que procede a cambiar una solución corriente a una nueva solución óptima es clasificado como un algoritmo dual de arco no factible. Este algoritmo es usado también en los problemas de flujo a costo mínimo que se estudiará más adelante. Se sonsidera el árbol inicial $D_T = [N, M_T]$, como un árbol de trayectoria más corta, cuyas condiciones de holgura complementaria son:

$$\Pi_i + h_k = \Pi_j \qquad \text{para cada arco del árbol } k\ (i,j) \in M_T$$

$$\Pi_i + h_k \geq \Pi_j \qquad \text{para cada arco no básico } k\ (i,j).$$

Si uno o más arcos son inadmisibles, éstos deben ser removidos y reemplazados con arcos admisibles, tal que los nuevos resultados del árbol satisfagan las condiciones de holgura complementaria y la factibilidad del dual.

Se selecciona primero un arco inadmisible para que salga de la base y no tenga arcos predecesores inadmisibles. Este es el arco $k_L\ (i_L, j_L)$, el que al salir particiona el árbol básico en dos subárboles $D_1 = [N_1, M_1]$ y $D_2 = [N_2, M_2]$, si los nodos $S \in N_1$ y $j_L \in N_2$, el arco k_L está en la treyectoria desde S a j_L. Luego se necesita elegir un arco $k_E\ (i_E, j_E)$ para que entre a la base tal que la nueva base satisfaga las condiciones de holgura complementaria y mantenga la factibilidad del dual. Durante este proceso el arco entrante debe cumplir con los siguientes requisitos:

1. El arco debe ser admisible.
2. Debe originarse en N_1 y terminar en N_2.
3. De todos los arcos que satisfacen las condiciones 1 y 2, el arco entrante debe tener el menor valor de:

$$d_k = \Pi_i + h_k - \Pi_j$$

Si no hay arcos que satisfacen 1 y 2, no existe árbol generador.

Sean Π_i los potenciales de los nodos en el nuevo árbol. Se define:

$$\Pi'_i = \Pi_i \qquad \text{para } i \in N_1$$

$$\Pi'_i = \Pi_i + d_k \qquad \text{para } i \in N_2$$

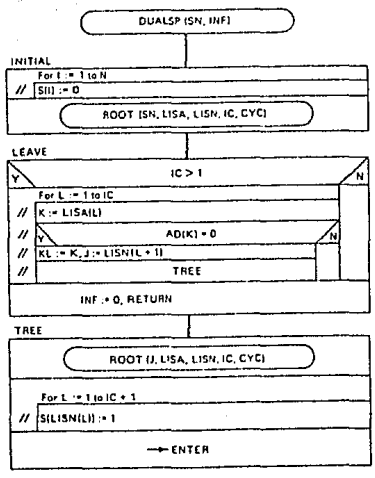
luego las condiciones de holgura complementaria se cumplen para todos los arcos del árbol incluyendo el arco k_E. Para todos los arcos que satisfacen -- las condiciones 1 y 2, se tiene:

$$d_k = \Pi'_i + h_k - \Pi'_j \geq 0$$

por la regla para elegir el arco entrante. Así el nuevo árbol satisface las condiciones de holgura complementaria y la factibilidad del dual En particular algunos de los arcos en la trayectoria desde j_E a j_L pueden ser inadmisibles, puesto que el arco reflejado de un arco admisible no es necesariamente admisible. El proceso continúa para elegir un nuevo arco admisible. En cada iteración el conjunto N_1 se incrementa en un nodo (nodo j_E). Así el -- máximo número de iteraciones es n - 1. Este procedimiento de solución se realiza mediante el algoritmo DUALPS; la secuencia del procedimiento depende -- del orden en el cual los arcos y nodos se encuentran en el listado de ROOT.

ALGORITMO DUALPS.

Propósito: Obtiene un nuevo árbol óptimo de trayectorias más cortas cuando uno o más arcos del árbol de trayectorias más cortas se hacen inadmisibles. El algoritmo comienza con un árbol de trayectoria más corta enraizado en el nodo fuente y con nodos potenciales -- que satisfacen la factibilidad del problema dual y las condiciones de holgura complementaria.

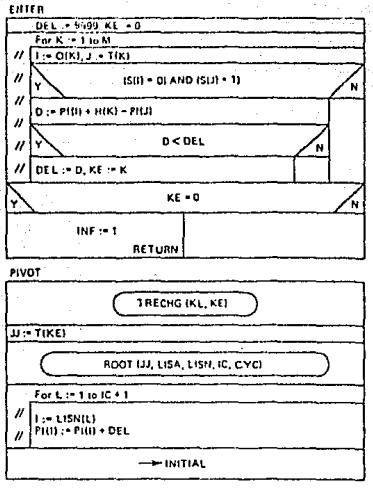


1. (INICIA) Lista los arcos del árbol en orden precedente.
2. (SALE) De los arcos inadmisibles en el árbol selecciona uno que no tiene

predecesores inadmisibles para salir de la base. Si no hay arcos inadmisibles, se detiene con un árbol óptimo. Si encuentra un arco $k_L(i_L, j_L)$, quita este arco y va al paso 3.

3. (ARBOL) Encuentra el conjunto de nodos en el árbol enraizado en el nodo j_L. Esto es, el conjunto N_2 ($j \in N_2$ si $S(j) = 1$).

4. (ENTRA) Encuentra el arco admisible con origen en N_1 y destino en N_2, para entrar al árbol básico, con el valor más pequeño de $d_k = \Pi_i + h_k - \Pi_j$.
Este es el arco $k_E(i_E, j_E)$. Si no hay arco entrante indica no factibilidad.



5. (PIVOTE) Cambia la base del árbol quitando el arco k_L, invirtiendo los arcos en la trayectoria desde el nodo j_L a j_E y añade al arco k_E. Encuentra la lista de arcos y nodos en el árbol enraizado en j_E. Suma d_{k_E} a los potenciales del árbol enraizado en j_E. Va al paso 1.

Este método de solución se ilustra mediante el ejercico de la figura - 4.13. La figura 4.13a es la red original, en la cual sólo los arcos hacia a-

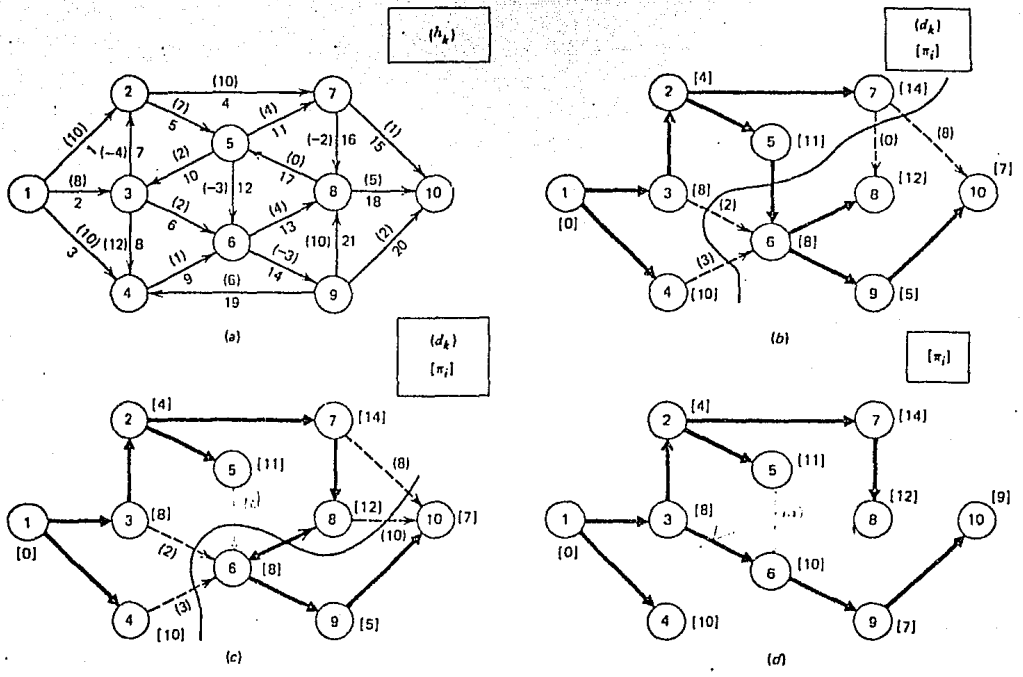


Figura 4.13

delante son admisibles (los arcos reflejados son inadmisibles). En la figura 4.13b se observa que el arco 12(5,6) es inadmisible y sale de la red original formandose dos subárboles definidos por los conjuntos de nodos

$$N_1 = \{1, 2, 3, 4, 5, 7\} \quad y \quad N_2 = \{6, 8, 9, 10\}$$

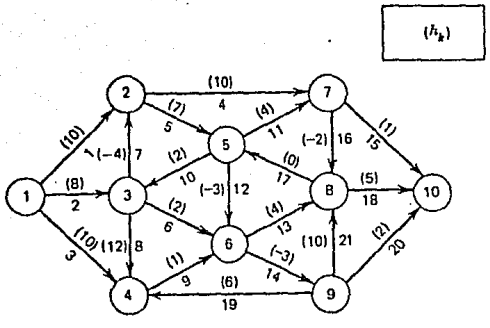
El arco que se origina en N_1 y termina en N_2 con el valor más pequeño -

de d_k es el 16(7,8), el cual al entrar a la base, forma el árbol de la figura 4.13c, con sus respectivos potenciales en los nodos. En este nuevo árbol se observa que el arco -13(8, 6) es inadmisible, el cual al salir, origina dos conjuntos de nodos:

$$N_1 = \{1, 2, 3, 4, 5, 7, 8\} \quad y \quad N_2 = \{6, 8, 9, 10\}$$

El arco que entra a la base es el 6(3, 6) con $d_k = 2$. Finalmente el resultado del árbol en la figura 4.13d incluye solamente arcos admisibles.

En el anexo A.2 se presenta la subrutina COMI47 codificada en FORTRAN, la misma que utiliza el algoritmo dual DUALPS, para resolver la red de la figura 4.13a cuyos datos de entrada y resultados se pueden observar en las figuras 4.14 y 4.15.



100	47				
200	1				
300	10				
400	1				
500	1		9		
600	2		-1		
700	3		-1		
800	4		-1		
900	5		-1		
1000	6		-1		
1100	7		-1		
1200	8		-1		
1300	9		-1		
1400	10		-1		
1500					
1600	01	02		10.	10.
1700	01	03		8.	8.
1800	01	04		6.	10.
1900	02	07		20.	10.
2000	02	05		6.	7.
2100	05	03		8.	2.
2200	03	06		4.	2.
2300	04	06		8.	1.
2400	09	04		2.	6.
2500	05	07		12.	4.
2600	08	05		12.	0.
2700	06	08		4.	4.
2800	06	09		8.	-3.
2900	07	10		8.	1.
3000	08	10		2.	5.
3100	09	10		4.	2.
3200	03	02		4.	-4.
3300	03	04		10.	12.
3400	05	06		2.	-3.
3500	07	08		5.	-2.
3600	09	08		6.	10.
3700					
3800					

Fig. 4.14

EL NRO. CLAVE DE ESTA SUBRUTINA ES : 47

PROGRAMA COMI47
DUAL DE RUTA MAS CORTA

REPRESENTACION DE LA RED--

NRO. DE NODOS-- 11 NRO. DE ARCOS-- 21

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	9
2	-1
3	-1
4	-1
5	-1
6	-1
7	-1
8	-1
9	-1
10	-1
11	0

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	O(K)	T(K)	C(K)	H(K)
1	1	2	10	10
2	1	3	8	8
3	1	4	6	10
4	2	7	20	10
5	2	5	6	7
6	3	6	4	2
7	3	2	4	-4
8	3	4	10	12
9	4	6	8	1
10	5	3	8	2
11	5	7	12	4
12	5	6	2	-3
13	6	8	4	4
14	6	9	8	-3
15	7	10	8	1
16	7	8	5	-2
17	8	5	12	0
18	8	10	2	5
19	9	4	2	6
20	9	10	4	2
21	9	8	6	10

RUTA MAS CORTA DESDE EL NODO RAIZ 1
AL RESTO DE NODOS CYC= 0 J= 8

I	PI(I)	PB(I)	O(I)	T(I)
1	0	0	0	0
2	4	7	3	2
3	8	2	1	3
4	10	3	1	4
5	11	5	2	5
6	8	12	5	6
7	14	4	2	7
8	12	16	7	8
9	5	14	6	9
10	7	20	9	10

```
RUTA MAS CORTA DESDE EL NODO RAIZ  1
AL RESTO DE NODOS       CYC=  0 J=    8

     I       PI(I)       PB(I)       O(I)       T(I)

     1           0           0          0          0
     2           4           7          3          2
     3           8           2          1          3
     4          10           3          1          4
     5          11           5          2          5
     6          10           6          3          6
     7          14           4          2          7
     8          12          16          7          8
     9           7          14          6          9
    10           9          20          9         10

FIN DEL PROGRAMA RFCLSG

EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES:  0.266667
```

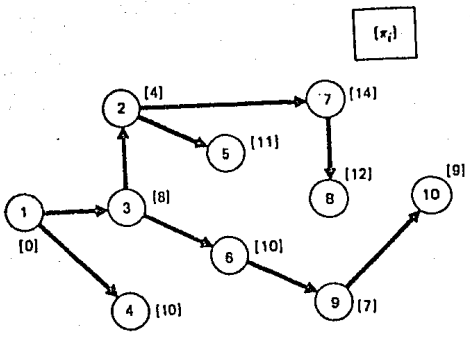


Fig. 4.15

CAPITULO V

EL PROBLEMA DE FLUJO MAXIMO

5.1 Introducción

5.2 Representación matemática del problema

5.3 Conceptos sobre incrementos de flujo

5.4 Solución del problema mediante el uso del algoritmo no básico

5.5 Solución del problema mediante el uso delalgoritmo básico.

5.1 INTRODUCCION

Este capítulo estudia el problema de flujo máximo. Los algoritmos que se introducen son usados para encontrar el flujo máximo o un flujo requerido V_r, que puede pasar desde un nodo fuente s hasta un nodo sumidero t, el cual no debe exceder al flujo máximo.

Para resolver el problema se utilizan un algoritmo no básico FPATH1, el que durante su proceso iterativo no mantiene un árbol básico, y un algoritmo básico FPATH2 que mantiene un árbol básico y sus procedimientos computacionales son más complejos.

5.2 REPRESENTACION MATEMATICA DEL PROBLEMA

Dada una red dirigida D=[N,M] con un parámetro capacidad en cada arco, se considera el problema de encontrar el flujo máximo total en la red, desde el nodo fuente s al nodo sumidero t. La figura 5.1 ilustra un ejemplo del problema. Puede establecerse como un problema de programación lineal en la siguiente forma:

$$\text{Max. } v \tag{1}$$

$$\text{s. a. } \sum_{k\epsilon M_{oi}} f_k - \sum_{k\epsilon M_{Ti}} f_k = 0 \qquad i\epsilon N-(s,t) \tag{2a}$$

$$\sum_{k\epsilon M_{os}} f_k - \sum_{k\epsilon M_{Ts}} f_k - v = 0 \tag{2b}$$

$$\sum_{k\epsilon M_{ot}} f_k - \sum_{k\epsilon MT_t} + v = 0 \tag{2c}$$

$$0 \leq v \leq v_r \tag{3}$$

$$0 \leq f_k \leq c_k \qquad k\epsilon M \tag{4}$$

El problema dual es:

$$\text{Min. } \sum_{k=1}^{m} c_k \delta_k + v_r \delta m+1 \tag{5}$$

$$\text{s.a. } \Pi_i - \Pi_j + \delta_k \geq 0 \quad \text{para } k(i,j)\epsilon M \tag{6}$$

$$\Pi_t - \Pi_s + \delta_{m+1} \geq 1$$

$$\Pi_i \text{ no restringido} \quad \text{para todo } i \qquad (7)$$

$$\delta k \geq 0 \qquad k = 1, 2, \ldots, m+1 \qquad (8)$$

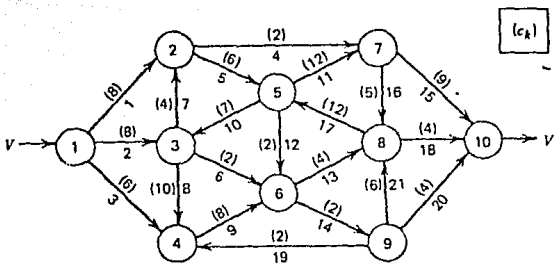


Fig. 5.1

(s = 1, t = 10)

La ecuación (3) puede salir desde las restricciones, permitiendo que v_r sea un número grande (esto implica un flujo inalcanzable). También la ecuación de conservación de flujo para el nodo s es redundante y sale de las restricciones.

Es conveniente desde un punto de vista conceptual, considerar a δ_k como una "variable indefinida". Si en la solución óptima, $\delta_k > 0$, el arco k no es el único elemento del conjunto de arcos que limitan el flujo máximo en la red. Considerando esta situación, se establece que el problema dual selecciona un conjunto de arcos de capacidad mínima total, que satisfacen las restricciones dual. Las restricciones dual solo permiten conjuntos de arcos, los cuales al salir de la red eliminan todas las rutas -

desde el nodo s al noto t, El conjunto de arcos con estas características se llama un corte, El corte con la capacidad total mínima es el corte mínimo. Se podrá notar que el valor del máximo flujo es igual a la capacidad del corte mínimo. Los algoritmos que se estudian en este capítulo resuelven los dos problemas simultaneamente, el de corte mínimo y el de flujo máximo.

Se consideran dos casos para la solución óptima del problema primal: cuando $v < v_r$ y cuando $v = v_r$.

En ambos casos los siguientes puntos son válidos:

1. Si la capacidad c_k para cada arco y el limite v_r son enteros, una solución básica para el problema primal es entera (todos los flujos tienen valores enteros).
2. Sin considerar que c_k y v_r son enteros, una solución básica del problema dual es entera (todos los Π_i y δ_k tienen valores enteros).
3. Hay una solución óptima dual para la cual todos los potenciales en los nodos son 0 o 1 ($\Pi_i=0$ o $\Pi_i=1$).
4. De las condiciones de holgura complementaria, se obtiene las condiciones de optimalidad. Para cada arco $k(i,j)\varepsilon M$, tenemos:

$$\text{Si } \Pi_i - \Pi_j > 0 \quad \text{luego } f_k = 0$$

$$\text{Si } \Pi_i - \Pi_j < 0 \quad \text{luego } f_k = c_k$$

$$\text{Si } \Pi_i - \Pi_j = 0 \quad \text{luego } 0 \leq f_k \leq c_k.$$

cuando $v = v_r$ se obtiene el limite superior de flujo y los siguientes puntos son válidos:

5. Hay una solución dual óptima para la cual todos los potenciales en los nodos son cero.
6. Una solución básica óptima forma un árbol generador dirigido en una red expandida, enraizada en el nodo fuente.

Cuando $v < v_r$, se obtiene el flujo máximo desde s a t y los siguientes puntos son válidos:

7. Los nodos pueden dividirse en dos conjuntos, N_1 y N_2 con

$$s\varepsilon N_1, \quad t\varepsilon N_2 \quad N_1 \cup N_2 = N \quad \text{y} \quad N_1 \cap N_2 = \phi$$

tal que:

$$\Pi_i = \begin{cases} 0 & \text{si } i \epsilon N_1 \\ 1 & \text{si } i \epsilon N_2 \end{cases} \qquad y \qquad \delta_k(i,j) = \begin{cases} 1 & i \epsilon N_1,\ j \epsilon N_2 \\ 0 & \text{otros} \end{cases}$$

para una solución dual optima. Los arcos que se originan en N_1 y terminan en N_2 forman el corte mínimo.

8. En la solución primal óptima, el flujo es igual a la capacidad para los arcos en el corte mínimo.

$$f_k(i,j) = c_k(i,j) \quad \text{para } i \epsilon N_1,\ j \epsilon N_2$$

y para los arcos que pasan desde N_2 a N_1, el flujo es cero

$$f_k(i,j) = 0 \quad \text{para } i \epsilon N_2,\ j \epsilon N_1$$

El valor del máximo flujo es igual a la capacidad del corte mínimo:

$$\sum_{\substack{i \epsilon N_1 \\ j \epsilon N_2}} C_k(i,j) = v$$

9. Una solución básica óptima consiste de la variable v y dos subárboles, uno enraizado en el nodo s y el otro enraizado en el nodo t.

5.3 CONCEPTOS SOBRE INCREMENTOS DE FLUJO

Considere la red marginal $D^* = (N, M^*)$ con la función de admisibilidad definida como:

$$A_d(k) = 1 \begin{cases} \text{Si } k>0 \quad y \quad f_k < c_k \\ \text{Si } k<0 \quad y \quad f_{-k} > 0 \end{cases}$$

$$A_d(k) = 0 \begin{cases} \text{Si } k>0 \quad y \quad f_k = c_k \\ \text{Si } k<0 \quad y \quad f_{-k} = 0 \end{cases} \qquad (9)$$

Sea $D_p=[N_p,M_p]$ una trayectoria dirigida desde s a t en D*. Luego M_p puede contener tantos arcos admisibles hacia adelante - como arcos admisibles reflejados. Se definen nuevos flujos en la red expandida, tal como:

$$f'_k = f_k \qquad \text{si } k \notin M_p \text{ y } -k \notin M_p$$

$$f'_k = f_k + \Delta \qquad \text{para } k \in M_p,\ k>0$$

$$f'_{-k} = f_{-k} - \Delta \qquad \text{para } k \in M_p,\ k<0 \qquad (10)$$

$$v = v + \Delta$$

donde f_k es el flujo en el arco k antes del cambio de flujo, f'_k es el flujo después del cambio de flujo, y Δ es la cantidad del cambio de flujo.

Se debe notar que los arcos reflejados son solamente una - construcción conceptual usada en la representación de un tipo - particular de trayectoria desde el nodo fuente al nodo sumidero.

Cuando aumenta el flujo en el arco hacia delante, el flujo en su respectivo arco reflejado disminuye. El nuevo flujo es factible para el problema de flujo máximo si se cumple:

$$0 \le f'_k \le c_k$$

lo cual se asegura con:

$$\Delta = \text{Min.}\left\{ \underset{k\in\{M_p \mid k>0\}}{\text{Min.}} (C_k - f_k) \ , \quad \underset{k\in\{M_p \mid k<0\}}{\text{Min.}} f_{-k} \right\}$$

Si $v < v_r$ y existe una trayectoria dirigida en la red marginal, el flujo en la red no puede ser máximo. Esta trayectoria dirigida en la red marginal es llamda trayectoria aumentada. El procedimiento de solución inicia con algún flujo factible en la red con $v < v_r$ y busca una trayectoria aumentada, si no lo encuentra, el flujo v es máximo desde s a t. Si encuentra una trayectoria, el flujo es aumentado de acuerdo a las ecuaciones (10). - Este proceso continua hasta que se obtiene el flujo v_r o hasta - cuando ya no se encuentra una trayectoria.

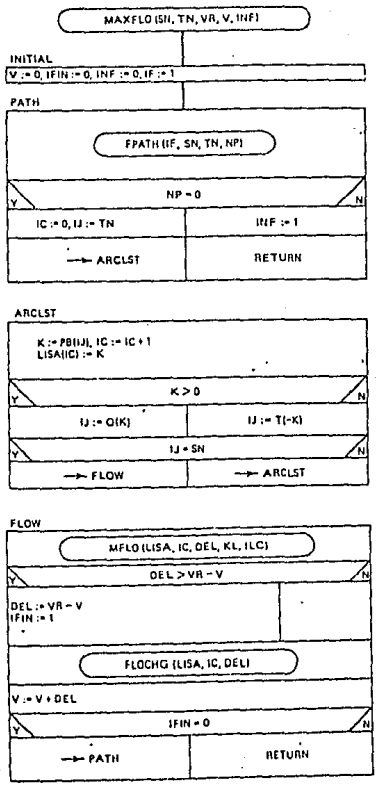
5.4 SOLUCION DEL PROBLEMA MEDIANTE EL USO DEL ALGORITMO NO BASICO.

Los problemas de un flujo específico y de flujo máximo, pueden ser resueltos por algoritmos que no requieren de un árbol básico y se caracterizan por tener un conjunto de arcos cuyos flujos se encuentran estrictamente entre los límites ($0<f_k<c_k$), en la solución no básica óptima. Estos algoritmos son LABEL1, llamado por el algoritmo FPATH1, es similar el algoritmo DIJKSTRA de trayectoria más corta, y además el algoritmo MAXFLO que encuentra el flujo en la red. Se construye un conjunto de nodos s, el cual inicialmente esta constituido solo del nodo fuente. Un nodo es adicionado al conjunto cuando se encuentra un arco admisible que se origina en un nodo en s y termina en otro nodo que no esta en s. Se encuentra una ruta formada por arcos admisibles y el proceso termina cuando el nodo t es adicionado al conjunto s.

ALGORITMO MAXFLO

PROPOSITO: Encuentra la red para la cual se obtiene un flujo dado VR del nodo fuente SN al nodo sumidero TN. Si VR no puede ser obtenido, el flujo máximo de SN a TN es proporcionado. Este proceso inicia con un flujo factible.

1. (INICIAR) hace V:=0
2. (TRAYECTORIA) Encuentra una trayectoria desde el nodo fuente al nodo sumidero, la cual consiste de arcos admisibles, es decir, arcos hacia adelante con flujo menor que su capacidad, o arcos reflejados con flujo positivo sobre su correspondiente arco hacia adelante. Si no encuentra ninguna trayectoria, termina con el flujo máximo desde el nodo fuente al sumidero. De otra manera va al paso 3.



3. (LISTA DE ARCOS) Forma la lista de arcos en la trayectoria - desde SN a TN usando los apuntadores hacia atrás.
4. (FLUJO) Encuentra el máximo incremento de flujo (DEL) que pue de ser enviado en la trayectoria. Si DEL excede a VR-V, incre menta el flujo por esta cantidad y termina. De lo contrario - incrementa el flujo en la trayectoria y retorna al paso 2.

Los principales pasos en este algoritmo son:

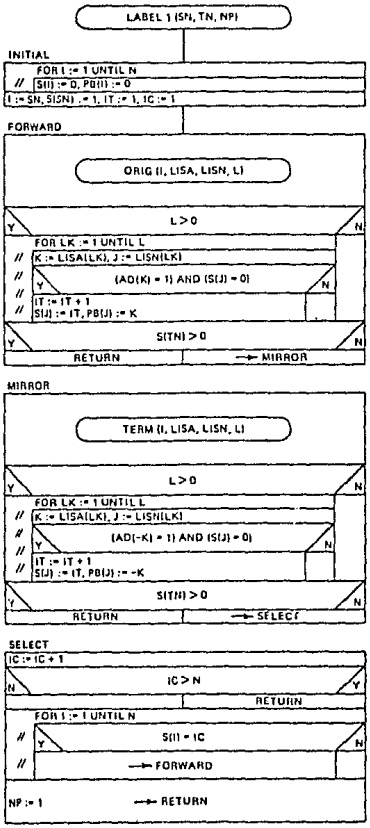
1. Encontrar una trayectoria aumentada
2. Determinar el aumento de flujo máximo posible.
3. Aumentar el flujo por esta cantidad.

Sea la solución básica o no básica, depende del procedimiento para encontrar la trayectoria. LABEL1 es usado cuando no se requiere una solución básica y LABEL2 es usado para obtener una solución básica.

ALGORITMO LABEL1

PROPOSITO: Encontrar una trayectoria desde el nodo SN al nodo TN usando solo arcos admisibles, es decir arcos hacia adelante con flujo menor que su capadidad, o arcos reflejados con flujo positivo sobre su correspondiente arco hacia adelante.

1. (INICIAR) El conjunto S de nodos - etiquetados es vacio; s_i=0. Los apuntadores hacia atrás son inicializados en cero; P_{Bi}:=0, luego el nodo fuente SN es incluido en S con la - etiqueta S(SN):=1. Los contadores - de etiquetas e iteraciones son inicializados a 1: I_c:=1, I_t:=1.
2. (ADELANTE) Encuentra el conjunto de arcos originados en el nodo i. Si el arco k(i,j) está en el conjunto y es admisible, el nodo j no esta etique-

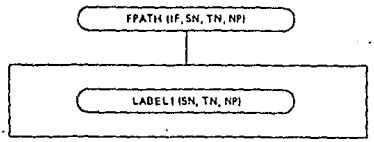


tado, entonces etiqueta al nodo j. Incrementa el contador de iteraciones $I_t := I_t + 1$ y efectúa las siguientes asignaciones: - $s_j := I_t$ y $P_{Bj} := k$.

3. (REFLEJADO) Encuentra el conjunto de arcos que terminan en el nodo i. Si el arco k(i,j) está en el conjunto, -k es admisible y j no está etiquetado, entonces etiqueta el nodo j. Hace $I_t := I_t + 1$; hace $s_j = I_t$ y $P_{Bj} = -k$. Si el nodo TN es etiquetado, - una trayectoria ha sido encontrada y termina. De lo contrario va al paso 4.
4. (SELECCIONA) El contador de etiquetas es incrementado $I_c := I_c + 1$. Selecciona el nodo con la etiqueta $s_j = I_c$. Si es encontrado uno, este nodo es i y va al paso 2. Si no hay tal nodo, no - existe la trayectoria desde SN a TN; termina.

ALGORITMO FPATH1

PROPOSITO: Llama a LABEL1 para el algoritmo no básico de flujo máximo.



La figura 5.2 muestra las iteraciones realizadas por el algoritmo no básico MAXFLO, partiendo desde la red original de la figura 5.1 con $v_r = 10$. Note que, cuando el nodo i es etiquetado, el - valor de s(i) es igual al número de nodos etiquetados, por lo - que los valores de s(i) nos dicen el orden en que los nodos fueros etiquetados.

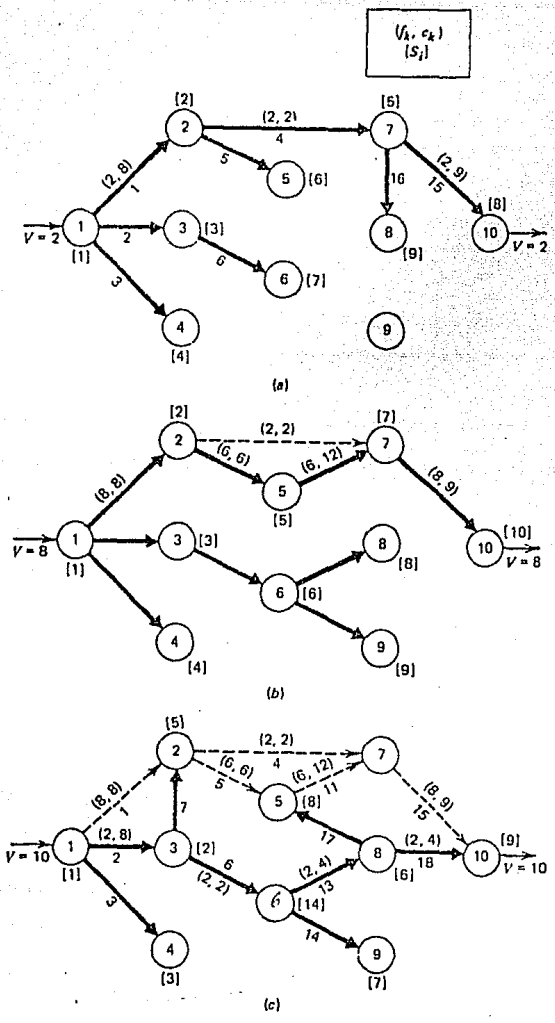


Fig. 5.2

Red con s=1, t=10 y $v_r = 10$

Una vez que el nodo t(nodo 10) es etiquetado, se asegura la existencia de una trayectoria aumentada formada por arcos admisibles. Por esta razón el nodo 9 no es etiquetado durante la iteración inicial, como se observa en la figura 5.2a. Partiendo del nodo 10 los apuntadores hacia atrás facilitan la identificación de los arcos 15, 4 y 1 como los miembros de la trayectoria aumentada. La -

capacidad del arco 4 limita la cantidad adicional de flujo a dos unidades. Con dos iteraciones más en suficiente para obtener el flujo requerido de 10; como se puede observar en las figuras -- 5.2b y 5.2c.

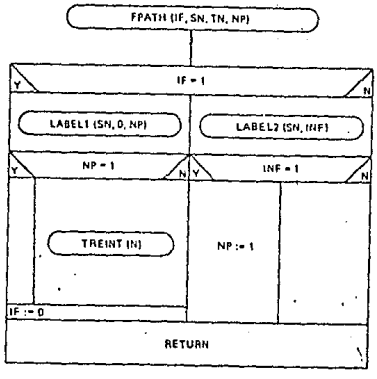
En el anexo A.2 se encuentra la subrutina principal FLMA54 codificada en FORTRAN, la misma que utiliza el algoritmo no básico MAXFLO con los algoritmos FPATH1 y LABEL1, para resolver la - red de figura 5.1 con v_r=14, cuyos datos de entrada y resultados se pueden observar en las figuras 5.3 y 5.4, respectivamente.

5.5 SOLUCION DEL PROBLEMA MEDIANTE EL USO DEL ALGORITMO BASICO.

Este algoritmo durante su proceso mantiene un árbol básico y utiliza una variante del algoritmo DUALSP para encontrar un aumento de flujo en la trayectoria en cada iteración después de la primera iteración. Este es implementado en el algoritmo FPATH2.

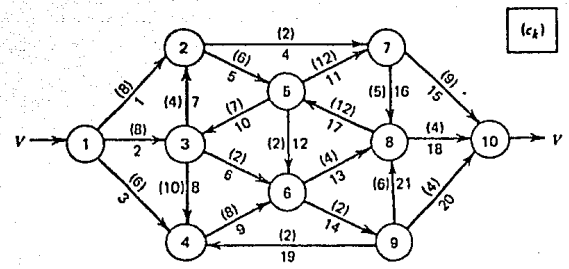
ALGORITMO FPATH2

PROPOSITO: Procede a encontrar la trayectoria para el algoritmo de trayecto más corta cuando se require una solución básica.



La primera vez el algoritmo usa el procedimiento de etiquetación para encontar un árbol generador formado de arcos admisibles. Si el árbol expandido no existe, termina. De otra manera obtiene los apuntadores para la representación del árbol. Después de la primera vez, usa el procedimiento dual para quitar y adicionar arcos admisibles al árbol básico.

En la primera iteración el algoritmo LABEL1 es usado para encontrar un árbol generador de arcos admisibles enraizado en el nodo fuente. Con el aumento del flujo desde el nodo fuente al nodo sumidero en una trayectoria definida por el árbol, uno o más arcos se vuelven inadmisibles. Estos son descubiertos y salen del árbol, mientras los arcos admisibles entran para formar el nuevo árbol -




```
100     54
200     10
300
400     01    02              8.
500     01    03             8.
600     01    04             6.
700     02    07              2.
800     02    05             6.
900     05    03             7.
1000     03    06              2.
1100     04    06              8.
1200     09    04              2.
1300     05    07             12.
1400     08    05             12.
1500     06    08              4.
1600     06    09              2.
1700     07    10              9.
1800     08    10              4.
1900     09    10              4.
2000     03    02              4.
2100     03    04             10.
2200     05    06              2.
2300     07    08              5.
2400     09    08              6.
2500
2600      1    10    99      0
2700
2800
*
```

Fig. 5.3

EL NRO. CLAVE DE ESTA SUBRUTINA ES : 54

PROGRAMA FLHA54
NO BASICO DE FLUJO MAXIMO

REPRESENTACION DE LA RED--

NRO. DE NODOS-- 11 NRO. DE ARCOS-- 21

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0

PARAMETROS TRANSFORMADOS DE LOS ARCOS

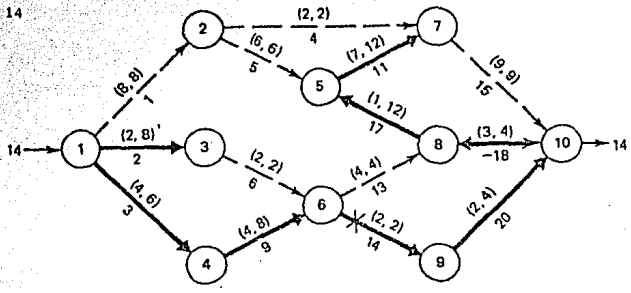
ARCO K	O(K)	T(K)	C(K)	H(K)
1	1	2	8	0
2	1	3	8	0
3	1	4	6	0
4	2	7	2	0
5	2	5	6	0
6	3	6	2	0
7	3	2	4	0
8	3	4	10	0
9	4	6	8	0
10	5	3	7	0
11	5	7	12	0
12	5	6	2	0
13	6	8	4	0
14	6	9	2	0
15	7	10	9	0
16	7	8	5	0
17	8	5	12	0
18	8	10	4	0
19	9	4	2	0
20	9	10	4	0
21	9	8	6	0

SOLUCION DEL MAX. FLUJO. FLUJO REQUERIDO 99

DESDE EL NODO FUENTE 1 AL NODO SUMIDERO 10

NO EXISTE RUTA ADICIONAL
FLUJO MAXIMO POSIBLE = 14

NODO I	P(I)	PB(I)
1	0	0
2	0	7
3	0	2
4	0	3
5	0	17
6	0	9
7	0	11
8	0	-18
9	0	14
10	0	20



ARC K	O(K)	T(K)	FLUJO	CAPACIDAD	COST/UNI.	COST.FLUJO
1	1	2	8	8	0	0
2	1	3	2	8	0	0
3	1	4	4	6	0	0
4	2	7	2	2	0	0
5	2	5	6	6	0	0
6	3	6	2	2	0	0
7	3	2	0	4	0	0
8	3	4	0	10	0	0
9	4	6	4	8	0	0
10	5	3	0	7	0	0
11	5	7	7	12	0	0
12	5	6	0	2	0	0
13	6	8	4	4	0	0
14	6	9	2	2	0	0
15	7	10	9	9	0	0
16	7	8	0	5	0	0
17	8	5	1	12	0	0
18	8	10	3	4	0	0
19	9	4	0	2	0	0
20	9	10	2	4	0	0
21	9	8	0	6	0	0

COSTO TOTAL = 0

FIN DEL PROGRAMA RFCLSG

EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES: 0.316667

$ET=3:04.2 PT=0.5 IO=0.2

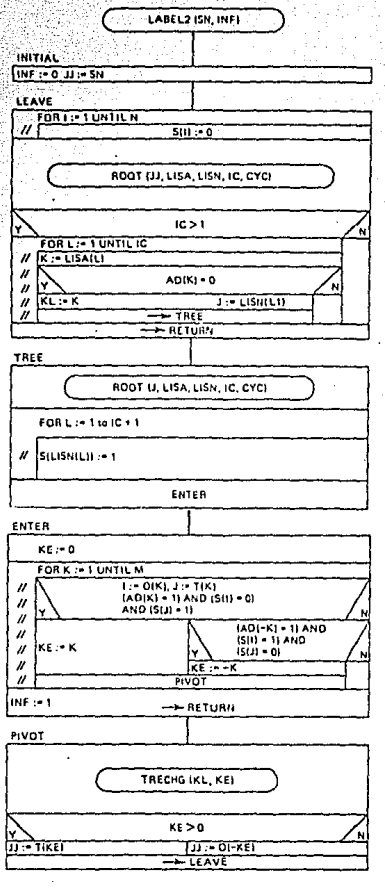
Fig. 5.4

básico. El procedimiento continúa hasta cuando el nodo sumidero se obtiene el flujo especificado v_r o el flujo máximo.

ALGORITMO LABEL2

PROPOSITO: Proporciona un nuevo árbol generador de arcos admisibles después de que uno o más arcos en el árbol original se hacen inadmisibles.

1. (INICIAR) Establece la condición inicial JJ:=SN
2. (SALE) Inicializa el conjunto S en cero, S(I):=0. Lista los nodos y los arcos en orden precedente. Encuentra un arco inadmisible k_L que no tiene predecesores inadmisibles Este arco sale de la base.
3. (ARBOL) Encuentra el conjunto de nodos en el árbol dirigido con raíz en el nodo terminal del arco k_L que sale de la base. Hace S(1) = 1 para los elementos de este conjunto.
4. (ENTRA) Inspecciona la lista de arcos. Si un arco hacia adelante k(i, j) es admisible, tal que $i \epsilon \bar{S}$ y $j \epsilon S$, entonces el arco k entra a la base y va al paso 5. Si un arco reflejado -k(i,j) es admisible, tal que $j \epsilon \bar{S}$ e $i \epsilon S$, entonces el arco -k entra a la base y va al paso 5. De otra manera va al próximo arco. Si no hay arcos admisibles que puedan entrar a la base, termina; el máximo flujo ha sido encontraro.
5. (PIVOTE) Sea $k_E(i,j)$ el arco que entra a la base. Cambia el árbol al incluir k_E y quitar k_L. Encuentra el nodo terminal de k_E y hace JJ=t(k_E). Va al paso 2.



El ejemplo de la figura 5.5 ilustra las aplicaciones del algoritmo básico para obtener el flujo máximo desde el nodo 1 al nodo 10. En las figuras del ejemplo se puede observar que los arcos en líneas sólidas constituyen los arcos básicos del árbol.

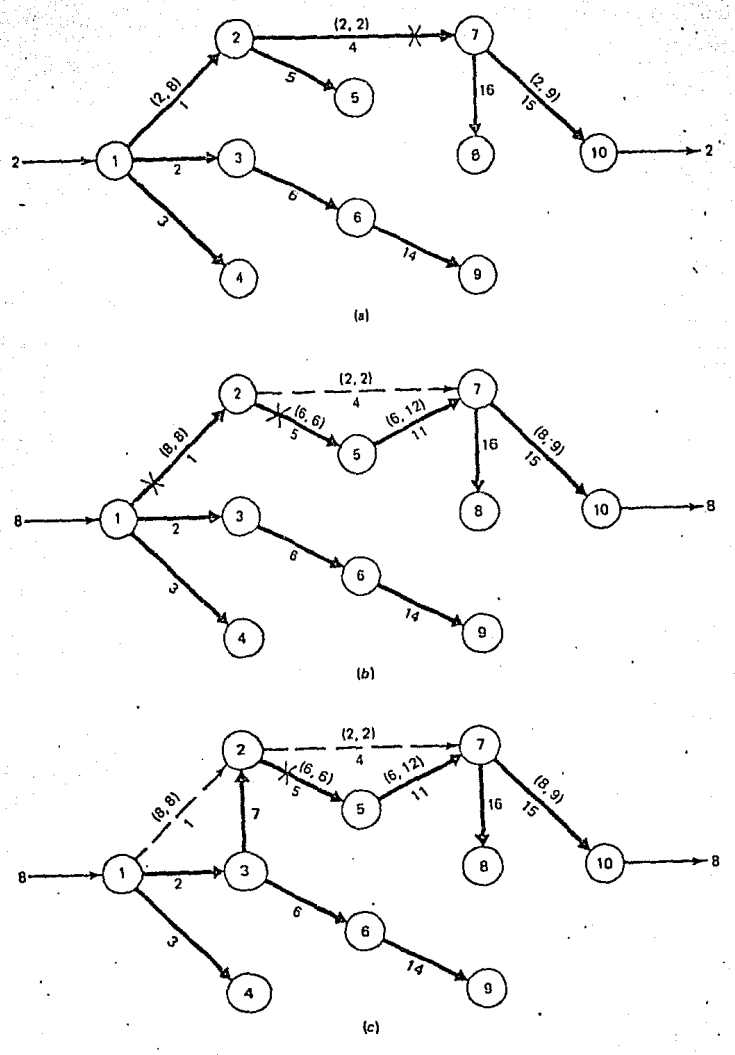


Figura 5.5
(algoritmo básico v_r=99)

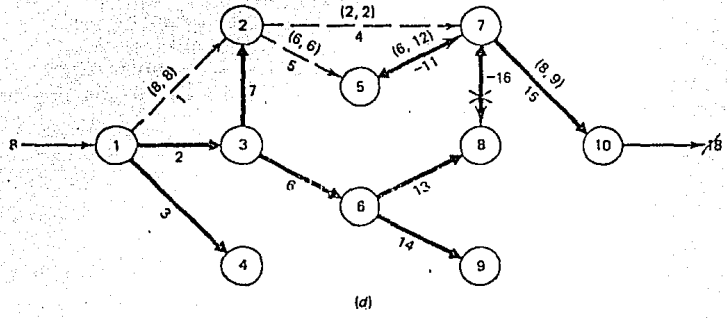


Figura 5.5 (continuación)

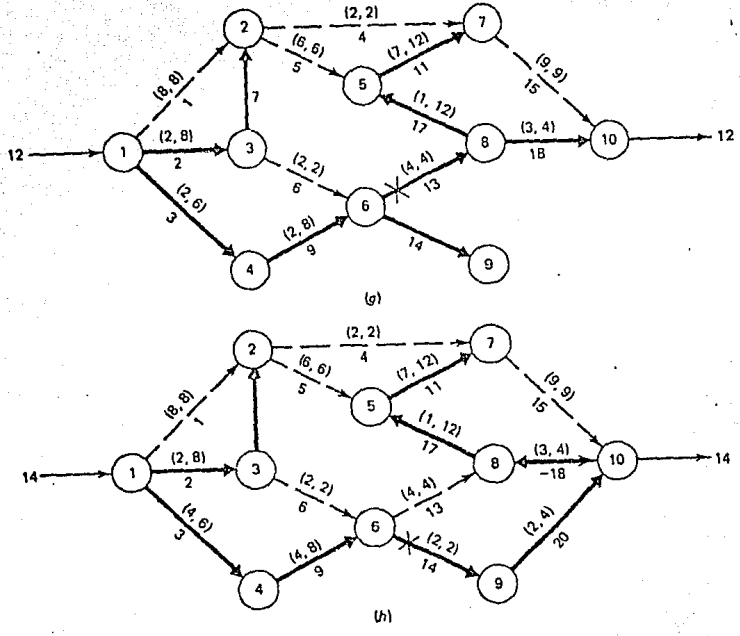
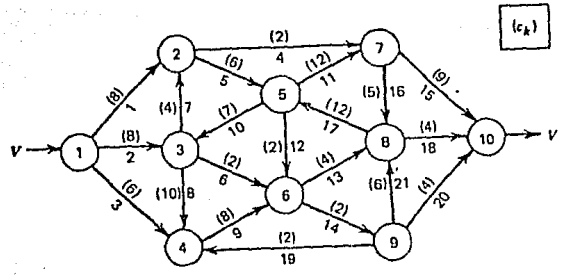


Figura 5.5 (continuación)

Las cabezas gruesas de las flechas indican la orientación de los arcos en la base, las lineas punteadas indican los arcos no básicos con el flujo igual a la capacidad del arco, y los arcos no básicos con el flujo igual a cero no aparecen. La figura 5.5a da el árbol expandido inicial obtenido por LABEL1. El aumento de flujo en dos unidades a través de los arcos 1,5,11 y 15, hace -

que el arco 4 se vuelva inadmisible. Removiendo el arco 4 y añadiendo el arco 11, se obtiene el árbol de la figura 5.5b, en el cual todos sus arcos son admisibles. Esta nueva trayectoria permite incrementar el flujo en seis unidades a través de los arcos 1,5, 11 y 15. Este aumento hace que los arcos 1 y 5 se vuelvan inadmisibles. Aquí se emplea el criterio del arco con el menor indice para que salga del árbol por tanto el arco 1 sale de la base y se añade el arco 7. Los resultados de este árbol se pueden observar en la figura 5.5c. Después el arco 5 es inadmisible, no se puede enviar flujo adicional a través de los arcos 2,7,5,11 y 15. Al cambiar el arco 5 por el 13 resulta una base degenerada, como se puede ver en la figura 5.5d. Desafortunadamente el arco 16 tiene flujo cero; por tanto su arco reflejado (-16) es inadmisible y no se puede enviar flujo adicional a través de los arcos 2,6,13,-16 y 15. Removiendo el arco -16 y añadiendo el arco 17, se puede aumentar el flujo en 1 unidad a través de los arcos 2,6,13,17,11 y 15, como se puede observar en la figura 5.5e. Luego el arco 15 se satura y sale del árbol, entrando en su lugar el arco 18; pudiendo incrementarse el flujo en 1 unidad, a través de los arcos 2,6,13 y 18. El arco 6 sale por volverse inadmisible y entra el arco 9, por tanto todos los arcos son admisibles y podemos incrementar el flujo en dos unidades a través de los arcos 3,9,13 y 18, como se observa en la figura 5.5g. El arco 13 sale por volverse inadmisible y entra el arco 20, luego todos los arcos son admisibles pudiendo incrementar el flujo en dos unidades a través de los arcos 3,9,14 y 20. El 14 se vuelve inadmisible, al remover este arco, no se encuentra una nueva base, por tanto no existen arcos admisbles, y el máximo flujo desde el nodo 1 al nodo 10 ha sido encontrado.

En el anexo A-2 se encuentra la subrutina principal FLMA55 codificada en FORTRAN, la misma que utiliza el algoritmo básico MAXFLO con los algoritmos FPATH2 y LABEL2, para resolver la red de la figura 5.1, con $v_r=14$, cuyos datos de entrada y resultados se pueden observar con las figuras 5.6 y 5.7, respectivamente.




```
100     55
200     10
300
400     01    02              8.
500     01    03              8.
600     01    04              6.
700     02    07              2.
800     02    05              6.
900     05    03              7.
1000     03    06             2.
1100     04    06             8.
1200     09    04             2.
1300     05    07             12.
1400     08    05             12.
1500     06    08              4.
1600     06    09             2.
1700     07    10             9.
1800     08    10             4.
1900     09    10             4.
2000     03    02             4.
2100     03    04             10.
2200     05    06             2.
2300     07    08             5.
2400     09    08             6.
2500
2600      1    10    99       0
2700
2800
4
```

Fig. 5.6

EL NRO. CLAVE DE ESTA SUBRUTINA ES : 55

PROGRAMA FLMA55
BASICO DE FLUJO MAXIMO

REPRESENTACION DE LA RED--

NRO. DE NODOS-- 11 NRO. DE ARCOS-- 21

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0

PARAMETROS TRANSFORMADOS DE LOS ARCOS

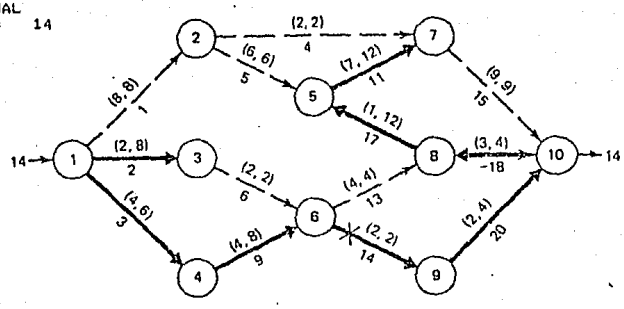
ARCO K	O(K)	T(K)	C(K)	H(K)
1	1	2	8	0
2	1	3	8	0
3	1	4	6	0
4	2	7	2	0
5	2	5	6	0
6	3	6	2	0
7	3	2	4	0
8	3	4	10	0
9	4	6	8	0
10	5	3	7	0
11	5	7	12	0
12	5	6	2	0
13	6	8	4	0
14	6	9	2	0
15	7	10	9	0
16	7	8	5	0
17	8	5	12	0
18	8	10	4	0
19	9	4	2	0
20	9	10	4	0
21	9	8	6	0

SOLUCION DEL MAX. FLUJO. FLUJO REQUERIDO 99

DESDE EL NODO FUENTE 1 AL NODO SUMIDERO 10

TODOS LOS FLUJOS INICIALES SON CERO
NO EXISTE RUTA ADICIONAL
MAXIMO FLUJO POSIBLE = 14

NODO I	P(I)	PD(I)
1	0	0
2	0	7
3	0	2
4	0	3
5	0	17
6	0	9
7	0	11
8	0	-18
9	0	14
10	0	20



ARC K	O(K)	T(K)	FLUJO	CAPACIDAD	COST/UNI.	COST.FLUJO
1	1	2	8	8	0	0
2	1	3	2	8	0	0
3	1	4	4	6	0	0
4	2	7	2	2	0	0
5	2	5	6	6	0	0
6	3	6	2	2	0	0
7	3	2	0	4	0	0
8	3	4	0	10	0	0
9	4	6	4	8	0	0
10	5	3	0	7	0	0
11	5	7	7	12	0	0
12	5	6	0	2	0	0
13	6	8	4	4	0	0
14	6	9	2	2	0	0
15	7	10	9	9	0	0
16	7	8	0	5	0	0
17	8	5	1	12	0	0
18	8	10	3	4	0	0
19	9	4	0	2	0	0
20	9	10	2	4	0	0
21	9	8	0	6	0	0

COSTO TOTAL = 0

FIN DEL PROGRAMA RFCLSG

EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES: 0.316667

#ET=3:06.0 PT=0.5 ID=0.2

Fig. 5.7

CAPITULO VI

EL PROBLEMA DE FLUJO A COSTO MINIMO.

6.1 Introducción.

6.2 Representación matemática del problema.

6.3 Método de flujo máximo para obtener una solución primal factible.

6.4 Método del arco artificial para obtener una solución primal factible.

6.5 Solución del problema mediante el uso - de un algoritmo primal no básico.

6.6 Solución del problema mediante el uso - de un algoritmo primal básico.

6.1 INTRODUCCION.

En este capítulo se resolverá el problema de flujo a costo mínimo, con costos lineales, mediante los algoritmos primales - no básico y básico.

Los algoritmos primales consideran tres pasos:

1. Encuentran una red de flujo inicial que satisface la -- factibilidad primal.
2. Determinan si la red de flujo es óptima. Si el flujo - es óptimo termina. De lo contrario va al paso 3.
3. Modifica la red de flujo cambiando los flujos en un ciclo y regresa al paso 2.

Para resolver este problema primeramente se encuentra una solución primal factible mediante los algoritmos de flujo máximo PHASE1 y del arco artificial ARTIFIC; después se emplean los algoritmos primal no básico PRIMAL1 y primal básico PRIMAL, para obtener la solución óptima.

6.2 REPRESENTACION MATEMATICA DEL PROBLEMA.

El problema de flujo a costo mínimo, puede ser expresado - como un problema de programación lineal en la siguiente forma:

$$\text{Min.} \quad \sum_{k=1}^{m} h_k f_k$$

$$\text{s.a.} \quad \sum_{k \in Mo_i} f_k - \sum_{k \in M_{Ti}} f_k = b_i \qquad i = 1, \ldots, n-1$$

$$f_k \leqslant c_k \qquad k = 1, \ldots, m$$

$$f_k \geqslant 0 \qquad k = 1, \ldots, m$$

El problema dual es:

$$\text{Min.} \quad \sum_{i=j}^{n-1} \Pi_i b_i + \sum_{k=1}^{m} \delta_k C_k$$

$$\text{s.a.} \quad \Pi_i - \Pi_j + \delta_k \geq - h_k \qquad \text{para} \quad k(i,j) \in M$$

$$\Pi_i \quad \text{no restringido} \qquad i=1, \ldots, n-1$$

$$\delta_k \geq 0 \qquad k \in M$$

Las condiciones de óptimalidad pueden ser escritas directamente desde el teorema 3. Por simplicidad se define:

$$d_k = \Pi_i - \Pi_j + h_k \qquad \text{para } k(i,j)$$

1. Factibilidad primal.

   (a) Conservación de flujo en cada nodo.
   (b) $0 \leq f_k \leq C_k$ para todos los arcos k.

2. Factibilidad restringida del dual.
   $d_k = \text{Máx} \ [0, -d_k]$

3. Condiciones de holgura complementaria.

   (a) $d_k = 0$ para $0 < f_k < C_k$
   (b) $f_k = 0$ para $d_k > 0$
   (c) $f_k = C_k$ para $d_k < 0$

Los problemas primal y dual son resueltos si se encuentran los flujos f y los potenciales Π en los nodos que satisfacen la factibilidad primal y las condiciones de holgura complementaria.

6.3 METODO DE FLUJO MAXIMO PARA OBTENER UNA SOLUCION PRIMAL -- FACTIBLE.

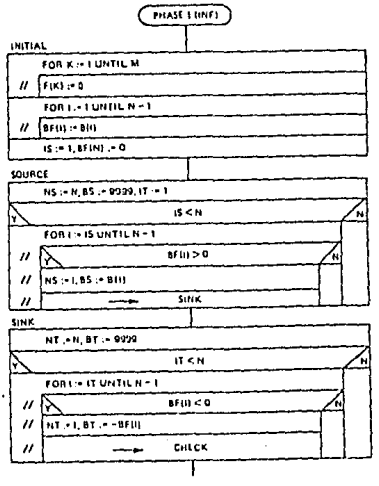
Aquí se obtiene una solución factible, que satisface los - flujos externos fijos en los nodos y no viola los límites de -- los flujos en los arcos. Los costos en los arcos no afectan al procedimiento y la solución obtenida no necesariamente es básica.

Se inicia con todos los flujos en cero, el procedimiento - elige arbitrariamente un par de nodos, uno con flujo externo positivo no satisfecho y otro con flujo externo negativo no satisfecho. Se usa el algoritmo de máximo flujo, para de los dos nodos elegidos, seleccionar el que tiene el menor valor absoluto y se trata de satisfacer el flujo externo. Pueden ocurrir dos cosas: el algoritmo de flujo máximo satisface el flujo externo fijo para el nodo elegido o se establece el flujo máximo entre el par de nodos. Cualquiera que sea el resultado se elige otro par de nodos con flujos externos fijos que aún no se satisfacen y se repite el proceso. Si solo existen nodos con flujos externos negativos no satisfechos o sólo existen nodos con flujos externos positivos no satisfechos, se elige el nodo anterior y -- uno de los nodos no satisfechos para la operación del máximo -- flujo. Si todos los flujos externos no pueden ser satisfechos, no hay solución factible del problema. Este método es implementado por el algoritmo PHASE1.

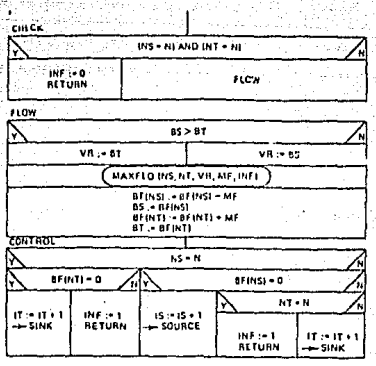
ALGORITMO PHASE1.

Propósito: Encuentra una red de flujo factible que satisface los flujos externos en los nodos.

1. (INICIAR) Hace todos los flujos - igual a cero. Inicialmente para el flujo no satisfecho b_{fi}, hace $b_{fi}=b_i$ y $b_{fn}=0$.
2. (FUENTE) Encuentra un nodo con -- flujo externo positivo no satisfecho. Sea este el nodo fuente s.



b_s es el fljjo no satisfecho para este nodo. Si no hay tales nodos, hace s=n (el nodo anterior) y -- b_s=R (donde R es un número grande).



3. (SUMIDERO) Encuentra un nodo con un flujo externo negativo no satisfecho, el cual no ha sido considerado previamente como un nodo s. Sea este el nodo sumidero t. b_t - es la magnitud del flujo no satisfecho en este nodo. Si no hay tales nodos, hace t=n y b_t=R.
4. (CHEQUEA) Si s y t son los nodos anteriores, termina; una solución factible ha sido encontrada. Si no, va al paso 5.
5. (FLUJO) Usa el algoritmo de flujo máximo para encontrar el - máximo flujo desde s a t limitado por el Min. ($|b_s|$, $|b_t|$). El algoritmo establece este flujo. Reduce la magnitud de los flujos externos no satisfechos para los nodos s y t, por la cantidad de flujo establecido entre los nodos mediante el algoritmo MAXFLO.
6. (CONTROL) Si s=n y b_{ft}=0, todos los flujos externos positi-- vos han sido satisfechos. Regresa al paso 3 y busca otro nodo sumidero con un flujo externo negativo no satisfecho. Si s=n y $b_{ft} \neq 0$, no hay flujo que pueda satisfacer al nodo t; - termina; el problema no es factible. $S \neq n$ y b_{fs}=0, el flujo externo para los nodosses satisfecho; va al paso 2 y encuentra otro nodo fuente. Si $s \neq n$, $b_{fs} \neq 0$ y t=n, no hay forma - de satisfacer el flujo externo del nodo s; termina; el pro-- blema es no factible. Si $s \neq n$, $b_{fs} \neq 0$ y $t \neq n$, va al paso 3 para buscar otro nodo sumidero.

La figura 6.2 ilustra el procedimiento aplicado al ejemplo de la figura 6.1, donde el algoritmo FPATH1 es usado para encontrar las trayectorias aumentadas. Todos los arcos que no aparecen tienen flujo cero. La figura 6.2a indica los flujos después de tres iteraciones con s=1 y t asumiendo los valores de 4, 5,

6 sucesivamente. Si el algoritmo siempre parte de un nodo fuente, significa que el flujo externo en este nodo ha sido satisfecho. Por esta razón al nodo 1 se le satisface su flujo externo en la tercera iteración. Como se puede notar en el nodo 1 se ofrecen cuatro unidades de flujo, que se distribuyen una unidad al nodo 4, dos unidades al nodo 5 y una unidad al nodo 6, por los arcos 1, 2 y 3, respectivamente, siendo 4, 5 y 6 los nodos sumidero. En la figura 6.2b se puede observar las tres siguientes iteraciones, en la que en los nodos sumidero 4 y 5 se satisface el flujo externo. Las figuras 6.2c y 6.2d indican las iteraciones siguientes. En la figura 6.2c se puede notar que del nodo fuente 3 salen tres unidades de flujo, una por el arco 9 al nodo 6, otra unidad que recorre por los arcos 7, -1, 3, y llega al nodo 6 y otra unidad que va por los arcos 7, -4, 6 y llega al nodo 6. Finalmente en la figura 6.2d se puede observar que del nodo 7 sale una unidad de flujo que recorre los arcos 11, -8, -2, 3 y llega al nodo 6, con lo cual queda satisfecho el flujo externo en el nodo 6. En esta forma se obtiene una solución inicial factible.

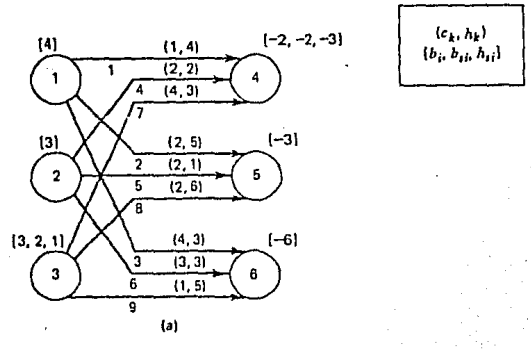


Figura 6.1

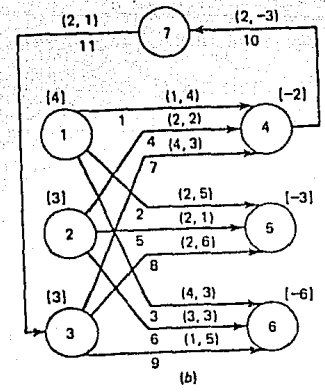


Figura 6.1 (continuación)
(a) Red original. (b) Red con el nodo de holgura.

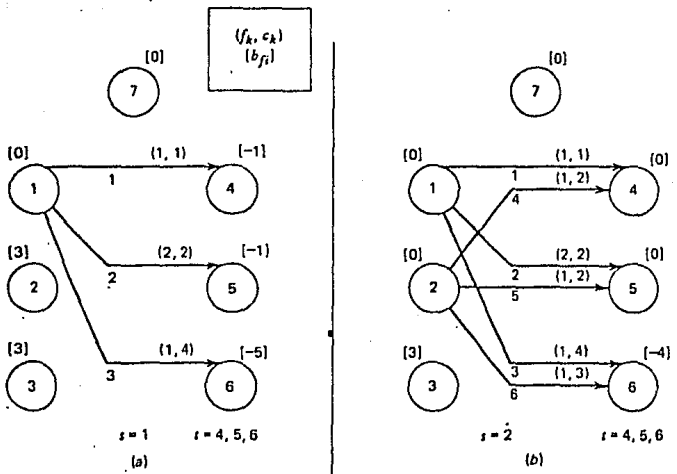


Figura 6.2

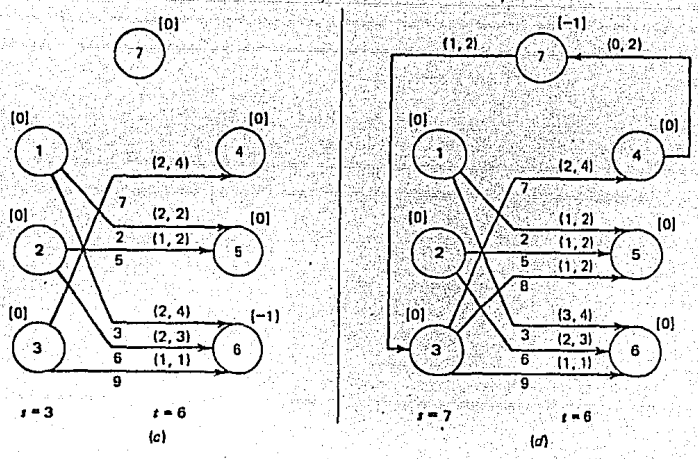


Figura 6.2 (continuación)
Se encuentra una solución factible usando el algoritmo de máximo flujo.

En la solución de este problema, inicialmente se emplea el método de flujo máximo, para obtener una solución primal factible y ésta sirve después para encontrar la solución óptima, mediante el uso de un algoritmo primal no básico PRIMAL1, que se estudiará más adelante en la sección 6.5.

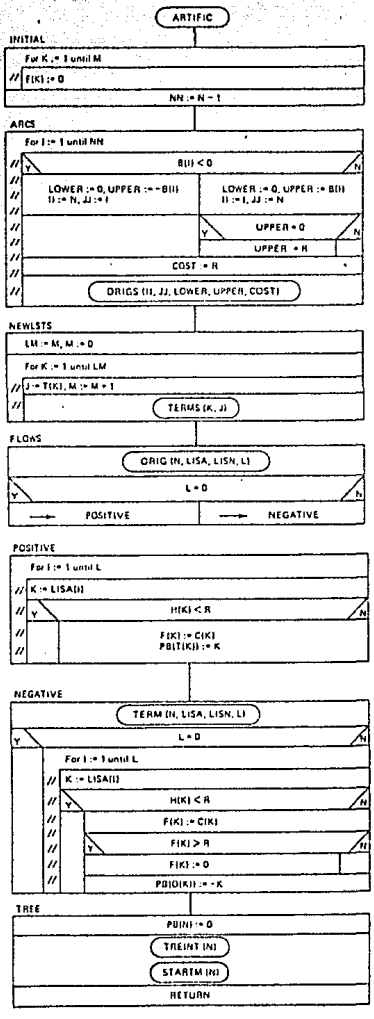
6.4 METODO DEL ARCO ARTIFICIAL PARA OBTENER UNA SOLUCION PRIMAL FACTIBLE.

Este método se emplea para obtener una solución factible - que también es básica, el cual suministra arcos artificiales para satisfacer las restricciones de factibilidad de los nodos, - poniendo un costo muy grande en los arcos. Si existe una solución factible el flujo en estos arcos se hacer cero. Una alternativa es poner el costo de una unidad en estos arcos y dar a los otros arcos un costo cero. Minimizando el costo en la red se puede obtener una solución con costo cero si una solución -- factible existe. El costo mínimo diferente de cero indica que la red no tiene solución factible.

Este método se implementa por medio del algoritmo ARTIFIC..

El algoritmo conecta cada nodo, mediante un arco artificial al nodo de holgura. Si el flujo externo fijo del nodo es mayor o igual a cero, el arco artificial termina en el nodo de holgura, de lo contrario el arco artificial se origina en el nodo de holgura y termina en el nodo original.

ALGORITMO ARTIFIC.
Propósito: Forma una base inicial -- con todos los arcos artificiales para una red de flujo a costo mínimo.



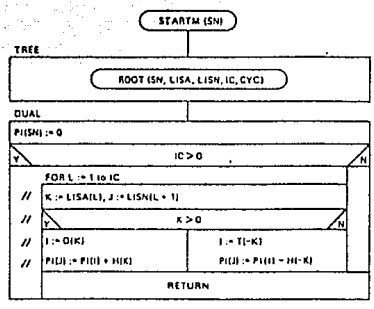
1. (INICIAR) Hace todos los flujos - iguales a cero y calcula el número de nodos (excluyendo el nodo - de holgura).
2. (ARCOS) Determina la dirección y los límites superiores de los arcos artificiales. Los límites inferiores los hace igual a cero, - los costos en los arcos los hace igual a un valor grande, y formúla la nueva lista P_o.
3. (NUEVA LISTA) Formúla las nuevas listas L_T y P_T.
4. (FLUJO) Obtiene el conjunto de arcos que se originan en el nodo de holgura. Si no hay tales arcos, va al paso 6. De otra manera va al paso 5.
5. (POSITIVO) Para cada arco en el - conjunto originado en el nodo de nolgura, determina si es un arco artificial. Si no va al próximo arco. Para cada arco artificial k, hace $f_k=C_k$, y determina que el arco k es el apuntador hacia atrás del nodo t_k.

6. (NEGATIVO) Obtiene el conjunto de arcos que terminan en el nodo de holgura. Si no hay tales arcos va al paso 7. De otra manera, para cada uno de estos arcos del conjunto, determina si es un arco artificial. Si no va al próximo arco. Para cada arco artificial, hace $f_k=C_k$. Si resulta $f_k \geqslant R$, hace $f_k=0$. (Esto implica $b(0_k)=0$) . Determina que el arco -k es el apuntador hacia atrás del nodo 0_k.
7. (ARBOL) Forma el árbol básico inicial y calcula los potenciales en los nodos.

El algoritmo STARTM calcúla los potenciales en los nodos en el árbol básico inicial.

ALGORITMO STARTM.

Propósito: Dado un árbol básico, este algoritmo establece los potenciales en los nodos, tal que $\Pi_i+h_k := \Pi_j$ para cada arco básico k(i,j). Se permiten arcos hacia adelante y reflejados.



1. (ARBOL) Lista los nodos y los arcos del árbol tal que cada nodo (arco) aparecen en la lista después de todos sus nodos (arcos) predecesores. La lista de nodos (LISN) y la lista de arcos (LISA) son construídos tal que el nodo terminal a la i-écima entrada en la lista de arcos, es la (i+1)-écima entrada en la lista de nodos. Hace $\ell=1$ y $\Pi_s=0$.
2. (DUAL) Sea k la ℓ-écima entrada en la lista de arcos. j es la (ℓ+1)-écima entrada en la lista de nodos. Encuentra el origen del arco k. Hace $\Pi_j := \Pi_i+h_k$. Si encuentra el final de la lista de arcos, termina. De lo contrario, incrementa ℓ en 1 y repite el paso 2.

Recordando que el costo del arco reflejado es el costo ne-

gativo de su respectivo arco hacia adelante, es fácil calcular el propio valor de Π_j si el nodo j es un arco reflejado por su apuntador hacia atrás en el árbol. Supóngase que $\Pi_5=3$, $h_{11}=5$ y $p_B(7)=-11$. Se desea en general, que STARTM determine los valores de los potenciales en los nodos tal que $\Pi_i+h_k=\Pi_j$. En este caso específico I=5, k=-11 y j=7. Por tanto se requiere que PI(5) - H(11) = PI(7) y ésta es la expresión usada en el algoritmo STARTM.

La figura 6.3 muestra los resultados de aplicar el algoritmo ARTIFIC para la red dada en la figura 6.1. La figura 6.3b - presenta la base artificial del árbol.

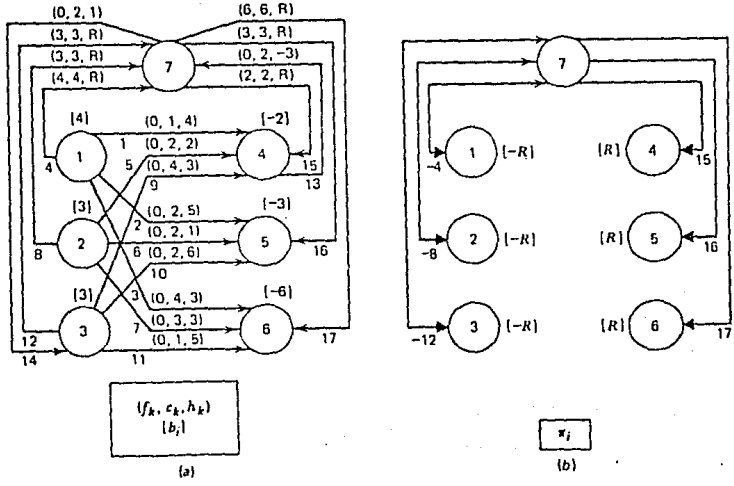


Figura 6.3
Resultados de aplicar el algoritmo ARTIFIC a la red de la figura 6.1

Luego esta base artificial primal básica factible es usada en el algoritmo PRIMAL BASICO, para obtener la solución óptima, como se verá más adelante en la sección 6.6.

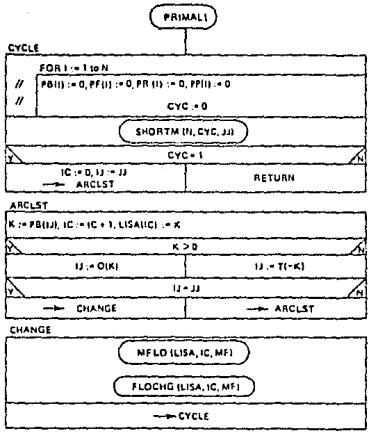
6.5 SOLUCION DEL PROBLEMA MEDIANTE EL USO DE UN ALGORITMO PRIMAL NO BASICO.

Este algoritmo requiere de una solución factible para los

flujos. Si no hay ciclos negativos en la red marginal, los flujos son óptimos. Si hay un ciclo negativo, se puede encontrar un conjunto de flujos que aún son factibles pero que tengan un costo total menor. Este nuevo conjunto de flujos es fácil de encontrar al incrementar los flujos en el ciclo negativo hasta que un arco en el ciclo se sature. El costo total de los flujos decrece en una cantidad igual al producto del incremento del flujo por el número de veces que gira alrededor del ciclo. El proceso continúa hasta que el ciclo se haga no negativo. El algoritmo se llama primal debido a que mantiene soluciones primales factibles durante el proceso, pero no son necesariamente básicas. Este procedimiento es implementado por el algoritmo PRIMAL1.

ALGORITMO PRIMAL1.

Propósito: Inicia con una solución primal factible y encuentra la solución óptima descubriendo y saturando iterativamente, ciclos con costos negativos en la red marginal.



1. (CICLO) Encuentra un ciclo con costo negativo en la red marginal. Si no existe ninguno, termina con la solución óptima. Si encuentra uno, va al paso 2.
2. (LISTA DE ARCOS) Encuentra la lista de arcos en el ciclo negativo usando los apuntados hacia atrás de los nodos.
3. (CAMBIO) Determina el cambio máximo del flujo en el ciclo, el cual debe mantener la factibilidad en los arcos. Cambia los flujos en los arcos del ciclo por esta cantidad. Va al paso 1.

Para ilustrar el método, considere la red de la figura 6.4a y su red marginal asociada en la figura 6.4b. Note que los flujos que constituyen una solución factible tienen un costo total de 45 unidades. No obstante, la red marginal no está libre de

ciclos negativos. Por ejemplo tenemos un ciclo negativo formado por los arcos -10, -9, 8 y el valor del ciclo es $h_{-10} + h_{-9} + h_8 = -2-3 + 1 = -4$.

Además podemos ver que al enviar una unidad de flujo desde el nodo 4 al nodo 7 por medio de los arcos 9 y 10, se incurre en un costo de 5 unidades. Enviando la misma unidad de flujo desde el nodo 4 al 7, por medio del arco 8 se incurre solamente en un costo de 1 unidad. Luego se economiza un costo de 4 unidades, el que está dado al enviar una unidad de flujo por este ciclo, que a su vez satura el arco 8, en el cual su flujo llega a cuatro unidades que es igual a su capacidad. Los resultados se observan en la figura 6.4c, cuyo costo total se reduce a 41 unidades.

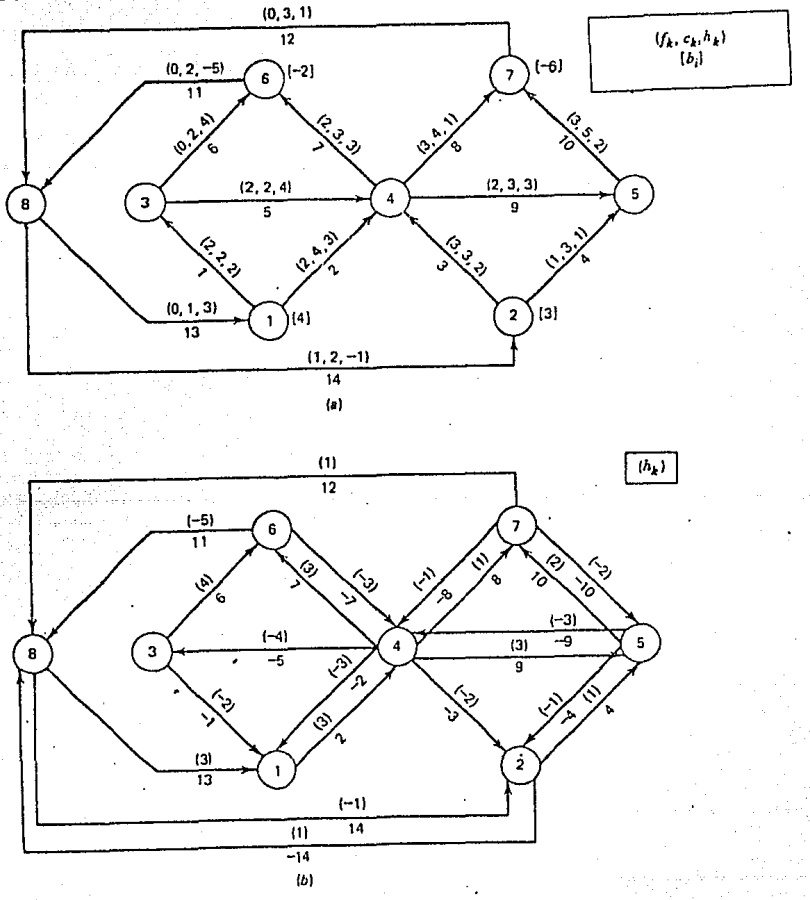
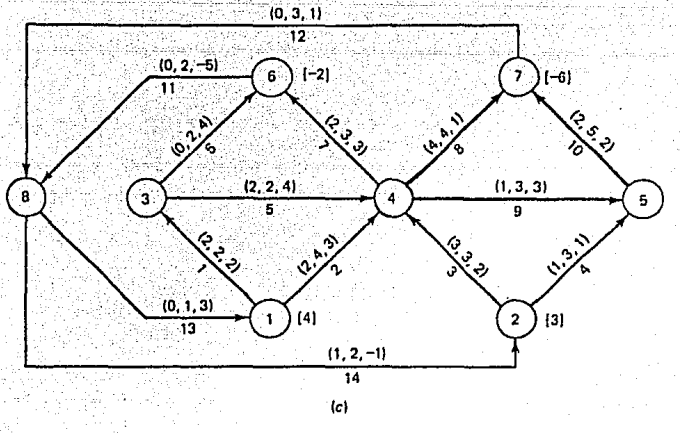


Figura 6.4

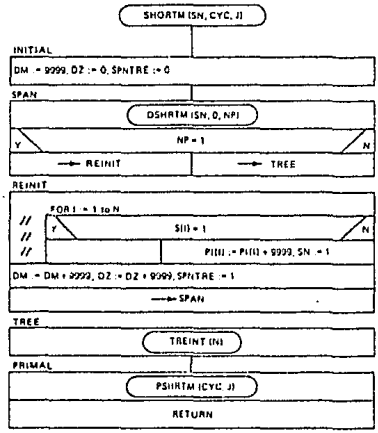


(c)

Figura 6.4 (continuación)
(a) Red original. (b) Red marginal. (c) Flujos resultantes.

El cico negativo es encontrado mediante el algoritmo SHORTM, que considera como arcos admisibles todos los arcos de la red - marginal (arcos hacia adelante y arcos reflejados), el cual además trabaja con los algoritmos DSHORTMy PSHORTM, éste último a su vez con el algoritmo SMNSPM.

ALGORITMO SHORTM.
Propósito: Encuentra el árbol de trayectoria más corta en una red que puede tener arcos con costos negativos. Considera los arcos hacia adelante y reflejados. El algoritmo termina -- cuando se encuentra el árbol de trayectoria más corta o cuando se detecta un ciclo.



1. (INICIAR) Inicializa DM, DZ y -- SPNTRE.
2. (EXPANDE) Usa el algoritmo Dijkstra para encontrar una solución básica factible para el árbol de trayectorias más cor-- tas. Si no existe un árbol expandido, va al paso 3. De otra manera va al paso 4.

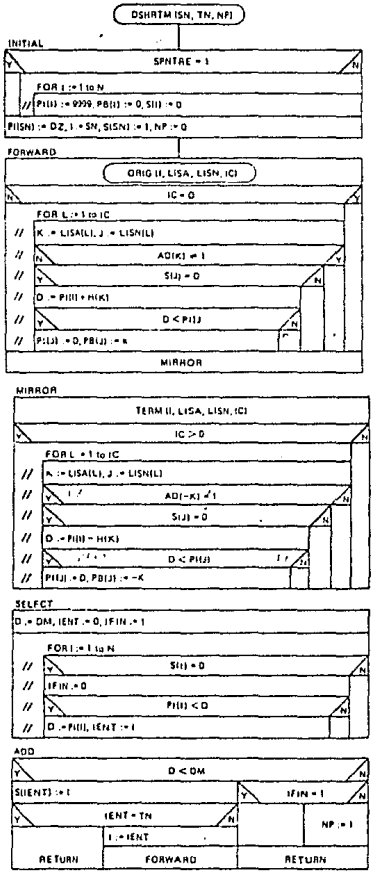
3. (REINICIA) Añade un número grande a los Π_i de los nodos que no hayan sido aún etiquetados como parte del árbol. Actualiza los valores de DM y DZ y pone una bandera indicando que ningún árbol expandido se ha obtenido aún. Va al paso 2.
4. (ARBOL) Crea una representación del bosque expandido o del árbol expandido mediante apuntadores, desde DSHORT.
5. (PRIMAL) Usa el algoritmo primal para probar la optimalidad de la solución inicial y hacer los cambios necesarios para obtener el árbol de trayectorias más cortas. Si se descubre algún ciclo en el proceso, lo indica con CYC igual a 1 y J algún nodo en el ciclo.

ALGORITMO DSHORTM

Propósito: Encuentra la trayectoria más corta desde el nodo s al nodo t, cuando todos los arcos admisibles tienen costos positivos (se permiten arcos hacia adelante y arcos reflejados). Si t=0, el algoritmo determina el árbol de trayectoria más corta.

1. (INICIAR) Sea Π_i=DM (DM un número grande), para todos los nodos excepto para s. Hace todos los apuntadores hacia atrás iguales a cero. Define el conjunto de nodos S= {s} . Hace i:=s y Π_s:=0.

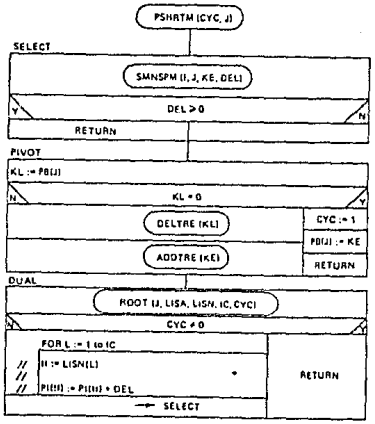
2. (ADELANTE) Para cada arco hacia adelante originado en el nodo i, k(i,j), tal que j ∈ S, calcula la longitud de la trayectoria al nodo j, a través del nodo i(Π_i+h_k). Si este valor es menor que Π_j, reemplaza Π_j por Π_i+h_k. Reemplaza el apuntador hacia atrás del nodo j con k.



3. (REFLEJADO) Para cada arco que termina en el nodo i, k(i,j), tal que $j \notin S$, determina si el arco reflejado $-k(i,j)$ es admisible. Si este es el caso, calcula la longitud al nodo j y a través del arco reflejado ($\Pi_i - h_k$). Si este valor es menor que Π_j, reemplaza Π_j por $\Pi_i - h_k$. Reemplaza el apuntador hacia atrás del nodo j con $-k$.
4. (SELECCIONA) Encuentra $D=\text{Min}\{\Pi_j : j \in N - S\}$. i_E es el nodo para el cual se obtiene el mínimo. Se hace IFIN igual a cero si algún nodo no pertenece o no está en el conjunto S.
5. (ADICIONA) Si $D < DM$, incluye i_E en S. Si $i_E = t$, termina con la trayectoria más corta desde s a t. Si $i_E \neq t$, hace $i = i_E$ y va al paso 2. Si D=DM y todos los nodos están en el conjunto S, termina con el árbol de trayectoria más corta. Si todos los nodos no están en S. Hace NP=1, para indicar que no hay trayectoria más corta desde s a t.

ALGORITMO PSHRTM.

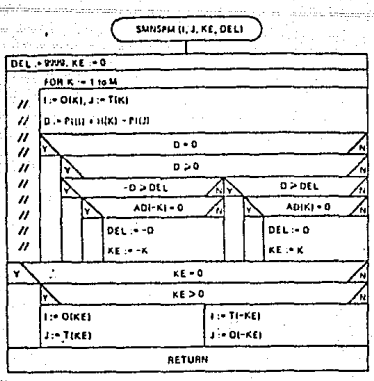
Propósito: Empezando con un árbol básico factible y con potenciales en los nodos que violan las condiciones de factibilidad del dual para algunos arcos no básicos, este algoritmo modifica iterativamente el árbol básico y los potenciales en los nodos para obtener una solución óptima.



1. (SELECCIONA) Encuentra un arco no básico $k_E(i,j)$ para el cual $\Pi_i + h_{k_E} < \Pi_j$. Si no existe tal arco, termina con la solución óptima. De esta manera, hace $d_E = \Pi_i + h_{k_E} - \Pi_j$.
2. (PIVOTE) Sea $k_L(i',j)$ el arco básico que termina en el nodo j. Borra este arco del árbol básico y añade k_E al árbol básico.
3. (DUAL) Encuentra los nodos del árbol con raiz en el nodo j. Si encuentra un ciclo negativo en este paso, termina. No hay solución al problema. De otra manera suma d_E a los potenciales de todos los nodos en el conjunto.

ALGORITMO SMNSPM.

Propósito: Encuentra el arco admisible que tiene el valor más negativo de $\Pi_i + h_k - \Pi_j$.



1. Hace D:=9999 y k_E:=0.
2. Para cada arco k(i,j), calcula -- $d_k = \Pi_i + h_k - \Pi_j$. Si d_k=0, va al siguiente arco. Si $d_k < 0$, el arco k es admisible, y $d_k < D$, hace -- k_E:=k y D:=d_k. Si $d_k > 0$, el arco -k es admisible, y $-d_k < D$, hace k_E:=k y D:=$-d_k$. Va al siguiente arco.
3. Si $k_E > 0$, hace i:=o(k_E) y j:=t(k_E). Si $k_E < 0$, hace i:=t($-k_E$) y j:=o($-k_E$).

La aplicación de DSHRTM a una red marginal no necesariamente produce un árbol expandido. Afortunadamente, un bosque expandido de arcos admisibles sirve como un árbol expandido en la implementación de PSHRTM. Dado el bosque expandido se puede aplicar el algoritmo PRIMAL independientemente a cada subred de la red original.

El uso de este algoritmo se ilustra en el ejemplo de la figura 6.5.

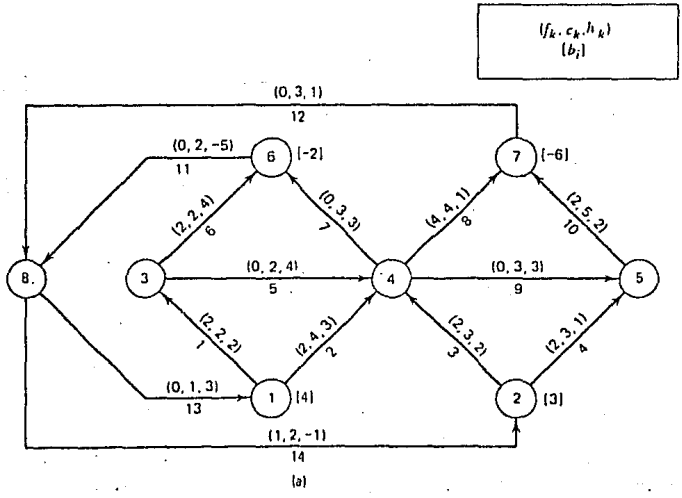


Figura 6.5.

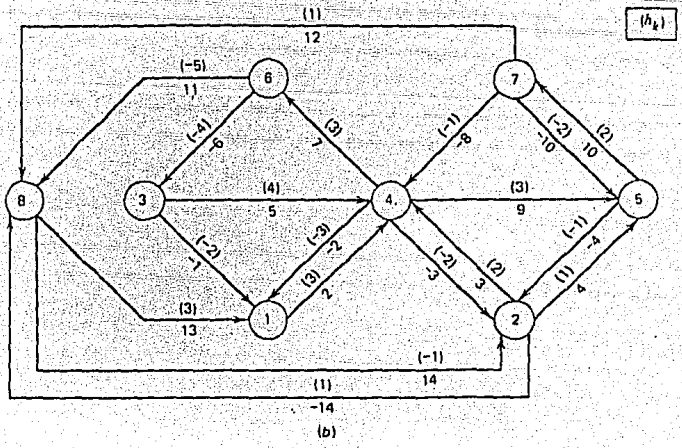


Figura 6.5 (continuación)
(a) Red inicial. (b) Red marginal.

En la figura 6.5a se presenta la configuración de los flujos obtenidos mediante el algoritmo PHASE1 (método de flujo máximo), después de ser aplicado a la red de la figura 6.4. En la figura 6.5b se presenta la red marginal de la red de la figura 6.5a.

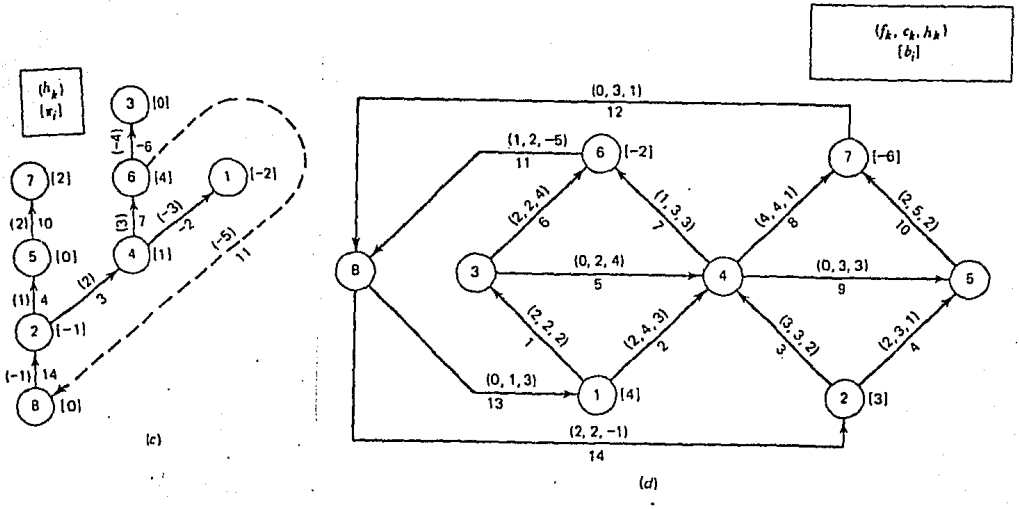


Figura 6.5.

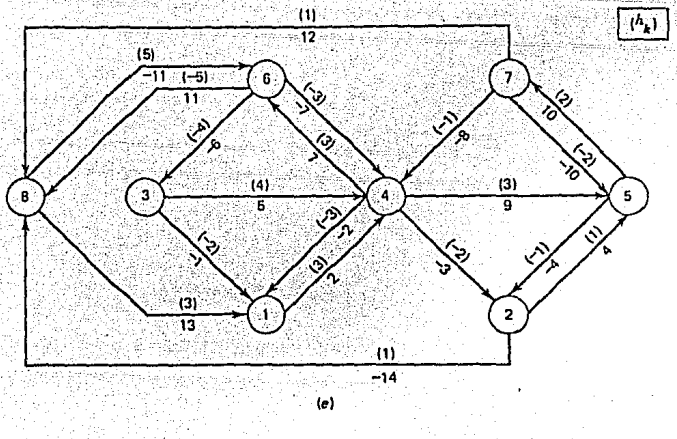


Figura 6.5
(c) Arbol determinado por DSHRTM. (d) Nueva red. (e) Red marginal.

Aplicando el algoritmo DSHRTM a la red de la figura 6.5b, se tiene el árbol expandido dado en la figura 6.5c. El arco en la línea punteada 11(6.8) es seleccionado por el algoritmo -- SMNSPM, como el arco admisible con el valor más negativo de d_k.

Se puede ver que al entrar el arco 11(6.8), se forma un ciclo negativo con el valor

$$d_{11} = \Pi_6 + h_{11} - \Pi_8 = 4 + (-5) - 0 = -1$$

Esto se puede verificar sumando los costos de los arcos en el ciclo, que es $d_{11}=h_{14}+h_3+h_7+h_{11}=-1$. Para obtener un nuevo flujo factible con menor costo que el dado en la figura 6.5a se necesita solo aumentar el flujo a través del ciclo. La cantidad de flujo aumentado a través del ciclo está limitado a 1, el cual satura los arcos 3 y 14.

Los resultados del nuevo flujo y la red marginal son dados en las figuras 6.5d y 6.5e, respectivamente.

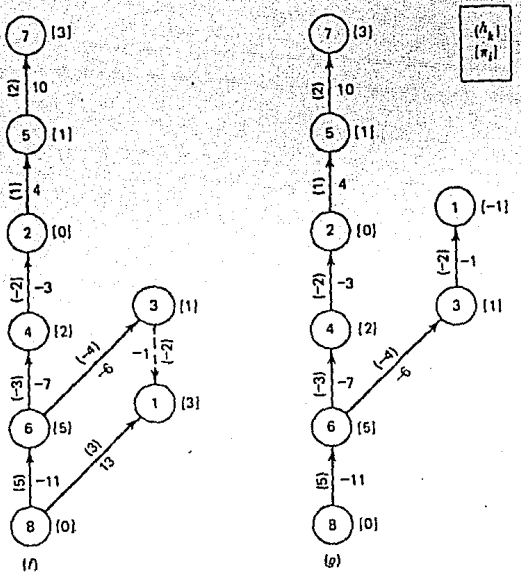
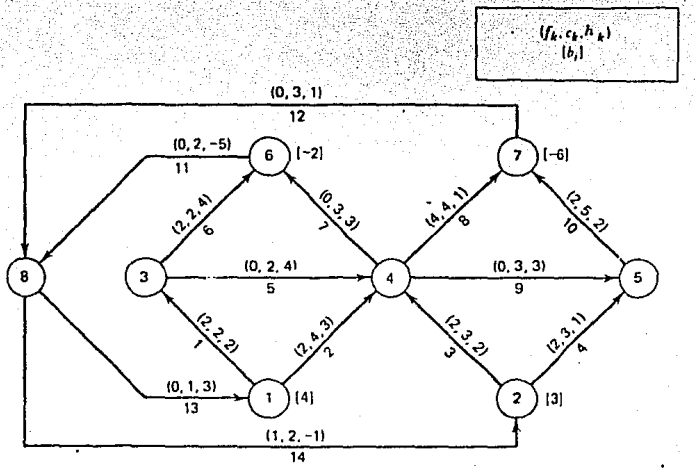


Figura 6.5
(f) Arbol determinado por DSHRTM. (g) Arbol determinado por TRECHG.

La figura 6.5f presenta el árbol expandido obtenido por -- DSHRTM, desde la figura 6.5e. El arco -1(3, 1) es luego seleccionado por SMNSPM para que entre a la base, pero no forma un ciclo negativo. Mediante TRECHG que saca el arco k_L=13 y añade el arco k_E=-1 se tiene el árbol expandido de la figura 6.5g, el cual es una solución óptima para el problema del árbol de tra-- yectoria más corta asociado con la figura 6.5e.

Puesto que no fue encontrado un ciclo negativo adicional, la configuración de los flujos de la figura 6.5d son una solu-- ción óptima para este ejemplo de flujo a costo mínimo.

En el anexo A-2 se presenta la subrutina principal FCMI65 codificada en FORTRAN, la misma que utiliza el método de flujo máximo (algoritmo primal factible PHASE1) para obtener una solución primal factible y después el método primal no básico (PRIMAL1) para obtener la solución óptima; de la red de la figura 6.5a, cuyos datos de entrada y resultados se pueden observar en las figuras 6.6 y 6.7, respectivamente.




```
100    65
200     7
300     1          4          1          3
400     2          3          2         -1
500     3          0          0          0
600     4          0          0          0
700     5          0          0          0
800     6         -2         -2         -5
900     7         -6         -3          1
1000
1100     1    3          0          2          2
1200     1    4          0          4          3
1300     2    4          0          3          2
1400     2    5          0          3          1
1500     3    4          0          2          4
1600     3    6          0          2          4
1700     4    6          0          3          3
1800     4    7          0          4          1
1900     4    5          0          3          3
2000     5    7          0          5          2
2100
2200
$
```

Fig. 6.6

NRO. CLAVE DE ESTA SUBRUTINA ES : 65

PROGRAMA FCMI65
PRIMAL NO BASICO PARA FLUJO A COSTO MINIMO

REPRESENTACION DE LA RED--

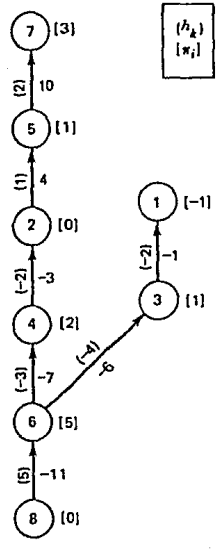
NRO. DE NODOS-- 8 NRO. DE ARCOS-- 14

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	4
2	3
3	0
4	0
5	0
6	-2
7	-6
8	0

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	O(K)	T(K)	C(K)	H(K)
1	1	3	2	2
2	1	4	4	3
3	2	4	3	2
4	2	5	3	1
5	3	4	2	4
6	3	6	2	4
7	4	6	3	3
8	4	7	4	1
9	4	5	3	3
10	5	7	5	2
11	6	8	2	-5
12	7	8	3	1
13	8	1	1	3
14	8	2	2	-1



SOLUCION OPTIMA DEL PROBLEMA

NODO I	P(I)	PB(I)
1	-1	-1
2	0	-3
3	1	-6
4	2	-7
5	1	4
6	5	-11
7	3	10
8	0	0

ARC K	O(K)	T(K)	FLUJO	CAPACIDAD	COST/UNI.	COST.FLUJO
1	1	3	2	2	2	4
2	1	4	2	4	3	6
3	2	4	3	3	2	6
4	2	5	2	3	1	2
5	3	4	0	2	4	0
6	3	6	2	2	4	8
7	4	6	1	3	3	3
8	4	7	4	4	1	4
9	4	5	0	3	3	0
10	5	7	2	5	2	4
11	6	8	1	2	-5	-5
12	7	8	0	3	1	0
13	8	1	0	1	3	0
14	8	2	2	2	-1	-2

COSTO TOTAL = 30

FIN DEL PROGRAMA RFCLSG

EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES: 0.283333

#ET=2:15.3 PT=0.4 IO=0.2

Fig. 6.7

6.6 SOLUCION DEL PROBLEMA MEDIANTE EL USO DE UN ALGORITMO PRIMAL BASICO.

Este algoritmo usa un árbol básico para implementar un procedimiento análogo al algoritmo primal simplex en el problema general de programación lineal. Este procedimiento inicia con un flujo primal básico factible, definido en un árbol expandido -- $D_T = [N, M_T]$ de arcos hacia adelante y reflejados. Los potenciales en los nodos son calculados por:

$$\Pi_i + h_k = \Pi_j \qquad \text{para el arco } k(i,j) \in M_T \qquad (1)$$

El flujo en el arco no básico k(i,j) es igual a cero o c_k. Las condiciones de holgura complementaria para los arcos no básicos son:

$$\text{Si } f_k = 0 , \qquad \Pi_i + h_k \geqslant \Pi_j \qquad (2)$$

y

$$\text{Si } f_k = C_k , \qquad \Pi_i + h_k \leqslant \Pi_j \qquad (3)$$

Estas condiciones no se satisfacen, sino hasta la terminación del algoritmo.

El algoritmo primal básico tiene cuatro pasos principales:

1. Selecciona un arco que viole una de las condiciones 2 o 3 y lo llama arco entrante $k_E(i_E, j_E)$.
   Este arco entra a la base. Cuando un arco no está en la base, su flujo es cero o su flujo es C_k.
   Si su flujo $f_k(i,j)=0$, el arco hacia adelante k(i,j) es admisible y se define:

$$d_k = \Pi_i + h_k - \Pi_j .$$

   Si su flujo $f_k(i,j)=C_k$, el correspondiente arco reflejado -k es admisible y se define:

$$d_{-k} = \Pi_j - h_k - \Pi_i$$

Los potenciales en los nodos están definidos tal que -- $d_k = 0$ para los arcos básicos; $d_k \geq 0$ para los arcos no básicos que satisfacen las condiciones de optimalidad, y $d_k < 0$ si no lo satisfacen.

Para elegir el arco entrante se aplica la regla "el primer nodo más negativo" mediante el algoritmo SELECT, en el cual, en cada nodo se va revisando los arcos hacia - adelante y los arcos reflejados que en estos nodos se originan. Cuando se encuentra el primer nodo con arcos $d_k < 0$, el arco con el d_k más negativo que se origina en ese nodo es elegido para entrar a la base y se lo define como el arco $k_E(i_E, j_E)$.

2. Determina el ciclo que se forma al incorporar el arco entrante $K_E(i_E, j_E)$, y encuentra el máximo flujo que se -- puede cambiar en el ciclo, para determinar si el arco - entrante o algún otro arco básico sale de la base. Este arco se lo denomina arco saliente $k_L(i_L, j_L)$.

   Sea $D_c = [N_c, M_c]$ el ciclo formado al añadir el arco -- k_E en la base. Si tomamos la dirección del ciclo en el mismo sentido de $k_E(i_E, j_E)$, tenemos que el ciclo se forma por los siguientes conjuntos de arcos:

$$M_c = M_F \cup M_R \cup k_E$$

   Donde M_F y M_R son los arcos en el mismo sentido y en el sentido inverso de la dirección del ciclo, respectivamente.

   El cambio máximo de flujo en el ciclo es:

$$\mu_k = \begin{cases} C_k - f_k & \text{si} \quad k \in M_c \quad y \quad k > 0 \\ f_{-k} & \text{si} \quad k \in M_c \quad y \quad k < 0 \end{cases}$$

para todo $k \in M_c$

Luego el máximo cambio de flujo para el ciclo D_c es:

$$\mu_{k_L} = \text{Min.} \{ \mu_k \mid \mu_k \in M_c \} \quad , \quad k_L \quad (u, v)$$

El cambio de flujo en la red original G, está dado por:

$$f'_k = f_k + \mu_{k_L} \quad \text{para} \quad k \in M_c \quad y \quad k > 0$$

$$f'_k = f_{-k} - \mu_{k_L} \quad \text{para} \quad k \in M_c \quad y \quad k < 0$$

$$f'_k = f_k \quad \text{para} \quad k \notin M_c$$

3. Modifica la base del árbol tal que k_E entra y k_L sale. Si k_E y k_L son los mismos arcos, la base es la misma, - salvo la modificación de los flujos.

   Si $k_L \neq k_E$ se deben considerar dos posibilidades:

   a) Si k_L está en M_R, la nueva base se forma quitando k_L, luego añadiendo k_E y finalmente invirtiendo los arcos de j a v, dandole la misma dirección de k_E; como se puede observar en la figura 6.9a.

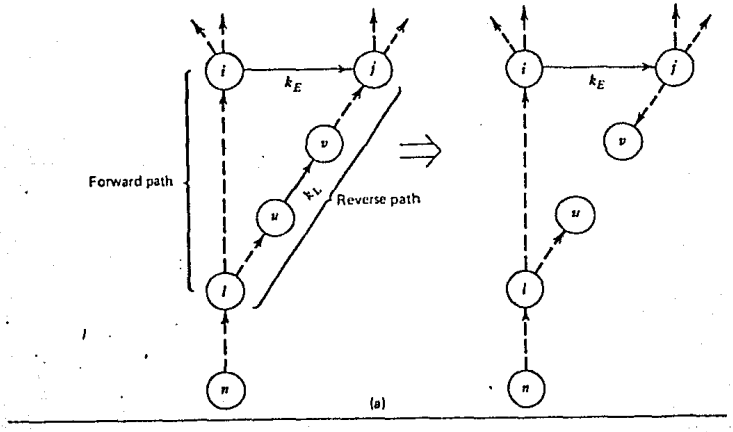


Figura 6.9

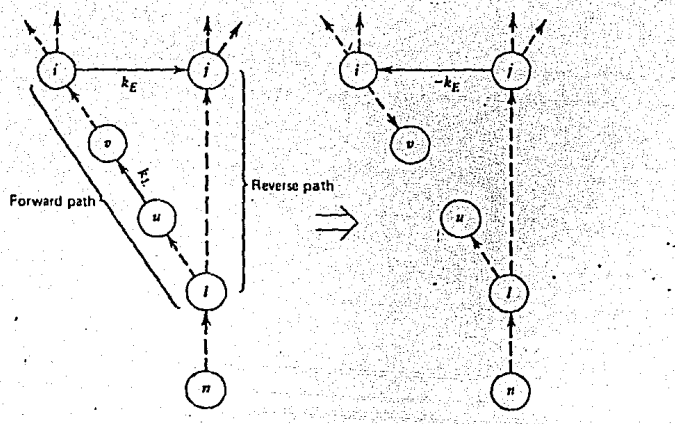


Figura 6.9 (continuación)
Cambios en el árbol básico.

b) Si $k_L \in M_F$ la nueva base se forma quitando k_L, luego añadiendo $-k_E$, e invirtiendo los arcos i a v, como se puede observar en la figura 6.9b.

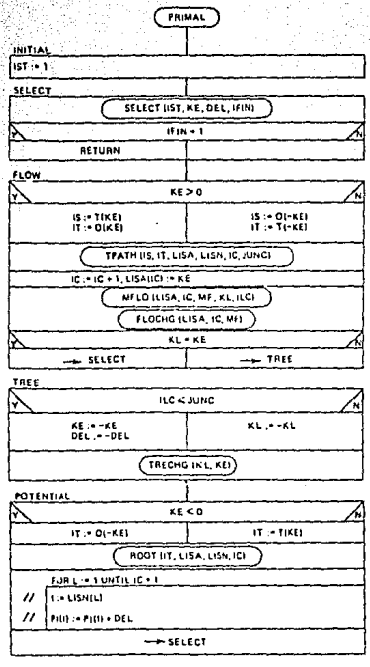
4. Modifica los potenciales en los nodos tal que la condición (1) sea satisfecha en el nuevo árbol y va al paso 1. Se modifican solamente los potenciales en el subárbol enraizado en el nodo terminal de k_E según su posición en el nuevo árbol.

   Ya que $d_k < 0$ para las dos posibilidades mencionadas en el paso 3. Según la figura 6.9, tenemos:

   a) Si $k_E \in M_R$, entonces $\Pi'_i = \Pi_i + d_{k_E}$, para $i \in$ {Subárbol enraizado en el nodo j_E} .
   Que comprende un decremento en los potenciales.
   b) Si $k_E \in M_F$, entonces $\Pi'_i = \Pi_i - d_k$, para $i \in$ {Subárbol enraizado en el nodo i_E} .
   Que corresponde a un incremento en los potenciales.

Estos cuatro pasos se repiten hasta que las condiciones de optimalidad son satisfechas para todos los arcos. Los detalles de este procedimiento se describen en el algoritmo PRIMAL.

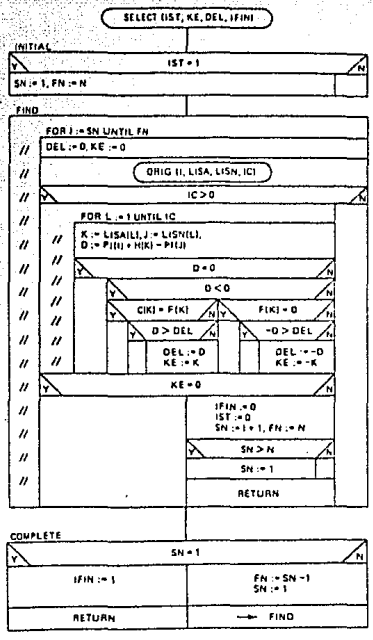
ALGORITMO PRIMAL.
Propósito: Ejecuta la técnica primal simplex para los problemas de flujo a costo mínimo. El algoritmo inicia con una solución básica factible.



1. (SELECCIONAR) Selecciona un arco $k_E(i,j)$ que viola las condiciones de optimalidad, para que entre a la base. Si no lo hay, termina con la solución óptima. Hace $\Delta = \Pi_i + h_{k_E} - \Pi_j$ si $k_E > 0$ o $-\Pi_i - h_{-k_E} + \Pi_j$ si $k_E < 0$.
2. (FLUJO) Encuentra el ciclo, formado por el arco entrante y los arcos básicos. Determina la cantidad máxima de flujo que se puede incrementar en estos arcos, manteniendo la factibilidad de los mismos. Cambia el flujo en los arcos por esta cantidad. Selecciona el arco, el cual se vuelve inadmisible con este cambio de flujo para que salga de la base; este arco es $k_L(u, v)$. Si el arco entrante es el mismo que el arco saliente, va al paso 1.
3. (ARBOL) Si el arco k_L está en la parte hacia adelante en la trayectoria desde i a j, hace $k_E := -k_E$ y $\Delta := -\Delta$. Si k_L está en la parte reversa de la trayectoria desde el nodo i a j, hace $k_L := -k_L$.
4. (POTENCIAL) Encuentra el nodo terminal de k_E, r'. Determina el conjunto de nodos en el árbol enraizado en r'. Adiciona Δ a todos los nodos en este conjunto. Va al paso 1.

ALGORITMO SELECT.

Propósito: Encuentra un arco que no satisface las condiciones de optimalidad, usando la regla "el primer nodo más negativo".



1. (INICIAR) Si éste es el primer paso en la subrutina, fija la búsqueda para los nodos de 1 a n.
2. (ENCUENTRA) Busca a través de la lista de nodos. Para cada nodo i, en la búsqueda, encuentra el conjunto de arcos que se originan en este nodo. Chequea las condiciones de optimalidad para cada arco. Si encuentra un arco que viola las condiciones, continúa el chequeo de los arcos en este nodo. Elige el arco para el cual las condiciones son violadas en mayor proporción. Establece la búsqueda del nodo, para el próximo paso, a través de este algoritmo desde i + 1 a n y retorna. Si no se encuentran arcos óptimos, va al paso 3.
3. (COMPLETA) Si en la búsqueda en el paso 2, yendo desde el nodo 1 no hay arcos óptimos, la solución es óptima. Si no, fija la búsqueda desde el nodo 1 a uno menos que lo recorrido en el paso 2 y retorna al paso 2.

En la figura 6.7 se ilustra este procedimiento primal básico. La figura 6.7a presenta una solución inicial básica factible y el árbol básico asociado. Los arcos admisibles no básicos se muestran en líneas punteadas con sus respectivos valores asociados d_k, luego la primera solución no es óptima. Se elige el arco con el valor más negativo, 5(2,5), para entrar a la base. Este arco conjuntamente con los arcos básicos, forman el ciclo $M_c = \{ 5, -2, 3, -6 \}$. El máximo cambio de flujo en este ciclo es 2, luego del cual los arcos 5 y 3 alcanzan su capacidad y el arco 2 llega su flujo a cero. Este cambio de los flu-

jos se puede observar en la figura 6.10b.

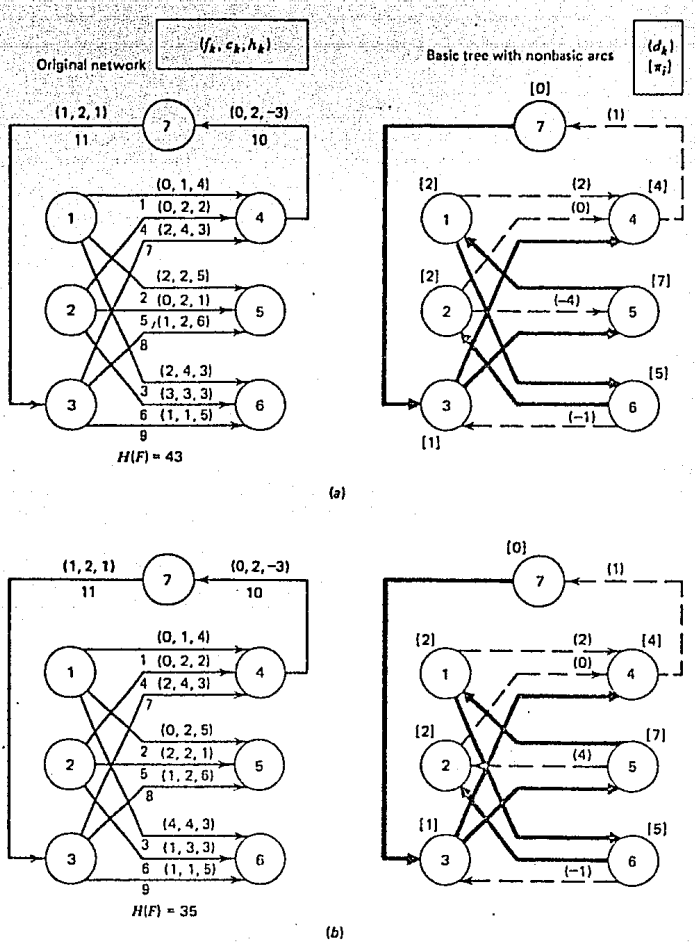


Figura 6.10
Aplicación del algoritmo primal.

De los posibles arcos que pueden salir del árbol, se elige arbitrariamente el arco 5, pero este arco fue el mismo que entró, luego el árbol básico no cambia en la figura 6.10b, los potenciales en los nodos permanecen iguales, no obstante los flujos cambian.

En la figura 6.10b se observa que sólo el arco -9(6,3) tie

ne un d_k negativo. Luego este arco entra a la base, formandose el ciclo M_c = {-9, 8, -2, 3} . Esta es una iteración degenerada, ya que, no hay posibilidad de cambio de flujo en el ciclo; el arco 3 está en su máxima capacidad y el arco 2 tiene flujo - cero. Se elige arbitrariamente al arco -2 para que salga de la base, el cual está en la parte hacia adelante del ciclo, luego el arco reflejado -9 entra a la base. Así el arco 9(3, 6) entra a la base y el arco -2(5, 1) sale, formandose el nuevo árbol, como se puede observar en la figura 6.10c.

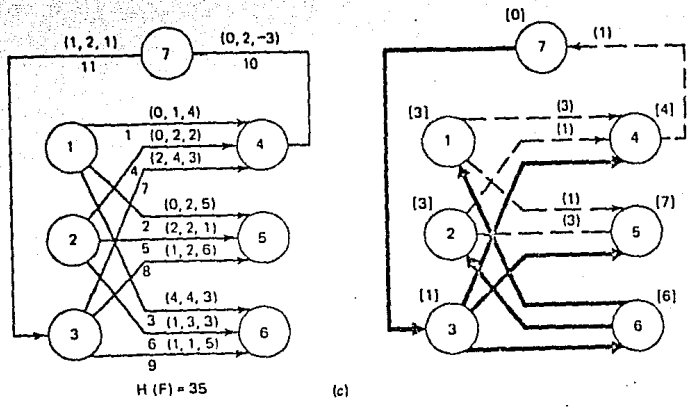


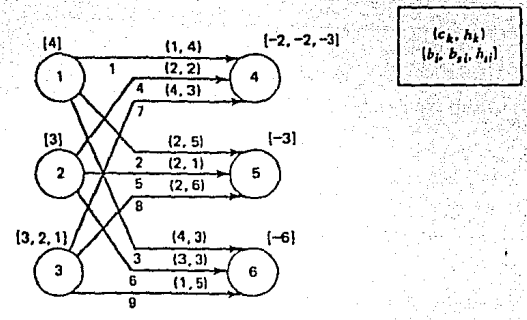
Figura 6.10 (continuación)
Aplicación del algoritmo primal.

Se cambian los potenciales en los nodos, para los nodos en el árbol enraizado en el nodo 6, que son los nodos 6, 1 y 2; -- por tanto d_k es positivo para todos los arcos admisibles no básicos, finalmente la solución óptima ha sido obtenida.

El algoritmo básico tiene ventajas sobre el algoritmo no - básico, ya que los ciclos negativos son fáciles de descubrir e identificar usando la base.

En el anexo A-2 se presenta la subrutina principal FCMI66, codificada en FORTRAN, la misma que utiliza el método del arco artificial (algoritmo primal factible ARTIFIC) para obtener una

base inicial artificial factible y después el método primal básico (PRIMAL) para obtener la solución óptima; de la red de la figura 6.3a, cuyos datos de entrada y resultados se pueden observar en las figuras 6.11 y 6.12, respectivamente.




```
100     66
200      6
300      1     4.
400      2     3.
500      3     3.          2.          1.
600      4    -2.         -2.         -3.
700      5    -3.
800      6    -6.
900
1000     1     4     0.          1.          4.
1100     1     5     0.          2.          5.
1200     1     6     0.          4.          3.
1300     2     4     0.          2.          2.
1400     2     5     0.          2.          1.
1500     2     6     0.          3.          3.
1600     3     4     0.          4.          3.
1700     3     5     0.          2.          6.
1800     3     6     0.          1.          5.
1900
#
```

Fig. 6.11

EL NRO. CLAVE DE ESTA SUBRUTINA ES : 66

PROGRAMA FCMI66
PRIMAL BASICO PARA FLUJO A COSTO MINIMO

REPRESENTACION DE LA RED--

NRO. DE NODOS-- 7 NRO. DE ARCOS-- 11

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	4
2	3
3	3
4	-2
5	-3
6	-6
7	0

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	O(K)	T(K)	C(K)	H(K)
1	1	4	1	4
2	1	5	2	5
3	1	6	4	3
4	2	4	2	2
5	2	5	2	1
6	2	6	3	3
7	3	4	4	3
8	3	5	2	6
9	3	6	1	5
10	4	7	2	-3
11	7	3	2	1

FORMACION DE LA RED CON ARCOS ARTIFICIALES

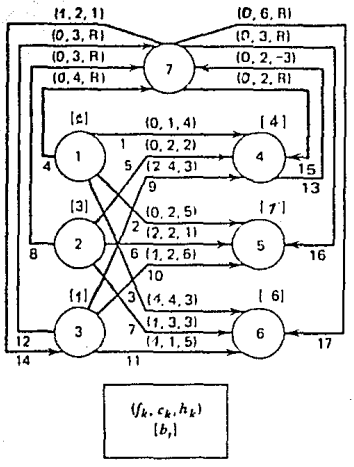
NRO. DE NODOS-- 7 NRO. DE ARCOS-- 17

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	4
2	3
3	3
4	-2
5	-3
6	-6
7	0

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	O(K)	T(K)	C(K)	H(K)	F(K)
1	1	4	1	4	0
2	1	5	2	5	0
3	1	6	4	3	0
4	1	7	4	9999	4
5	2	4	2	2	0
6	2	5	2	1	0
7	2	6	3	3	0
8	2	7	3	9999	3
9	3	4	4	3	0
10	3	5	2	6	0
11	3	6	1	5	0
12	3	7	3	9999	3
13	4	7	2	-3	0
14	7	3	2	1	0
15	7	4	2	9999	2
16	7	5	3	9999	3
17	7	6	6	9999	6



SOLUCION OPTIMA DEL PROBLEMA

NODO I	P(I)	PB(I)
1	2	-2
2	3	-7
3	1	14
4	4	9
5	7	10
6	6	11
7	0	0

ARC K	O(K)	T(K)	FLUJO	CAPACIDAD	COST/UNI.	COST.FLUJO
1	1	4	0	1	4	0
2	1	5	0	2	5	0
3	1	6	4	4	3	12
4	1	7	0	4	9999	0
5	2	4	0	2	2	0
6	2	5	2	2	1	2
7	2	6	1	3	3	3
8	2	7	0	3	9999	0
9	3	4	2	4	3	6
10	3	5	1	2	6	6
11	3	6	1	1	5	5
12	3	7	0	3	9999	0
13	4	7	0	2	-3	0
14	7	3	1	2	1	1
15	7	4	0	2	9999	0
16	7	5	0	3	9999	0
17	7	6	0	6	9999	0

COSTO TOTAL = 35

FIN DEL PROGRAMA RFCLSG

EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES: 0.366667

#ET=3:22.8 PT=0.5 IO=0.2

Fig. 6.12

ANEXO A - 1

MANUAL DEL USUARIO

1. Próposito
2. Tipos de redes de flujo que resuelve
3. Codificación de datos de entrada y descripción de variables

1. PROPOSITO

Este manual tiene la finalidad de proporcionar al usuario los conocimientos necesarios para que pueda manejar el programa principal RFCLSG, codificado en FORTRAN, e implementado en la computadora B7800.

2. TIPOS DE REDES DE FLUJO QUE RESUELVE

Este programa resuelve tres tipos de problemas de redes de flujo, aplicando varios métodos y en diferentes condiciones.

Estos problemas son:

a. Costo mínimo (Ruta más corta)

Este problema se resuelve por cinco métodos, que son:

a.1 Método de DIJKTRA, implementado por el algoritmo DSHORT, se aplica cuando los costos asociados a cada arco en la red son positivos.

La subrutina principal que resuelve el problema con este método es COMI43 (número clave 43)

a.2 Método PRIMAL, implementado por el algoritmo PSHORT, se aplica cuando algunos arcos tienen costos negativos, con ciclos positivos. La subrutina principal que resuelve el problema con este método es COMI44 (Número clave 44)

a.3 Método COMBINADO (de los algoritmos DSHORT y PSHORT), implementado por el algoritmo SHORT, se aplica cuando algunos arcos tienen costos negativos y con uno o más ciclos negativos. La subrutina principal que resuelve el problema con este método es COMI45 (número clave 45)

a.4 Método NO BASICO, implementado por el algoritmo NBSHORT, se aplica cuando algunos arcos tienen costos negativos. La subrutina principal que resuelve el problema con este método es COMI46 (número clave 46)

a.5 Método DUAL, implementado por el algoritmo DUALSP, se aplica cuando algunos arcos tienen costos negativos. La subrutina principal

que resuelve el problema con este método es COMI47 (número clave 47)

b. Flujo máximo

Este problema se resuelve por dos métodos, que son:

b.1 Método NO BASICO, implementado por los algoritmos MAXFLO Y LABEL1. La subrutina principal que resuelve el problema con este método es FLMA 54 (número clave 54)

b.2 Método BASICO, implementado por los algoritmos MAXFLO y LABEL2. La subrutina principal que resuelve el problema con este método es FLMA 55 (número clave 55)

c. Flujo a costo mínimo

Este problema se resuelve por dos métodos que son:

c.1 Método PRIMAL NO BASICO, implementado por el algoritmo PRIMA1, el cual inicialmente usa el método de FLUJO MAXIMO, implementado por el algoritmo primal factible PHASE1, para obtener una solución inicial primal factible. La subrutina principal que resuelve el problema con este método es FCMI65 (número clave 65)

c.2 Método PRIMAL BASICO, implementado por el algoritmo PRIMAL, el cual inicialmente usa el método de DEL ARCO ARTIFICIAL, implementado por el algoritmo primal factible artificial ARTIFIC, para obtener una solución básica inicial artificial factible. La subrutina principal que resuelve el problema con este método es FCMI66 (número clave 66)

3. CODIFICACION DE DATOS DE ENTRADA Y DESCRIPCION DE VARIABLES

Cuando el usuario desee resolver un problema por medio de redes de flujo, primeramente tiene que estructurarlo en forma de una red de flujo, enumerando en forma ordenada los nodos y los arcos, y determinando los parámetros asociados tanto a los nodos como a los arcos.

Antes de entrar a la codificación de datos de entrada, es importante conocer perfectamente el significado del número clave, que se encuentra definido por la variable NS. A esta variable se le puede asignar solamente los valores 43, 44, 45, 46, 47, 54, 55, 65 y 66. Por ejemplo si se le asigna el

valor de 43, de acuerdo a lo indicado en el numeral 2, llama a la subrutina -- COMI43 que resuelve el problema de costo mínimo empleando el método de DIJKSTRA. Si se le asigna el número 44, llama a la subrutina COMI44 que resuelve el proble ma de costo mínimo empleando el método PRIMAL y así sucesivamente.

CODIFICACION DE LOS DATOS DE ENTRADA PARA RESOLVER EL PROBLEMA DE COSTO MINIMO

TARJETA TIPO	COLUMNAS	FORMATO	VARIABLE	DESCRIPCION
1	1 - 5	I5	NS	Número clave que llama a la subrutina principal.
2	1 - 5	I5	SN	Número del nodo fuente o raíz, que generalmente es 1
3	1 - 5	I5	N	Número de nodos de la red original.
DATOS DE LOS NODOS (Una tarjeta por cada nodo)				
4	1 - 5	I5	I	Número asignado al nodo
	6 - 15	F10.0	BF	Flujo externo fijo en el no do I.
	16 - 25	F10.0	BS	Cota superior del flujo de holgura en el nodo I.
	26 - 35	F10.0	CS	Costo unitario del flujo de holgura en el nodo I.
5	Tarjeta en blanco después de los datos de los nodos.			

	TARJETA TIPO	COLUMNAS	FORMATO	VARIABLE	DESCRIPCION
		DATOS DE LOS ARCOS (UNA TARJETA POR CADA ARCO)			
	6	1 - 5	I5	I	Número del nodo origen del arco
		6 - 10	I5	J	Número del nodo terminal del arco
		11 - 20	F10.0	LOWER	Costo inferior del arco
		21 - 30	F10.0	UPPER	Costo superior del arco
		31 - 40	F10.0	COST	Costo unitario del arco
*	7	Tarjeta en blanco después de los datos de los arcos			
		ARBOL BASICO INICIAL FACTIBLE			
*	8	1 - 4	I4	PB(I)	Arco apuntador hacia atrás, que llega al nodo I
*		5 - 8	I4	PB(I)	
*		9 - 12	I4	PB(I)	
*		- - -	- - -	- - -	
*		77 - 80	I4	PB(I)	

* La tarjeta en blanco tipo 7 y la tarjeta de datos del árbol básico inicial factible; solamente se incluyen cuando se desea resolver el problema con la subrutina principal COMI44, o sea cuando se usa la clave 44.

CODIFICACION DE LOS DATOS DE ENTRADA PARA RESOLVER LOS PROBLEMAS DE FLUJO MAXIMO Y FLUJO A COSTO MINIMO

TARJETA TIPO	COLUMNAS	FORMATO	VARIABLE	DESCRIPCION
1	1 - 5	I5	NS	Número clave que llama a la subrutina principal
2	1 - 5	I5	N	Número de nodos de la red original
	DATOS DE LOS NODOS (una tarjeta por cada nodo)			
3	1 - 5	I5	I	Número asignado al nodo
	6 - 6	F10.0	BF	Flujo externo fijo en el nodo I
	16 - 25	F10.0	BS	Cota superior del flujo de holgura en el nodo I
	26 - 35	F10.0	CS	Costo unitario del flujo de holgura en el nodo I

TARJETA TIPO	COLUMNAS	FORMATO	VARIABLE	DESCRIPCION
4	Tarjeta en blanco después de los datos de los nodos			
	DATOS DE LOS ARCOS (una tarjeta por cada arco)			
5	1 - 5	I5	I	Número del nodo origen del arco
	6 - 10	I5	J	Número del nodo terminal del arco
	11 - 20	F10.0	LOWER	Cota inferior del arco
	21 - 30	F10.0	UPPER	Cota superior del arco
	31 - 40	F10.0	COST	Costo unitario del arco
* 6	Tarjeta en blanco después de los datos de los arcos			
* 7	1 - 5	I5	SN	Número del nodo fuente o raíz, que generalmente es 1.
*	6 - 10	I5	IN	Número del nodo terminal o sumidero
*	11 - 15	I5	VR	Flujo deseado
*	16 - 20	I5	IFLOW	Flujo inicial, que generalmente es cero

* La tarjeta en blanco tipo 6 y la tarjeta tipo 7 con sus respectivos datos, solamente se incluyen cuando se desea resolver el problema de flujo máximo con las subrutinas principales FLMA54 ó FLMA55, ó sea cuando se usan las claves 54 ó 55.

NOTA: El valor de las variables enteras con formato I debe ir impreso ajustándose a la derecha dentro del campo asignado.

Ejemplo: si la varible J tiene formato I5 cuyo valor es 20, se codifica

_ _ _ 2 0

El valor de las variables reales con formato F, puede ir impreso en cualquier posición dentro del campo asignado, siempre que sea del tipo FXX.0.
Ejem. Si la variable BF tiene formato F10.0 cuyo valor es 35., se codifica

3 5 . _ _ _ _ _ _ _ , _ _ _ _ _ 3 5 . _ _ ó _ _ _ _ _ _ _ 3 5 .

En los ejemplos de aplicación (anexo A-2) se puede ver la interpretación de los resultados.

ANEXO A-2

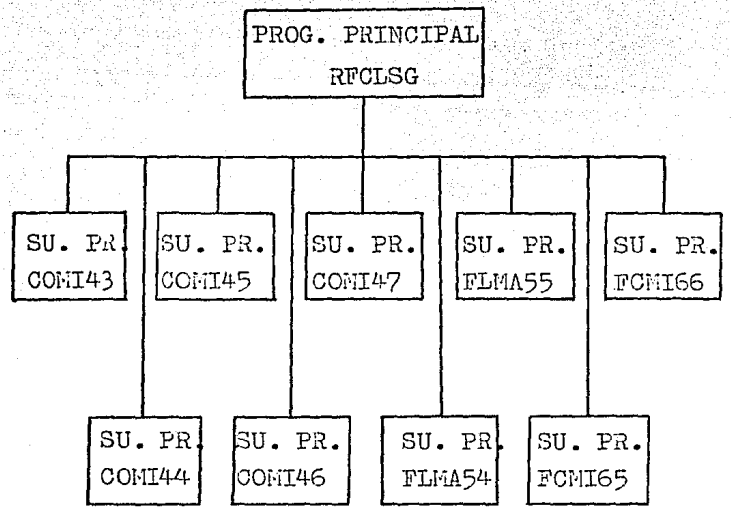
1. Estructura del programa principal RFCLSG(REDES DE FLUJO CON COSTOS LINEALES SIN GANANCIA).

2. Estructura de las subrutinas principales que resuelven - los problemas de costo mínimo (ruta más corta), flujo máximo y flujo a costo mínimo.

3. Programa RFCLSG, codificado en FORTRAN

4. Ejemplos de aplicación.

ESTRUCTURA DEL PROGRAMA PRINCIPAL



SIGNIFICADO DE LOS NOMBRES DEL PROGRAMA PRINCIPAL Y DE LAS SUBRUTINAS PRINCIPALES

RFCLSG : REDES DE FLUJO CON COSTOS LINEALES SIN GANANCIA

COMI43 : PROBLEMA DE COSTO MINIMO PRESENTADO EN EL CAPITULO 4, SECCION 4.3.

COMI44 : PROBLEMA DE COSTO MINIMO PRESENTADO EN EL CAPITULO 4, SECCION 4.4.

COMI45 : PROBLEMA DE COSTO MINIMO PRESENTADO EN EL CAPITULO 4, SECCION 4.5.

COMI46 : PROBLEMA DE COSTO MINIMO PRESENTADO EN EL CAPITULO 4, SECCION 4.6.

COMI47 : PROBLEMA DE COSTO MINIMO PRESENTADO EN EL CAPITULO 4, SECCION 4.7.

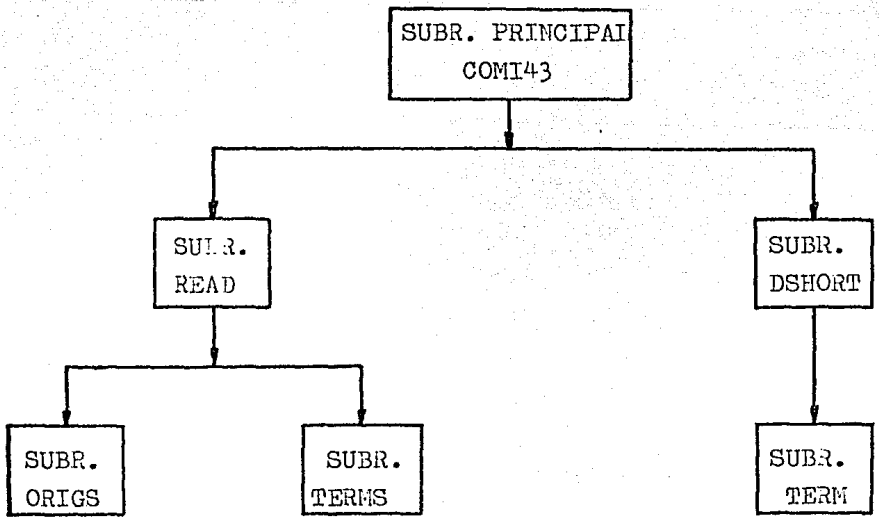
FLMA54 : PROBLEMA DE FLUJO MAXIMO PRESENTADO EN EL CAPITULO 5, SECCION 5.4.

FLMA55 : PROBLEMA DE FLUJO MAXIMO PRESENTADO EN EL CAPITULO 5, SECCION 5.5.

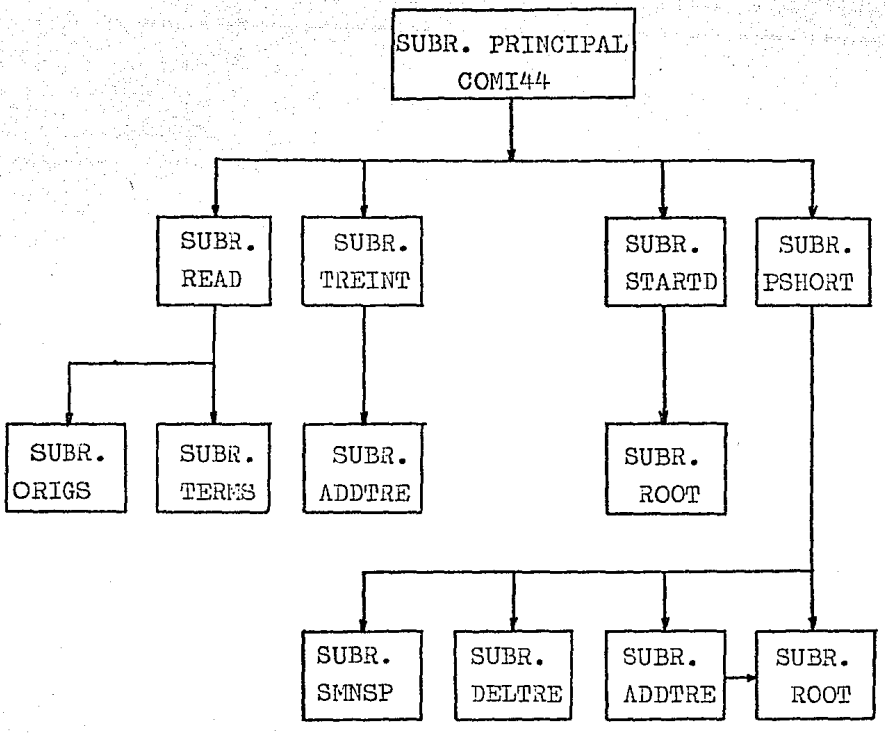
FCMI65 : PROBLEMA DE FLUJO A COSTO MINIMO PRESENTADO EN EL CAPITULO 6, SECCION 6.5.

FCMI66 : PROBLEMA DE FLUJO A COSTO MINIMO PRESENTADO EN EL CAPITULO 6, SECCION 6.6

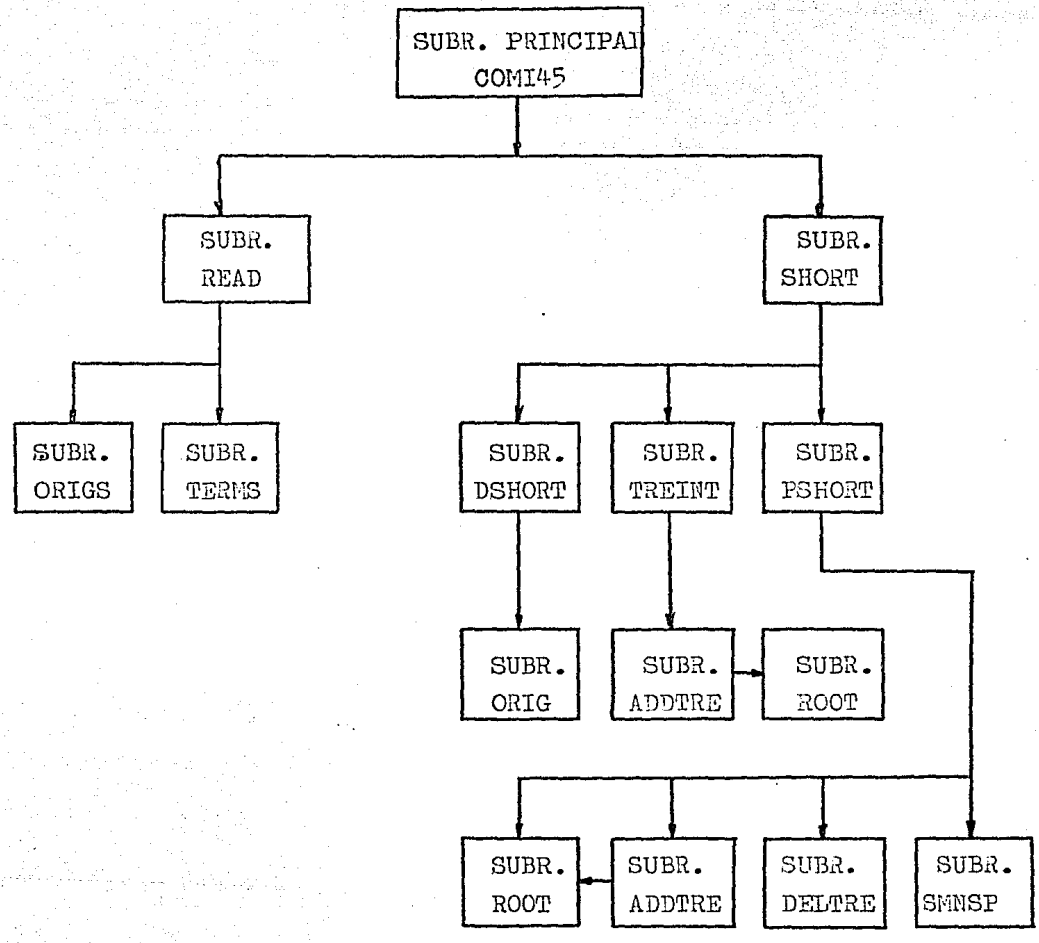
ESTRUCTURA DE LA SUBRUTINA PRINCIPAL COMI43



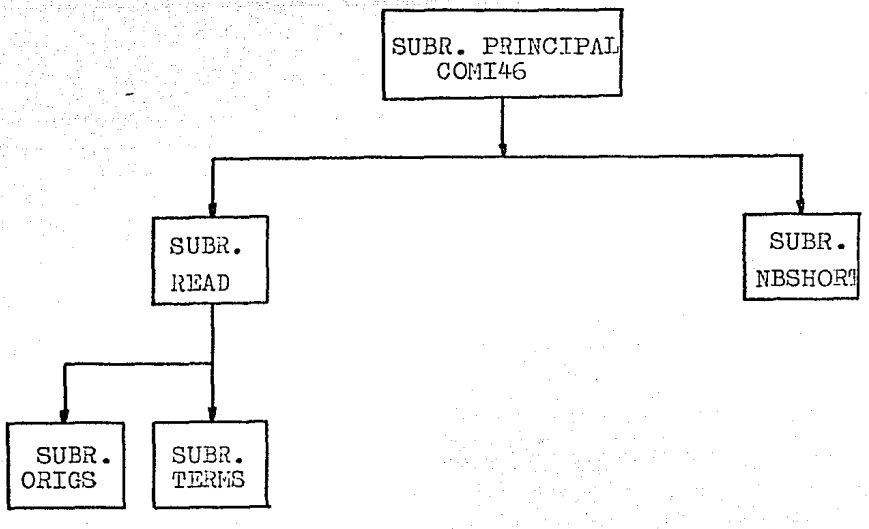
ESTRUCTURA DE LA SUBRUTINA PRINCIPAL COMI44



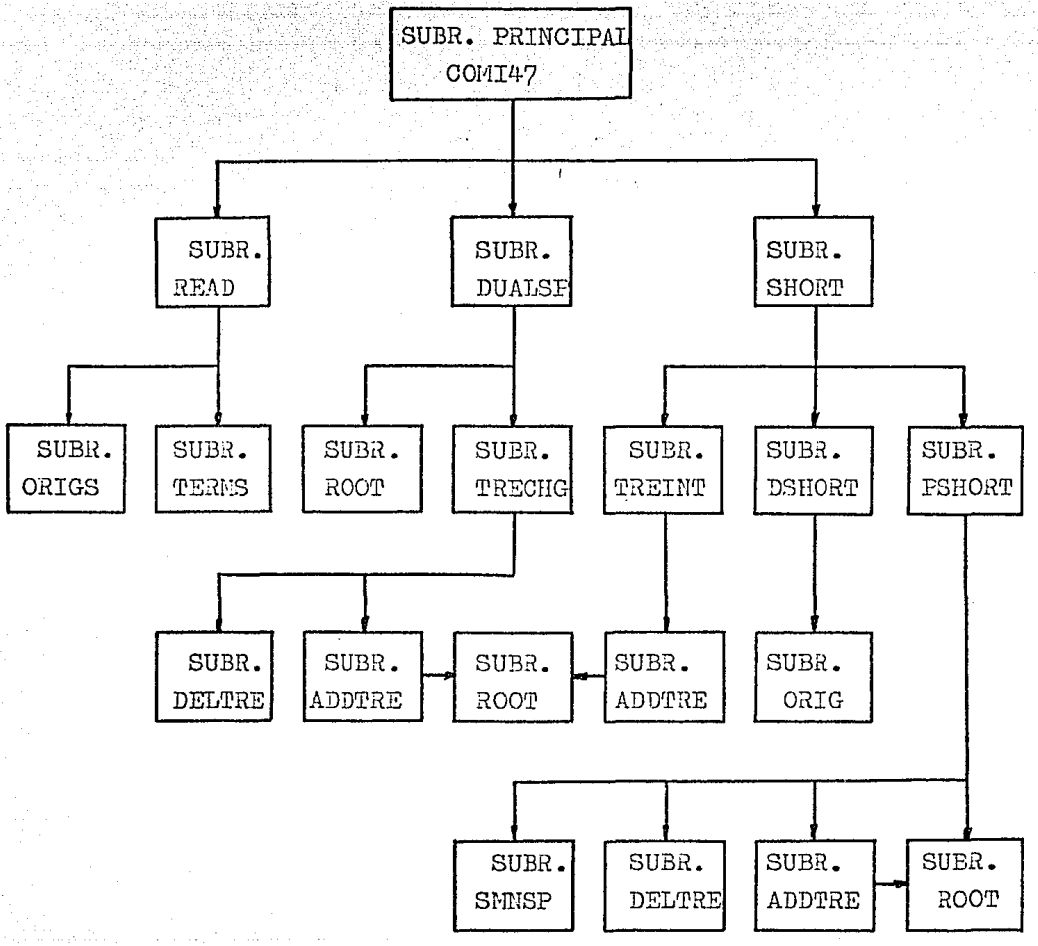
ESTRUCTURA DE LA SUBRUTINA PRINCIPAL COMI45



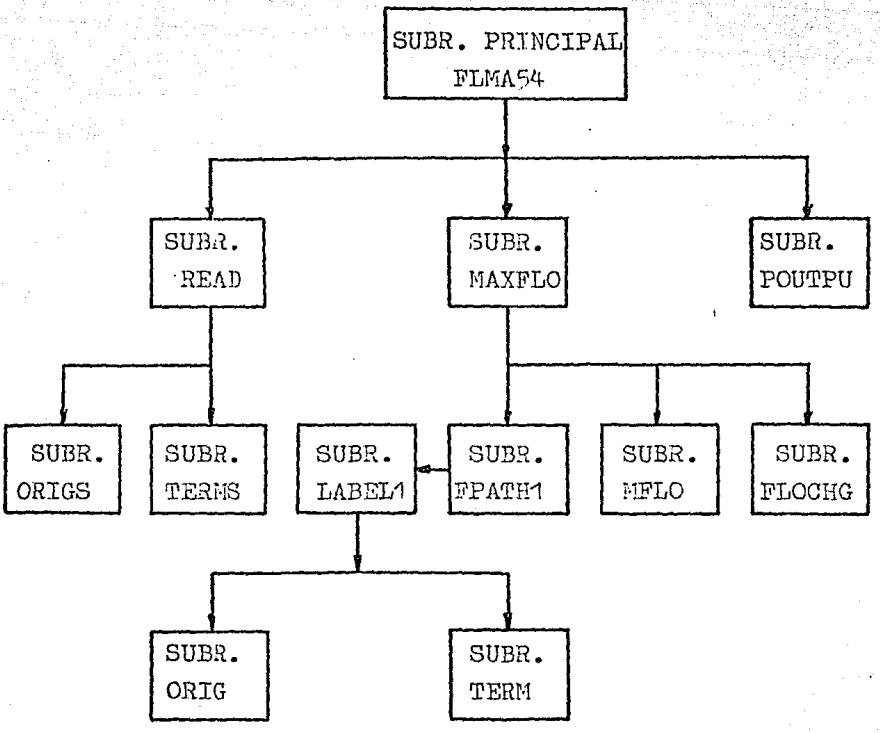
ESTRUCTURA DE LA SUBRUTINA PRINCIPAL COMI46



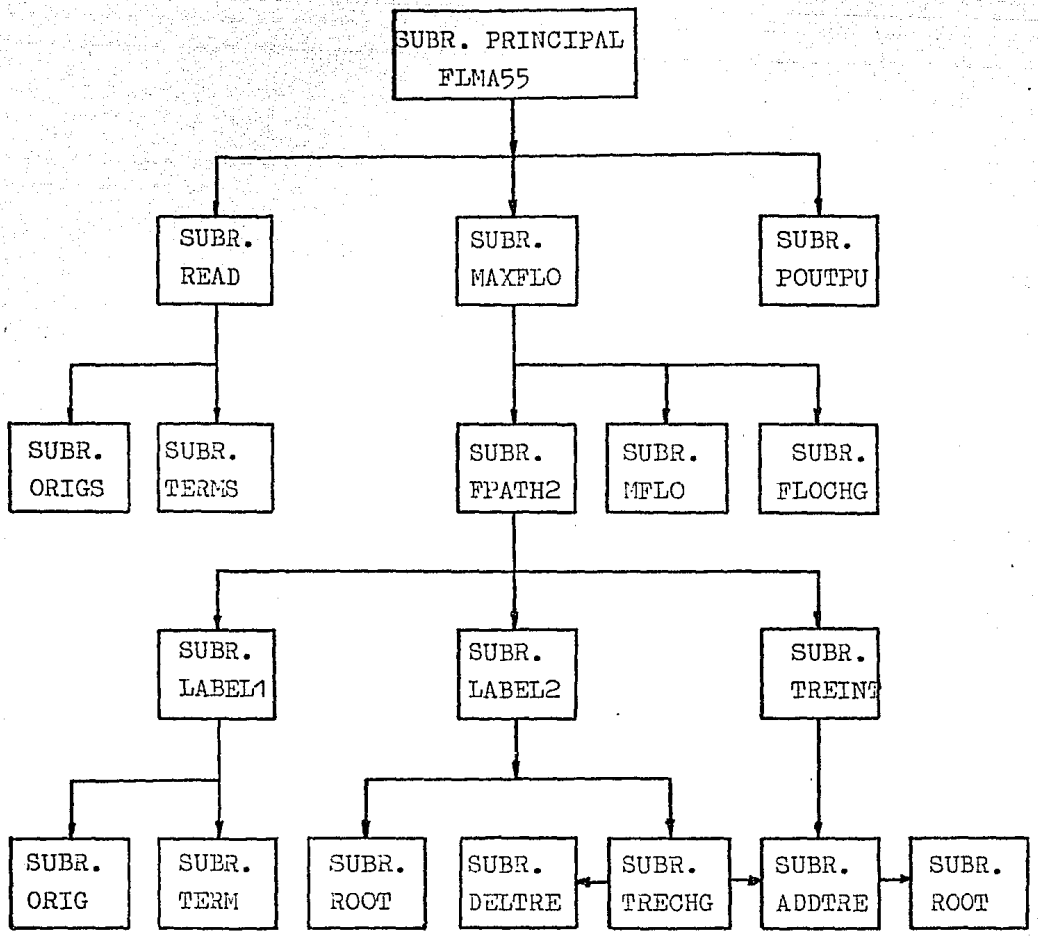
ESTRUCTURA DE LA SUBRUTINA PRINCIPAL COMI47



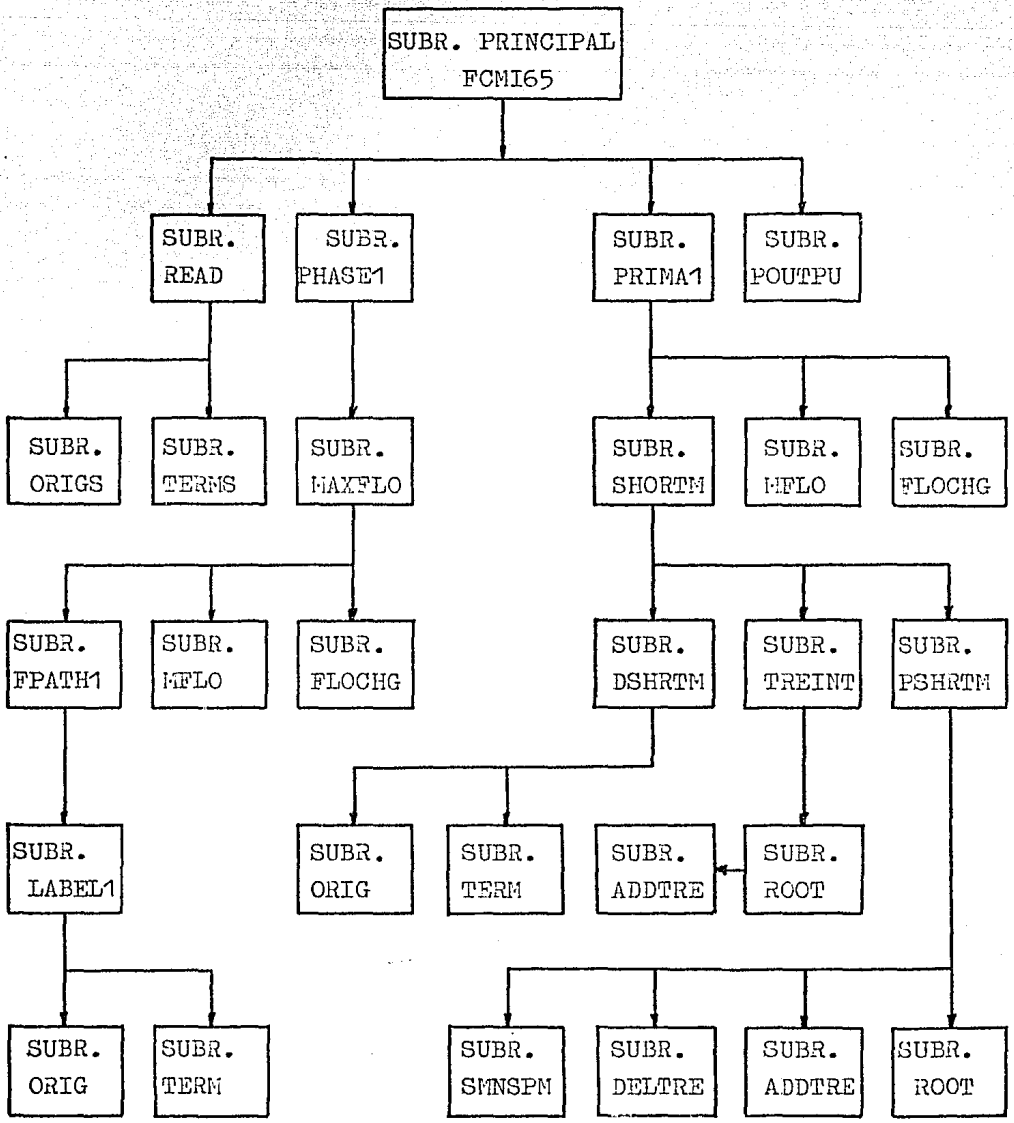
ESTRUCTURA DE LA SUBRUTINA PRINCIPAL FLMA54



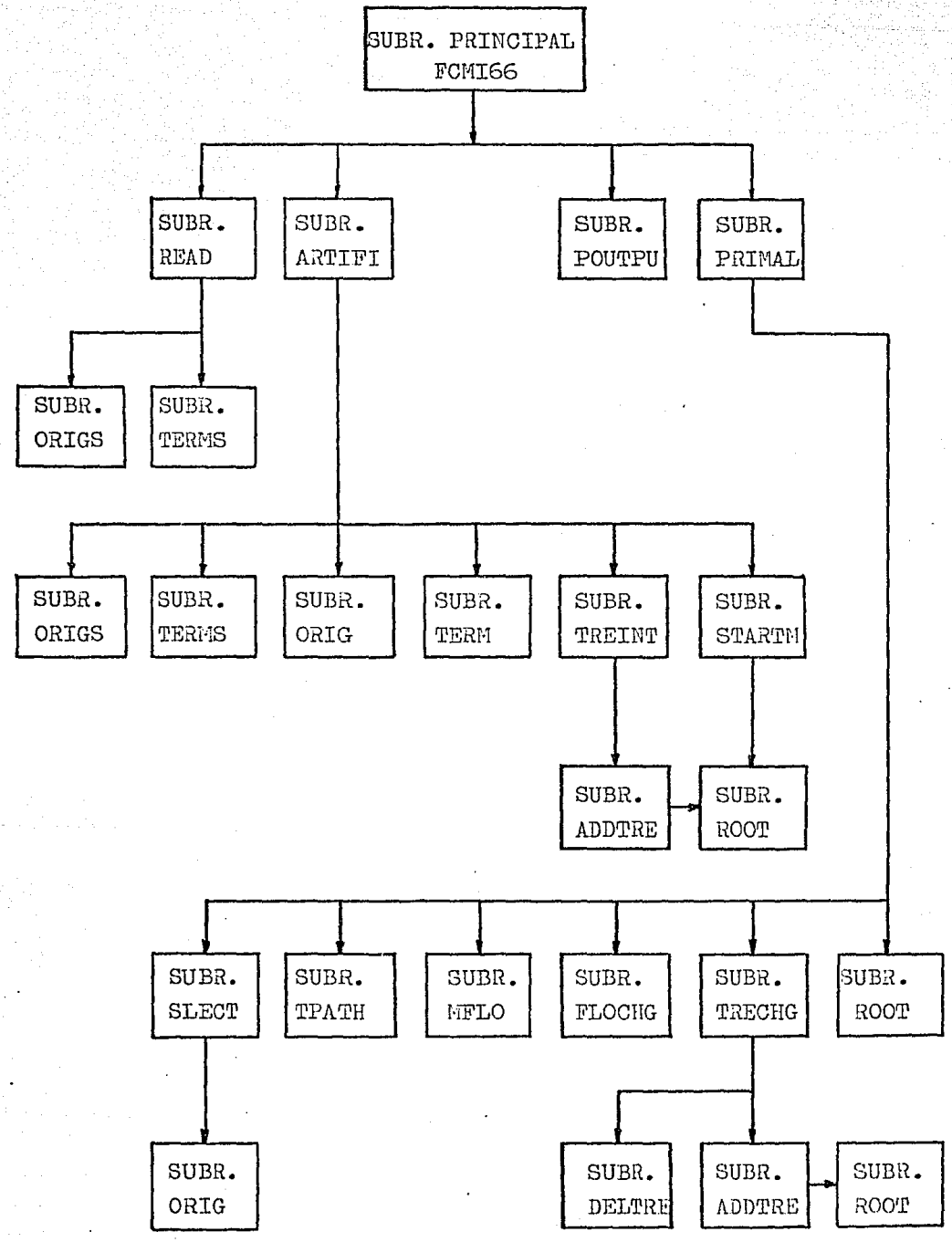
ESTRUCTURA DE LA SUBRUTINA PRINCIPAL FLMA55



ESTRUCTURA DE LA SUBRUTINA PRINCIPAL FCMI65



ESTRUCTURA DE LA SUBRUTINA PRINCIPAL FCMI66



```
50 %
60 %     PROGRAM RFCLSG (INPUT,OUTPUT)
70       COMMON /A/ O(0:120) /B/ T(0:120) /G/ M,N /H/ PT(0:120)
80       COMMON /C/ PO(0:120)  /D/ PF(0:120)  /E/ PR(0:120)  /P/ PI(0:120)
90      *   /I/ LT(0:120)  /J/ PO(0:120)  /K/ C(0:120)  /L/ H(0:120)
100     *   /M/ B(0:120)  /N/ CL(0:120)  /O/ F(0:120)
110      COMMON /R/PD(0:120)/S/PP(0:120)/NP/NP(0:120)
120      COMMON /SHRT/S(0:120)
130      DIMENSION LISA(0:120),LISN(0:120)
140      INTEGER PP,PD,CYC,AD
150      INTEGER O,T,PD,PF,PR,PT,LT,PO,C,H,B,CL,F,PI
160      INTEGER SC,SK,VR,SN,TN,V
170      INTEGER S
180 %
190 %    ************************************************************
200 %    *                                                          *
210 %    *         PROGRAMA PRINCIPAL :                             *
220 %    *         *******************                              *
230 %    *         *       RFCLSG      *                            *
240 %    *         *******************                              *
250 %    *         ASOCIADO CON LA TESIS, CAPITULOS :               *
260 %    *         ***********************************              *
270 %    *         * (CUATRO, CINCO Y SEIS (4,5 Y 6) *              *
280 %    *         ***********************************              *
290 %    *                                                          *
300 %    *  PROGRAMA INTEGRADO Y MODIFICADO POR EL ING. ALBERTO     *
310 %    *  CADENA LANDETA, COMO PARTE DE SU TESIS, PREVIA A LA     *
320 %    *  OBTENCION DEL GRADO DE MAESTRO EN INVESTIGACION  DE     *
330 %    *  DE OPERACIONES, EN LA DIVISION DE ESTUDIOS DE POST-     *
340 %    *  GRADO FACULTAD DE INGENIERIA. (D.E.P.F.I.)              *
350 %    *                                                          *
360 %    *  UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO (U.N.A.M.)      *
370 %    *                                                          *
380 %    *                  MEXICO, SEPTIEMBRE DE 1982              *
390 %    *                                                          *
400 %    *                                                          *
410 %    *  PROPOSITO : RESOLVER TRES TIPOS DE PROBLEMAS DE RE-     *
420 %    *  DES DE FLUJO, APLICANDO VARIOS METODOS Y EN DIFEREN-    *
430 %    *  TES CONDICIONES. LOS PROBLEMAS QUE RESUELVE SON :       *
440 %    *  1. COSTO MINIMO (RUTA MAS CORTA)                        *
450 %    *     1.1 CUANDO TODOS LOS ARCOS TIENEN COSTOS POSITIVOS,  *
460 %    *         USANDO EL METODO DE DIJKSTRA ALGORITHO DSHORT,   *
470 %    *         (SUBRUTINA PRINCIPAL COMI43).                    *
480 %    *     1.2 CUANDO ALGUNOS ARCOS TIENEN COSTOS NEGATIVOS     *
490 %    *         CON CICLOS POSITIVOS, USANDO EL ALGORITMO PRI-   *
500 %    *         MAL PSHORT. (SUBRUTINA PRINCIPAL COMI44).        *
510 %    *     1.3 CUANDO ALGUNOS ARCOS TIENEN COSTOS NEGATIVOS Y   *
520 %    *         CON UNO O MAS CICLOS NEGATIVOS, USANDO EL ALGO-  *
530 %    *         RITMO COMBINADO SHORT (COMBINACION DE LOS ALGO-  *
540 %    *         RITMOS DSHORT Y PSHORT). (SUBRUTINA PRINCIPAL    *
550 %    *         COMI45).                                         *
560 %    *     1.4 CUANDO ALGUNOS ARCOS TIENEN COSTOS NEGATIVOS,    *
570 %    *         USANDO EL ALGORITMO NO BASICO NBSHORT. (SUB-     *
580 %    *         RUTINA PRINCIPAL COMI46).                        *
590 %    *     1.5 CUANDO ALGUNOS ARCOS TIENEN COSTOS NEGATIVOS,    *
600 %    *         USANDO EL ALGORITMO DUAL DUALSP. (SUBRUTINA      *
610 %    *         PRINCIPAL COMI47).                               *
620 %    *  2. FLUJO MAXIMO                                         *
630 %    *     2.1 USANDO EL METODO NO BASICO ALGORITMOS MAXFLO     *
640 %    *         CON LABEL1. (SUBRUTINA PRINCIPAL FLMA54).        *
650 %    *     2.2 USANDO EL METODO BASICO  ALGORITMOS MAXFLO CON   *
660 %    *         LABEL2. (SUBRUTINA PRINCIPAL FLMA54).            *
670 %    *  3. FLUJO A COSTO MINIMO                                 *
680 %    *     3.1 PRIMERO USA EL METODO DE FLUJO MAXIMO ALGORIT-   *
690 %    *         MO PRIMAL FACTIBLE PHASE1  PARA OBTENER UNA SO-  *
700 %    *         LUCION PRIMAL FACTIBLE, Y LUEGO USA EL METODO    *
710 %    *         PRIMAL NO BASICO ALGORITMO PRIMAL PARA OBTENER   *
720 %    *         LA SOLUCION OPTIMA. (SUBRUTINA PRINCIPAL FCMI65)*
730 %    *     3.2 PRIMERO USA EL METODO DEL ARCO ARTIFICIAL ALGO-  *
740 %    *         RITMO PRIMAL FACTIBLE ARTIFIC PARA OBTENER UNA   *
750 %    *         BASE INICIAL ARTIFICIAL FACTIBLE, Y LUEGO USA    *
760 %    *         EL METODO PRIMAL BASICO ALGORITMO PRIMAL PARA    *
770 %    *         OBTENER LA SOLUCION OPTIMA. ( SUBRUTINA PRINCI-  *
780 %    *         PAL FCMI66).                                     *
790 %    *                                                          *
800 %    ************************************************************
```

```
C
C
C
      READ 333,NS
  333 FORMAT (I5)
      PRINT 933,NS
  933 FORMAT(//,' EL NRO. CLAVE DE ESTA SUBRUTINA ES :',I5,//)
C
C  LLAMA A LA SUBRUTINA PRINCIPAL QUE RESUELVE EL PROBLEMA
C
      IF (NS .NE. 43) GO TO 336
      TA=TIME(2)
      CALL COMI43
      GO TO 334
  336 IF (NS .NE. 44) GO TO 337
      TA=TIME(2)
      CALL COMI44
      GO TO 334
  337 IF (NS .NE. 45) GO TO 338
      TA=TIME(2)
      CALL COMI45
      GO TO 334
  338 IF (NS .NE. 46) GO TO 339
      TA=TIME(2)
      CALL COMI46
      GO TO 334
  339 IF (NS .NE. 47) GO TO 340
      TA=TIME(2)
      CALL COMI47
      GO TO 334
  340 IF (NS .NE. 54) GO TO 341
      TA=TIME(2)
      CALL FLMA54
      GO TO 334
  341 IF (NS .NE. 55) GO TO 342
      TA=TIME(2)
      CALL FLMA55
      GO TO 334
  342 IF (NS .NE. 65) GO TO 343
      TA=TIME(2)
      CALL FCMI65
      GO TO 334
  343 IF (NS .NE. 66) GO TO 344
      TA=TIME(2)
      CALL FCMI66
      GO TO 334
  344 PRINT 345
  345 FORMAT (///,8X,'NO DIO BIEN EL NUMERO CLAVE DE LA SUBRUTINA',/
     *           ,8X,'PRINCIPAL PARA RESOLVER EL PROBLEMA QUE UD',/
     *           ,8X,'DESEA. POR FAVOR REVISE BIEN EL MANUAL DEL',/
     *           ,8X,'USUARIO A-1 PARA QUE DE  ACUERDO AL TIPO',/
     *           ,8X,'DE PROBLEMA SELECCIONE EL NUMERO QUE DEBE ',/
     *           ,8X,'IMPRIMIR EN EL PRIMER RENGLON DE LOS DATOS',/
     *           ,8X,'DE ENTRADA CON FORMATO I5')
C
  334 PRINT 346
  346 FORMAT (//,' FIN DEL PROGRAMA RFCLSG')
      TR=(TIME(2)-TA)/60
      PRINT 351,TR
  351 FORMAT(//,' EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES:',F10.6,//)
C
      STOP
      END
C
C
C
C
      SUBROUTINE COMI43
      COMMON /A/ O(0:120)  /B/ T(0:120)  /G/ M,N  /H/ PT(0:120)
      COMMON /C/ PB(0:120)  /D/ PF(0:120)  /F/ PR(0:120)  /P/ PI(0:120)
     * /I/LT(0:120)/J/PO(0:120)/K/C(0:120)/R/PD(0:120)/U/PP(0:120)
     * /L/ H(0:120) /M/ B(0:120)  /N/ CL(0:120)  /O/ F(0:120)
      INTEGER O,T,PB,PF,PR,PT,LT,PO,C,H,B,CL,F,PI,PP,PD,SN,TN
```

```
1530 %
1540 %     ************************************************************
1550 %     *                                                          *
1560 %     *          SUBRUTINA PRINCIPAL :                           *
1570 %     *                                                          *
1580 %     *         **********************                           *
1590 %     *         *     COMI43         *                           *
1600 %     *         **********************                           *
1610 %     *         ASOCIADO CON EL CAPITULO :                       *
1620 %     *         **********************                           *
1630 %     *         *     CUATRO(4)      *                           *
1640 %     *         **********************                           *
1650 %     *                                                          *
1660 %     *  PROPOSITO : RESUELVE EL PROBLEMA DE RUTA MAS CORTA      *
1670 %     *  CUANDO TODOS LOS ARCOS ADMISIBLES TIENEN COSTOS         *
1680 %     *  POSITIVOS E IMPLEMENTA EL ALGORITMO DSHORT DEL CAPITU-  *
1690 %     *  LO 4, SECCION 4.3.                                      *
1700 %     *                                                          *
1710 %     *    LAS SUBRUTINAS REQUERIDAS SON LOS ALGORITMOS :        *
1720 %     *    DSHORT,    READ,       ORIGS,    TERMS,     ORIG     *
1730 %     *                                                          *
1740 %     ************************************************************
1750 %
1760       PRINT 100
1770   100 FORMAT(/,' PROGRAMA COMI43',/,
1780      *' DIJKSTRA, COSTO MINIMO CON COSTOS POSITIVOS',//)
1790 %
1800       READ 101,SN,TN
1810   101 FORMAT(2I5)
1820       PRINT 102,SN
1830   102 FORMAT(' EL NODO',I5,'  ES EL NODO RAIZ DE LA RUTA MAS CORTA')
1840       CALL READ
1850 %
1860       N=N-1
1880 %
1890       IF(TN.EQ.0) GO TO 65
1900       PRINT 80,TN
1910    80 FORMAT(//,' NODO',I4,'ES EL NODO TERMINAL DE LA RUTA MAS CORTA')
1920       GO TO 58
1930    65 CONTINUE
1940 %     PRINT 10
1950 %  10 FORMAT(/,' CUANDO TN=0 SE OBTIENE LA RUTA MAS CORTA')
1960 %
1970 %
1980 %     RUTA MAS CORTA, O ARBOL SOLUCION DE LA RUTA MAS CORTA___
1990 %     TODOS LOS ARCOS CON COSTOS (+), USANDO EL ALGORITMO DIJKSTRA
2000 %     SN=NODO RAIZ O NODO FUENTE
2010 %
2020       CALL DSHORT(SN,TN,NP)
2021       PRINT 703
2022   703 FORMAT(//,' LA SOLUCION OPTIMA DEL PROBLEMA ES :',/)
2030       PRINT 99
2040    99 FORMAT(//,9X,'I',7X,'P(I)',5X,'PB(I)',6X,'O(I)',6X,'T(I)',/)
2050    57 FORMAT (5I10)
2060    58 PRINT 57,(I,PI(I),PB(I),O(PB(I)),T(PB(I)),I=1,N)
2070 %
2080       IF(NP.NE.1) GO TO 20
2090       IF(TN.NE.0) PRINT 30,SN,TN
2100    30 FORMAT(//,'NO EXISTE RUTA ENTRE EL NODO',I4,'AL NODO',I4)
2110       IF(TN.EQ.0) PRINT 40,SN
2120    40 FORMAT(//,' NO EXISTE ARBOL ENRAIZADO DE RUTA MAS CORTA AL NODO
2130      *',I4)
2140    20 RETURN
2150       END
2160
2170 %
2180 %
2190 %
```

```
      SUBROUTINE COMI44
      COMMON /A/ O(0:120)  /B/ T(0:120)  /G/ M,N  /H/ PT(0:120)
      COMMON /C/ PB(0:120)  /B/ PF(0:120)  /F/ PR(0:120)  /P/ PI(0:120)
     *   /I/ LT(0:120)  /J/ PO(0:120)  /K/ C(0:120)  /L/ H(0:120)
     *   /M/ B(0:120)  /N/ CL(0:120)  /O/ F(0:120)
      COMMON/R/PD(0:120)/O/PP(0:120)
      INTEGER PP,PD,CYC
      INTEGER O,T,PB,PF,PR,PT,LT,PO,C,H,B,CL,F,PI,SN
%
%     ************************************************************
%     *                                                          *
%     *       SUBRUTINA PRINCIPAL :                              *
%     *                                                          *
%     *       **********************                             *
%     *       *     COMI44         *                             *
%     *       **********************                             *
%     *       ASOCIADO CON EL CAPITULO :                         *
%     *       **********************                             *
%     *       *     CUATRO(4)      *                             *
%     *       **********************                             *
%     *                                                          *
%     *  PROPOSITO : RESUELVE EL PROBLEMA DE RUTA MAS CORTA      *
%     *  PARTIENDO DE UNA SOLUCION BASICA FACTIBLE. EL USUARIO   *
%     *  PROPORCIONA EL ARBOL BASICO INICIAL POR MEDIO DE LOS    *
%     *  APUNTADORES HACIA ATRAS PB(I) A CADA NODO DEL ARBOL.    *
%     *  IMPLEMENTA EL ALGORITMO PSHORT DEL CAPITULO 4, SECCION  *
%     *  4.4.                                                    *
%     *                                                          *
%     *  LAS SUBRUTINAS REQUERIDAS SON LOS ALGORITMOS :          *
%     *      PSHORT,    SMNSP,    READ,    ORIGS,    TERMS,     *
%     *      START0,    TREINT,   DELTRE,  ROOT,     ADDTRE.    *
%     *                                                          *
%     ************************************************************
%
%
      READ 100,SN
      PRINT 110,SN
  110 FORMAT(' PROGRAMA COMI44',/,' PRIMAL DE RUTA MAS CORTA'
     *,//,' EL USUARIO DEBE PROPORCIONAR UN ARBOL BASICO INICIAL',///,
     *' EL NODO RAIZ ES ',I3)
%
      CALL READ
%
%     LEE EL ARBOL E IMPRIME LOS APUNTADOES
%
  100 FORMAT (2I5)
%     PRINT 36
%  36 FORMAT (////,10X,'CONSTRUCCION DEL ARBOL--',/ )
      READ 37, (PB(I),I=1,N)
   37 FORMAT (20I4)
      CALL TREINT(N)
      N = N-1
%
%
   57 FORMAT (9X,'I',6X,'PI(I)',5X,'PB(I)',6X,'O(I)',6X,'T(I)',
     *   //,(5I10))
%
   58 FORMAT(//,' RUTA MAS CORTA DESDE EL NODO RAIZ ',I3,/)
%
%
%      ARBOL DE RUTA MAS CORTA EN UNA RED CON ALGUN ARCO (-)
%      LA BASE INICIAL DEL ARBOL ES DADA POR EL USUARIO
%      SN=NODO FUENTE O RAIZ
      CALL START0(SN)
%     PRINT 91,(I,PI(I),I=1,N)
%  91 FORMAT(2I10)
      CALL PSHORT(CYC,J)
      IF(CYC.NE.0) PRINT 10,J
   10 FORMAT(' UN CICLO NEGATIVO FUE ENCONTRADO; PUEDE SER UBICADO
     *POR LOS APUNTADORES HACIA ATRAS PB(I),DESDE EL NODO',I3)
      IF(CYC.NE.0) PRINT 20,SN
   20 FORMAT(' EL CICLO BUSCADO SE ENCUENTRA,
     *BAJO ',I5)
      IF(CYC.EQ.0) PRINT 50,SN
      PRINT 57,(I,PI(I),PB(I),O(PB(I)),T(PB(I)),I=1,N)
      RETURN
      END
%
```

```
2980 %
2990 %
3000 %
3010       SUBROUTINE COMI45
3020       COMMON /A/ O(0:120)  /B/ T(0:120)  /G/ M,N  /H/ PT(0:120)
3030       COMMON /C/ PB(0:120)  /D/ PF(0:120)  /F/ PR(0:120)  /P/ PI(0:120)
3040      *   /I/ LT(0:120)  /J/ PO(0:120)  /K/ C(0:120)  /L/ H(0:120)
3050      *   /M/ B(0:120)  /N/ CL(0:120)  /O/ F(0:120)
3060       COMMON/R/PD(0:120)/Q/PP(0:120)/NP/NP
3070       DIMENSION LISA(0:120),LISN(0:120)
3080       INTEGER PP,PD,CYC
3090       INTEGER O,T,PB,PF,PR,PT,LT,PO,C,H,B,CL,F,PI,SN,TN
3100 %
3110 %     ************************************************************
3120 %     *                                                          *
3130 %     *       SUBRUTINA PRINCIPAL:                               *
3140 %     *                                                          *
3150 %     *       **********************                             *
3160 %     *       *     COMI45         *                             *
3170 %     *       **********************                             *
3180 %     *    ASOCIADO CON EL CAPITULO :                            *
3190 %     *       **********************                             *
3200 %     *       *     CUATRO(4)      *                             *
3210 %     *       **********************                             *
3220 %     *                                                          *
3230 %     * PROPOSITO : DETERMINAR EL ARBOL DE RUTA MAS CORTA EN     *
3240 %     * UNA RED QUE PUEDE TENER ARCOS CON COSTOS NEGATIVOS E     *
3250 %     * IMPLEMENTA EL ALGORITMO SHORT , PRESENTADO EN EL CAPI-   *
3260 %     * TULO 4 , SECCION 4.5.                                    *
3270 %     *                                                          *
3280 %     *  LAS SUBRUTINAS REQUERIDAS SON :                         *
3290 %     *    SHORT,  DSHORT, ORIG,  TREINT,  PSHORT,  SMNSP,       *
3300 %     *    READ,   ORIGS,  TERMS, DELTRE,  ADDTRE,  ROOT .       *
3310 %     *                                                          *
3320 %     ************************************************************
3330 %
3340       NP = 0
3350       CYC=0
3360       READ 100,SN
3370       PRINT 31,SN
3380    31 FORMAT(' PROGRAMA COMI45',/,' COMBINADO DE RUTA MAS CORTA'
3390      *,/,' CON UN ARBOL INICIAL DADO POR EL ALGORITMO DIJKSTRA'
3400      *,//,' EL NODO RAIZ ES ',I3)
3410 %
3420 %
3430       CALL READ
3440 %     LEE EN EL ARBOL E IMPRIME APUNTADORES
3450 %
3460   100 FORMAT (2I5)
3470    57 FORMAT (9X,'I',6X,'PI(I)',5X,'PB(I)',6X,'O(I)',6X,'T(I)',
3480      *   //,(5I10))
3490       N=N-1
3500 %
3510 %
3520 %
3530 %     ARBOL DE RUTA MAS CORTA EN UNA RED CON ALGUN ARCO (-)
3540 %     SN=NODO FUENTE O RAIZ
3550 %
3560       CALL SHORT (SN,CYC,J)
3570 %
3580       IF(NP.EQ.1) GO TO 999
3590       IF(CYC.NE.0) PRINT 10,J
3600    10 FORMAT(//,' UN CICLO NEGATIVO FUE ENCONTRADO '
3610      *,/,' SE LOCALIZA POR LOS APUNTADORES PB(I) DESDE EL NODO',I3,//)
3620 %
3630       IF(CYC.EQ.0) PRINT 58,SN
3640    58 FORMAT (////,5X,' RUTA MAS CORTA DESDE EL NODO RAIZ',I3
3650      *,/,'  A TODOS LOS DEMAS NODOS',//)
3660       PRINT 57, (I,PI(I),PB(I),O(PB(I)),T(PB(I)),I=1,N)
3670 %
3680       GO TO 63
3690   999 PRINT 91,SN
3700    91 FORMAT(' NO EXISTE RUTA MAS CORTA DESDE EL NODO RAIZ',I3)
3710    63 RETURN
3720       END
3730
```

```
3740 %
3750 %
3760 %
3770       SUBROUTINE COMI46
3780       COMMON /A/ O(0:120)  /B/ T(0:120)  /G/ M,N  /H/ PT(0:120)
3790       COMMON /C/ PB(0:120)/P/PI(0:120)
3800      *   /I/ LT(0:120)  /J/ PO(0:120)  /K/ C(0:120)  /L/ H(0:120)
3810      *   /M/ B(0:120)  /N/ CL(0:120)  /O/ F(0:120)
3820       INTEGER O,T,PB,PT,LT,PO,C,H,B,CL,F,PI,SN,CYC
3830 %
3840 %     ************************************************************
3850 %     *                                                          *
3860 %     *       SUBRUTINA PRINCIPAL:                               *
3870 %     *                                                          *
3880 %     *       **********************                             *
3890 %     *       *     COMI46         *                             *
3900 %     *       **********************                             *
3910 %     *     ASOCIADO CON EL CAPITULO :                           *
3920 %     *       ************************                           *
3930 %     *       *     CUATRO(4)        *                           *
3940 %     *       ************************                           *
3950 %     *                                                          *
3960 %     *  PROPOSITO : RESOLVER EL PROBLEMA DE RUTA MAS CORTA      *
3970 %     *  USANDO UN ARBOL NO BASICO. ADEMAS IMPLEMENTA EL         *
3980 %     *  ALGORITMO NBSHOR PRESENTADO EN EL CAPITULO 4, SECCION   *
3990 %     *  4.6.                                                    *
4000 %     *  LAS SUBRUTINAS REQUERIDAS SON : NBSHOR, READ, ORIGS     *
4010 %     *                                  TERMS .                 *
4020 %     *                                                          *
4030 %     ************************************************************
4040 %
4050       READ 100,SN
4060       PRINT 50, SN
4070    50 FORMAT(' PROGRAMA COMI46',/,' NO BASICO DE RUTA MAS CORTA',///,
4080      *' EL NODO RAIZ ES ',I3)
4090       CALL READ
4100 %
4110       N=N-1
4120   100 FORMAT (2I5)
4130    57 FORMAT (9X,'I',6X,'PI(I)',5X,'PB(I)',6X,'O(I)',6X,'T(I)',
4140      *   //,(5I10))
4150 %
4160    58 FORMAT(////,5X,'RUTA MAS CORTA DESDE EL NODO RAIZ',I3,
4170      * ' A TODOS LOS OTROS NODOS',//)
4180       DO 60 I = 1,N
4190    60 PB(I) = 0
4200       CALL NBSHOR (SN,CYC)
4210 %
4220       IF(CYC.NE.0) PRINT 10,J
4230    10 FORMAT(' UN CICLO NEGATIVO FUE ENCONTRADO PUEDE IDENTIFICARSE'
4240      *,/,' RECORRIENDO LOS APUNTADORES PB(I) DESDE EL NODO ',I3)
4250 %     IF(CYC.NE.0) PRINT 20
4260 %  20 FORMAT(' EL CICLO ENCONTRADO ESTA DADO POR LOS APUNTADORES,/
4270 %    *,EXPRESADOS EN LA SIGUIENTE TABLA :',//)
4280 %
4290       IF(CYC.EQ.0) PRINT 58,SN
4300       PRINT 57,(I,PI(I),PB(I),O(PB(I)),T(PB(I)),I=1,N)
4310       RETURN
4320       END
4330
4340 %
4350 %
4360 %
4370       SUBROUTINE COMI47
4380       COMMON /A/ O(0:120) /B/ T(0:120) /G/ M,N /H/ PT(0:120)
4390       COMMON /C/ PB(0:120)  /D/ PF(0:120)  /E/ PR(0:120)  /P/ PI(0:120)
4400      *   /I/ LT(0:120)  /J/ PO(0:120)  /K/ C(0:120)  /L/ H(0:120)
4410      *   /M/ B(0:120)  /N/ CL(0:120)  /O/ F(0:120)
4420       COMMON/R/PB(0:120)/U/PF(0:120)
4430       COMMON /SHRT/S(0:120)
4440       DIMENSION LISA(0:120),LISN(0:120)
4450       INTEGER PF,PB,CYC,AD
4460       INTEGER O,I,PB,PF,PR,PT,LT,PO,C,H,B,CL,F,PI
4470       INTEGER SC,SK,VR,SN,TN,V
4480       INTEGER S
```

```
4490 %
4500 %     ************************************************************
4510 %     *                                                          *
4520 %     *       SUBRUTINA PRINCIPAL:                               *
4530 %     *                                                          *
4540 %     *       **********************                             *
4550 %     *       *     COMI47         *                             *
4560 %     *       **********************                             *
4570 %     *      ASOCIADO CON EL CAPITULO 4                          *
4580 %     *       **********************                             *
4590 %     *       *     CUATRO(4)      *                             *
4600 %     *       **********************                             *
4610 %     *                                                          *
4620 %     *  PROPOSITO : RESUELVE EL PROBLEMA DE RUTA MAS CORTA.     *
4630 %     *  OBTIENE UN NUEVO ARBOL DE TRAYECTORIA MAS CORTA         *
4640 %     *  CUANDO UNO O MAS ARCOS DE ESTE ARBOL SE VUELVEN         *
4650 %     *  INADMISIVLES. IMPLEMENTA EL ALGORITMO  DUALPS ,         *
4660 %     *  PRESENTADO EN EL CAPITULO 4, SECCION 4.7.               *
4670 %     *                                                          *
4680 %     *  LAS SUBRUTINAS REQUERIDAS SON :                         *
4690 %     *    SHORT,     FSHORT,     ORIG,      TREINT,    SMNSP,   *
4700 %     *    READ,      ORIGS,      TERMS,     DUALSP,    TRECHG,  *
4710 %     *    DSHORT,    DELTRE,     ADDTRE,    ROOT.                *
4720 %     *                                                          *
4730 %     ************************************************************
4740 %
4750       CYC =0
4760       PRINT 61
4770    61 FORMAT (' PROGRAMA COMI47',/,' DUAL DE RUTA MAS CORTA')
4780       READ 100,SN
4790   100 FORMAT (2I5)
4800       CALL READ
4810       N=N-1
4820    57 FORMAT (9X,'I',6X,'PI(I)',5X,'PB(I)',6X,'O(I)',6X,'T(I)',
4830      *   //,(5I10))
4840 %
4850 %
4860 %
4870 %     ARBOL DE RUTA MAS CORTA CON ALGUN ARCO (-),
4880 %     Y ALGUNOS ARCOS INADMISIBLES
4890 %     SN=NODO FUENTE O RAIZ
4900 %
4910       CALL SHORT (SN,CYC,J)
4920       PRINT 58, SN,CYC,J
4930    58 FORMAT (////,5X,'RUTA MAS CORTA DESDE EL NODO RAIZ',I3,
4940      */,5X,'AL RESTO DE NODOS',5X,' CYC=',I3,' J=',I5,//)
4950       PRINT 57, (I,PI(I),PB(I),O(PB(I)),T(PB(I)),I=1,N)
4960       INF =0
4970       CALL DUALSP(SN,INF)
4980       IF ( INF .NE. 0) PRINT 300
4990   300 FORMAT (//,'LAS CONDICIONES DEL PROBLEMA NO PERMITEN UNA
5000      * SOLUCION FACTIBLE')
5010       IF (INF .EQ. 0) PRINT 58, SN,CYC,J
5020       PRINT 57,(I,PI(I),PB(I),O(PB(I)),T(PB(I)),I=1,N)
5030       RETURN
5040       END
5050
5060 %
5070 %
5080 %
5090       SUBROUTINE FLNA54
5100       COMMON /A/ O(0:120)  /B/ T(0:120)  /G/ M,N  /H/ PT(0:120)
5110       COMMON /C/ PB(0:120)  /D/ PF(0:120)  /E/ PR(0:120)  /P/ PI(0:120)
5120      *   /I/ LT(0:120)  /J/ PD(0:120)  /K/ C(0:120)  /L/ H(0:120)
5130      *   /M/ B(0:120)  /N/ CL(0:120)  /O/ F(0:120)
5140       COMMON/R/PD(0:120)/Q/PP(0:120)
5150       INTEGER PP,PD,AD
5160       INTEGER O,T,PB,PF,PR,PT,LT,PD,C,H,B,CL,F,PI
5170       INTEGER VR,SN,TN,V
5180 %
```

```
5170 %     ************************************************************
5200 %     *                                                          *
5210 %     *       SUBRUTINA PRINCIPAL:                               *
5220 %     *                                                          *
5230 %     *       **********************                             *
5240 %     *       *    FLMA54          *                             *
5250 %     *       **********************                             *
5260 %     *      ASOCIADO CON EL CAPITULO :                          *
5270 %     *       ************************                           *
5280 %     *       *      CINCO(5)        *                           *
5290 %     *       ************************                           *
5300 %     *                                                          *
5310 %     *  PROPOSITO : DADO UNA RED DIRIGIDA D=(N,M) CON CAPA-     *
5320 %     *  CIDAD EN CADA ARCO, SE CONSIDERA EL PROBLEMA DE EN-     *
5330 %     *  CONTRAR EL MAXIMO FLUJO TOTAL QUE PUEDE PASAR POR       *
5340 %     *  LA RED DESDE EL NODO FUENTE (SN) HASTA EL NODO SUMI-    *
5350 %     *  DERO (TN). EL USUARIO DEBE PROPORCIONAR UN FLUJO FAC-   *
5360 %     *  TIBLE INICIAL, CAPITULO 5, SECCION 5.4.                 *
5370 %     *                                                          *
5380 %     *  LAS SUBRUTINAS REQUERIDAS SON :                         *
5390 %     *     MAXFLO,     FPATH1,     LABEL1,      MFLO,     FLOCHG, *
5400 %     *     ORIG,       TERM,       READ,        ORIGS,       AD,  *
5410 %     *     POUTPU,     TERMS.                                   *
5420 %     *                                                          *
5430 %     ************************************************************
5440 %
5450       PRINT 501
5460    501 FORMAT (' PROGRAMA FLMA54',/,' NO BASICO DE FLUJO MAXIMO')
5470       CALL READ
5480 %
5490       N=N-1
5500       READ 110,SN,TN,VR,IFLOW
5510    110 FORMAT(4I5)
5520       PRINT 81, VR,SN,TN
5530     81 FORMAT (///,5X,'SOLUCION DEL MAX. FLUJO, FLUJO REQUERIDO',I5
5540      * ,//,' DESDE EL NODO FUENTE',I3,' AL NODO SUMIDERO',I3,//)
5550       IF  (IFLOW.NE.1) GO TO 80
5560     58 FORMAT(' TODOS LOS FLUJOS INICIALES SON CERO')
5570       IF (IFLOW.NE.1) GO TO 80
5580       PRINT 60
5590     60 FORMAT(' FLUJO FACTIBLE INICIAL DADO POR EL USUARIO')
5600 %
5610 %     LEE LOS FLUJOS INICIALES DE LA RED
5620       READ 84, (F(I),I=1,M)
5630     84 FORMAT (16I5)
5640       PRINT 86,   (F(I),I=1,M)
5650     86 FORMAT (//,10X,'FLUJOS INICIALES EN LOS ARCOS---',/, (16I5) / )
5660     80 CONTINUE
5670       CALL MAXFLO(SN,TN,VR,V,INF)
5680       IF(INF.NE.1) GO TO 85
5690       PRINT 82
5700     82 FORMAT(10X,'NO EXISTE RUTA ADICIONAL')
5710     85 CONTINUE
5720       IF(V.LT.VR) PRINT  83,V
5730     83 FORMAT (10X,'FLUJO MAXIMO POSIBLE =',I5,//)
5740       IF(V.GE.VR) PRINT 91,V
5750     91 FORMAT(' EL FLUJO DESEADO DE',I5,' UNI. HA SIDO PROCESADO')
5760       CALL POUTPU
5770       RETURN
5780       END
5790
5800 %
5810 %
5820 %
5830       SUBROUTINE FLMA55
5840       COMMON /A/ O(0:120)  /B/ T(0:120)  /G/ N,N  /H/ PT(0:120)
5850       COMMON /C/PR(0:120)/D/PF(0:120)/F/PR(0:120)/P/PI(0:120)
5860      *   /I/ LT(0:120)  /J/ PO(0:120)  /K/ C(0:120)  /L/ H(0:120)
5870      *   /M/ B(0:120)  /N/ CL(0:120)  /O/ F(0:120)
5880       COMMON/R/PD(0:120)/Q/PP(0:120)
5890       INTEGER PP,PD,AD
5900       INTEGER O,T,PD,PF,PR,PT,LT,PO,C,H,B,CL,F,PI
5910       INTEGER VR,SN,TN,V
```

```
5920 %
5930 %     ************************************************************
5940 %     *                                                          *
5950 %     *       SUBRUTINA PRINCIPAL:                               *
5960 %     *                                                          *
5970 %     *       **********************                             *
5980 %     *       *     FLMA55         *                             *
5990 %     *       **********************                             *
6000 %     *      ASOCIADO CON EL CAPITULO :                          *
6010 %     *       ************************                           *
6020 %     *       *       CINCO(5)       *                           *
6030 %     *       ************************                           *
6040 %     *                                                          *
6050 %     *  PROPOSITO : ENCONTRAR EL MAXIMO FLUJO TOTAL EN LA RED   *
6060 %     *  PROPORCIONANDO UN NUEVO ARBOL GENERADOR LUEGO DE QUE    *
6070 %     *  UNO O MAS ARCOS EN EL ARBOL ORIGINAL SE HACEN INADMI-   *
6080 %     *  SIBLES. CAP.5, SECCION 5.5.                             *
6090 %     *                                                          *
6100 %     *  LAS SUBRUTINAS REQUERIDAS SON :                         *
6110 %     *     MAXFLO,     FPATH2     LABEL2,     LABEL1,     TREINT*
6120 %     *     TRECHG,     MFLO,      FLOCHG,     AD,         READ, *
6130 %     *     ORIGS,      TERMS,     POUTPU,     ORIG,       TERM, *
6140 %     *     ADDTRE,     DELTRE,    ROOT.                         *
6150 %     *                                                          *
6160 %     ************************************************************
6170 %
6180       PRINT 503
6190   503 FORMAT (' PROGRAMA FLMA55',/,' BASICO DE FLUJO MAXIMO')
6200       CALL READ
6210 %
6220       N=N-1
6230       READ 110,SN,TN,VR,IFLOW
6240   110 FORMAT(4I5)
6250       PRINT 81, VR,SN,TN
6260    81 FORMAT (////,5X,'SOLUCION DEL MAX. FLUJO. FLUJO REQUERIDO',I4
6270      *,//,' DESDE EL NODO FUENTE',I3,' AL NODO SUMIDERO',I3,//)
6280       IF(IFLOW.NE.1) PRINT 58
6290    58 FORMAT(' TODOS LOS FLUJOS INICIALES SON CERO')
6300       IF (IFLOW.NE.1) GO TO 80
6310       PRINT 60
6320    60 FORMAT(' LOS FLUJOS INICIALES FACTIBLES LO DA EL USUARIO').
6330 %
6340 %        LEE LOS FLUJOS INICIALES DE LA RED
6350       READ 84, (F(I),I=1,M)
6360    84 FORMAT (16I5)
6370       PRINT 86,   (F(I),I=1,M)
6380    86 FORMAT (//,10X,'FLUJOS INICIALES EN LOS ARCOS---',/, (16I5) / )
6390    80 CONTINUE
6400       CALL MAXFLO(SN,TN,VR,V,INF)
6410       IF(INF.NE.1) GO TO 85
6420       PRINT 82
6430    82 FORMAT(10X,'NO EXISTE RUTA ADICIONAL')
6440    85 CONTINUE
6450       IF(V.LT.VR) PRINT  83,V
6460    83 FORMAT (10X,'MAXIMO FLUJO POSIBLE =',I5,//)
6470       IF(V.GE.VR) PRINT 91,V
6480    91 FORMAT(' EL FLUJO DESEADO DE',I5,' UNI. HA SIDO PROSESADO')
6490       CALL POUTPU
6500       RETURN
6510       END
6520
6530 %
6540 %
6550 %
6560       SUBROUTINE FCM165
6570       COMMON /A/ O(0:120)  /B/ T(0:120)  /G/ M,N   /H/ PT(0:120)
6580       COMMON /C/ PR(0:120)  /D/ PF(0:120)  /F/ PR(0:120)  /P/ PI(0:120)
6590      *   /I/ LT(0:120)  /J/ PO(0:120)  /K/ C(0:120)  /L/ H(0:120)
6600      *   /M/ D(0:120)  /N/ CL(0:120)  /O/ F(0:120)
6610       COMMON/R/PD(0:120)/U/PP(0:120)
6620       COMMON /SHRT/S(0:120)
6630       INTEGER PP,PD,CYC,AD
6640       INTEGER O,T,PD,PF,PR,PT,LT,PO,C,H,D,CL,F,PI
6650       INTEGER VR,SN,TN,V
6660       INTEGER S
```

```
6670 %
6680 %     ************************************************************
6690 %     *                                                          *
6700 %     *       SUBRUTINA PRINCIPAL:                               *
6710 %     *                                                          *
6720 %     *       **********************                             *
6730 %     *       *     FCMI65         *                             *
6740 %     *       **********************                             *
6750 %     *       ASOCIADO CON EL CAPITULO :                         *
6760 %     *       **********************                             *
6770 %     *       *       SEIS(6)      *                             *
6780 %     *       **********************                             *
6790 %     *                                                          *
6800 %     *  PROPOSITO : OBTENER LA SOLUCION OPTIMA PARA UN PROBLE-   *
6810 %     *  DE REDES DE FLUJO A COSTO MINIMO, PRIMERO ENCUENTRA     *
6820 %     *  UNA SOLUCION PRIMAL FACTIBLE USANDO EL METODO DEL       *
6830 %     *  CAPITULO 6, SECCION 6.3. EL RESULTADO  DE LA SOLUCION   *
6840 %     *  PRIMAL FACTIBLE ES LUEGO USADO PARA EL ALGORITMO PRIMAL *
6850 %     *  NO BASICO DEL CAPITULO 6, SECCION 6.5, PARA OBTENER     *
6860 %     *  LA SOLUCION OPTIMA.                                     *
6870 %     *                                                          *
6880 %     *     LAS SUBRUTINAS REQUERIDAS SON :                      *
6890 %     *    PHASE1,     PRIMA1,      SHORTM,      MFLO,          *
6900 %     *    FLOCHG,     DSHRTM,      TREINT,      PSHRTM,        *
6910 %     *    ORIG,       AD,          SMNSPM,      POUTPU,        *
6920 %     *    MAXFLO,     FPATH1,      LABEL1,      TERM,          *
6930 %     *    READ,       ORIGS,       TERMS,       DELTRE,        *
6940 %     *    ADDTRE,     ROOT.                                     *
6950 %     *                                                          *
6960 %     ************************************************************
6970       PRINT 505
6980   505 FORMAT(' PROGRAMA FCMI65',/,' PRIMAL NO BASICO PARA FLUJO A COSTO
6990      *MINIMO')
7000 %
7010 %
7020       CALL READ
7030       CALL PHASE1(INF)
7040 %     CALL POUTPU
7050       CALL PRIMA1
7060       PRINT 507
7070   507 FORMAT (//,5X,'SOLUCION OPTIMA DEL PROBLEMA',/)
7080       CALL POUTPU
7090       RETURN
7100       END
7110 %
7120 %
7130 %
7140 %
7150       SUBROUTINE FCMI66
7160       COMMON /A/ O(0:120)  /B/ T(0:120)  /G/ M,N  /H/ PT(0:120)
7170       COMMON /C/ PB(0:120)  /D/ PF(0:120)  /E/ PR(0:120)  /P/ PI(0:120)
7180      *   /I/ LT(0:120)  /J/ PO(0:120)  /K/ C(0:120)  /L/ H(0:120)
7190      *   /M/ B(0:120)  /N/ CL(0:120)  /O/ F(0:120)
7200       COMMON/R/PD(0:120)/S/PP(0:120)
7210       COMMON /SHRT/S(0:120)
7220       INTEGER PF,PB,CYC,AD
7230       INTEGER O,T,PB,PF,PR,PT,LT,PO,C,H,B,CL,F,PI
7240       INTEGER S
7250 %
```

```
C     ************************************************************
C     *                                                          *
C     *       PROGRAMAH PRINCIPAL :                              *
C     *                                                          *
C     *       **********************                             *
C     *       *     FCMI66         *                             *
C     *       **********************                             *
C     *       ASOCIADO CON EL CAPITULO :                         *
C     *       **********************                             *
C     *       *     SEIS(6)        *                             *
C     *       **********************                             *
C     *                                                          *
C     *  PROPOSITO : ENCONTRAR LA SOLUCION OPTIMA PARA UN PRO-   *
C     *  BLEMA DE FLUJO A COSTO MINIMO. PRIMERO OBTIENE UNA      *
C     *  SOLUCION INICIAL PRIMAL BASICA FACTIBLE ARTIFICIAL ,    *
C     *  USANDO EL METODO DEL CAPITULO 6, SECCION 6.3,           *
C     *  ESTA SOLUCION ES LUEGO USADA PARA EL ALGORITMO PRIMAL   *
C     *  BASICO DEL CAPITULO 6, SECCION 6.6, PARA OBTENER LA     *
C     *  SOLUCION OPTIMA.                                        *
C     *                                                          *
C     *    LAS SUBRUTINAS REQUERIDAS SON :                       *
C     *    ARTIFI,      POUTPU,      PRIMAL,     STARTH,        *
C     *    ORIGS,       TERMS,       ORIG,       TERM,          *
C     *    READ,        TREINT,      SELECT,     TPATH,         *
C     *    MFLO,        FLOCHG,      TRECHG,     ADDTRE,        *
C     *    DELTRE,      ROOT.                                   *
C     *                                                          *
C     ************************************************************
C
      PRINT 601
  601 FORMAT (' PROGRAMA FCMI66',/,
     * ' PRIMAL BASICO PARA FLUJO A COSTO MINIMO')
      CALL READ
      CALL ARTIFI
C     CALL POUTPU
      CALL PRIMAL
      PRINT 509
  509 FORMAT(//,5X,'SOLUCION OPTIMA DEL PROBLEMA',/)
      CALL POUTPU
      RETURN
      END



      SUBROUTINE READ
      COMMON/H/B(0:120)/G/M,N/B/T(0:120)/H/PT(0:120)/A/O(0:120)
     * /I/LT(0:120)/J/P0(0:120)/K/C(0:120)/L/H(0:120)
      INTEGER SALACK,T,B,O,C,H,P0,PT
      REAL LOWER
C     ************************************************************
C     *                                                          *
C     PROPOSITO : LEER Y GUARDAR LOS DATOS CORRESPONDIENTES A
C     NODOS Y ARCOS , PARA EL PROBLEMA DE REDES DE FLUJO  A
C     COSTO MINIMO.
C
C     LAS SUBRUTINAS REQUERIDAS SON LOS ALGORITMOS :
C     ORIGS Y TERMS.
C     *                                                          *
C     ************************************************************
C
      READ 41,N
   41 FORMAT (I5)
      M=0
      SLACK=N+1
      N=N+1
      DO 51 I=1,N
   51 B(I)=0
C
C  NODE
C
```

```
      21 READ  61,I,BF,BS,CS
      61 FORMAT (I5,3F10.0)
         IF(I.EQ.0) GO TO 30
         B(I)=BF
         IF ( ABS(BS) .LE. .0001) GO TO 21
         IF(BS.GT.0.) GO TO 71
         J=SLACK
         LOWER=0
         UPPER=-BS
         COST=CS
         GO TO 81
      71 J=I
         I=SLACK
         LOWER=0
         UPPER=BS
         COST=CS
      81 CALL ORIGS(I,J,LOWER,UPPER,COST)
         GO TO 21
C
C  ARCO
C
      30 READ 9,I,J,LOWER,UPPER,COST
       9 FORMAT (2I5,3F10.0)
         IF(I.EQ.0) GO TO 10
         CALL ORIGS(I,J,LOWER,UPPER,COST)
         GO TO 30
C
C  EXIT
C
      10 CONTINUE
         LM=M
         M=0
         DO 11 K=1,LM
         J=T(K)
         M=M+1
         CALL TERMS(K,J)
      11 CONTINUE
         PRINT 1, N,M
       1 FORMAT(///,10X,'REPRESENTACION DE LA RED--',/
        *  /,15X, 'NRO. DE NODOS--',I5, 10X, 'NRO. DE ARCOS--',I5 )
         PRINT 2
       2 FORMAT (//,5X,'PARAMETROS TRANSFORMADOS DE LOS NODOS',//,
        * 16X,'NODO I',5X,'B(I)',/)
         DO 5 I=1,N
         PRINT 31, I,B(I)
      31 FORMAT (10X,2I10)
       5 CONTINUE
         PRINT 4
       4 FORMAT(//,5X,'PARAMETROS TRANSFORMADOS DE LOS ARCOS',//,
        * 6X,'ARCO K',5X,'O(K)',6X,'T(K)',6X,'C(K)',6X,'H(K)',/)
         PRINT 6, (J,O(J),T(J),C(J),H(J), J=1,M)
       6 FORMAT ( 5I10)
C        PRINT 7
C     7 FORMAT(///,' INDICE I/L',5X,'PO(I)',5X,'PT(I)',5X,'LT(L)')
C        PRINT 8, (I,PO(I),PT(I),LT(I), I=1,M)
C     8 FORMAT (4I10)
         RETURN
         END
C
C
         SUBROUTINE ORIGS (I,J,LOWER,UPPER,COST)
         COMMON /J/PO(0:120)/A/O(0:120)/B/T(0:120)/K/C(0:120)/N/CL(0:120)
        */L/H(0:120)/N/B(0:120)/G/M,N
         COMMON/O/F(0:120)
         INTEGER F
         INTEGER PO,O,T,C,CL,H,B
         REAL LOWER
C
C        ************************************************************
C        *                                                          *
C        PROPOSITO : ALMACENAR EL CONJUNTO DE DATOS CORRESPONDIENTES
C        A UN ARCO , EN UNA LISTA ORDENADA INCREMENTANDO EL INDICE
C        DEL NODO ORIGEN.
C        *                                                          *
C        ************************************************************
C
```

```
      NPLUS1=N+1
%
%  INICIA
%
      IF ( M .NE. 0) GO TO 10
    1 DO 5 II = 1,NPLUS1
    5 PO(II) = 1
%
%  MUEVE
%
   10 M = M + 1
   11 IPLUS1 = I + 1
      DO 15 II = IPLUS1,NPLUS1
   15 PO(II) = PO(II) + 1
      IF ( PO(I+1) .GT. M) GO TO 25
   16 MPO1 = M - PO(I+1) + 1
      DO 20 L=1,MPO1
      K=M-L
      O(K+1)=O(K)
      T(K+1)=T(K)
      CL(K+1)=CL(K)
      C(K+1)=C(K)
      F(K+1)=F(K)
   20 H(K+1)=H(K)
%
%  ARCO
%
   25 K=PO(I+1) -1
      O(K) = I
      T(K)=J
      CL(K) = 0
      F(K)=0
      C(K) = UPPER - LOWER
      H(K) = COST
      B(I) = B(I) - LOWER
      B(J) = B(J) + LOWER
      RETURN
      END
%
%
      SUBROUTINE TERMS(K,J)
      COMMON /A/O(0:120)/H/PT(0:120)/I/LT(0:120)/G/M,N
      INTEGER O,T,PB,PF,PR,PT,LT,PO
%
%     ************************************************************
%     *                                                          *
%     PROPOSITO : ELABORA LA LISTA DE APUNTADORES (LT) DE LOS
%     INDICE DE LOS ARCOS INCREMENTANDO EL NODO TERMINAL.
%     TAMBIEN OBTIENE LOS APUNTADORES (PT) A LOS NODOS
%     TERMINALES DE LOS ARCOS EN LA LISTA (LT) , TAL QUE
%     (LT) PUEDE SER RAPIDAMENTE REFERENCIADA. ESTE ALGORITMO
%     ES LLAMADO UNA VEZ POR CADA ARCO EN LA RED.
%     *                                                          *
%     ************************************************************
%
%
      IF(M.NE.1) GO TO 25
      IJ = N+1
      DO 20 I = 1,IJ
      PT(I) = 1
   20 CONTINUE
   25 CONTINUE
%
%  MUEVE
%
      IF(J.GE.N) GO TO 50
      JI = J+1
      DO 30 JJ = JI,N
      PT(JJ) = PT(JJ)+1
   30 CONTINUE
      IF(PT(J+1).GT.M) GO TO 50
      JI = M-PT(J+1)+1
      DO 40 L = 1,JI
```

```
9470        KK = M-L
9480        LT(KK+1) = LT(KK)
9490     40 CONTINUE
9500     50 CONTINUE
9510        PT(N+1) = PT(N+1)+1
9520        KK = PT(J+1)-1
9530        LT(KK) = K
9540        RETURN
9550        END
9560 %
9570 %
9580        SUBROUTINE ORIG (I,LISA,LISN,L)
9590        COMMON /J/PO(0:120)/B/T(0:120)
9600        DIMENSION LISA(0:120),LISN(0:120)
9610       .INTEGER PO,T
9620 %
9630 %      ************************************************************
9640 %      *                                                          *
9650 %      PROPOSITO : DETERMINAR LA LISTA DE ARCOS LISA (MOI)
9660 %      ORIGINADOS EN EL NODO I , ENCUENTRA EL NODO TERMINAL
9670 %      DE CADA ARCO Y LO PONE EN LA LISTA DE NODOS LISN.
9680 %      *                                                          *
9690 %      ************************************************************
9700 %
9710 %      PRINT 50,I
9720 %   50 FORMAT (/,' SUBRUTINA ORIG CON EL NODO',I4)
9730        ISTA=PO(I)
9740        ISTO=PO(I+1)-1
9750        L=0
9760        IF ( ISTO .LT. ISTA ) GO TO 40
9770     10 DO 35 K=ISTA,ISTO
9780        L=L+1
9790        LISA(L)=K
9800     35 LISN(L)=T(K)
9810     40 CONTINUE
9820 %      PRINT 80,I,L
9830        IF ( L .EQ. 0) GO TO 41
9840 %      PRINT 60,(LISA(IW),IW=1,L)
9850 %      PRINT 70,(LISN(IW),IW=1,L)
9860 %   80 FORMAT('NRO.DE ARCCOS ORIGINADOS  EN EL  NODO',I4,' ES',I4)
9870 %   60 FORMAT ( ' ARCOS ORIGINADOS',/,(20I6))
9880 %   70 FORMAT (' NODOS TERMINALES',/,(20I6))
9890     41 RETURN
9900        END
9910 %
9920
9930        SUBROUTINE TERM(I,LISA,LISN,L)
9940        COMMON /A/O(0:120)/H/PT(0:120)/I/LT(0:120)
9950        DIMENSION LISA(0:120),LISN(0:120)
9960        INTEGER O,T,PB,PF,PR,PT,LT,PO,C,H,B,CL,F
9970 %
9980 %      ************************************************************
9990 %      *                                                          *
10000 %      PROPOSITO : DETERMINAR LA LISTA DE ARCOS LISA (MTI)
10010 %      QUE TERMINAN EN EL NODO (I) A TRAVES DE LA LISTA AUXILIAR
10020 %      (LT) DE TERMS, ADEMAS ENCUENTRA EL NODO ORIGINAL DE CADA
10030 %      ARCO Y LO PONE EN LA LISTA DE NODOS LISN.
10040 %      *                                                          *
10050 %      ************************************************************
10060 %
10070        ISTA =PT(I)
10080        ISTO=PT(I+1)-1
10090        L=0
10100        IF(ISTO-ISTA)1,2,2
10110     2  DO 3 KK=ISTA,ISTO
10120        K=LT(KK)
10130        L=L+1
10140        LISA(L)=K
10150     3  LISN(L)=O(K)
10160     1  CONTINUE
10170        RETURN
10180         END
10190 %
```

```
10200
10210       SUBROUTINE ROOT(IROOT,LISA,LISN,IC,CYC)
10220       COMMON /A/O(0:120)/B/T(0:120)/C/PB(0:120)/D/PF(0:120)/F/PR(0:120)
10230       DIMENSION LISA(0:120),LISN(0:120)
10240       INTEGER O,T
10250       INTEGER CYC,PB,PF,PR
10260 %
10270 %     ************************************************************
10280 %     *                                                          *
10290 %     PROPOSITO : ENCONTRAR LA LISTA DE ARCOS (LISA) Y LA LISTA
10300 %     DE NODOS (LINS) QUE ESTAN EN EL ARBOL DIRIGIDO ENRAIZADO
10310 %     EN EL NODO (IROOT) , SI LOS APUNTADORES DETECTAN UN CICLO
10320 %     SE HACE CYC=1.
10330 %     *                                                          *
10340 %     ************************************************************
10350 %
10360 %
10370 %     INICIO
10380 %
10390 %    PRINT 280,IROOT
10400 %280 FORMAT (/,' SUBRUTINA ROOT',/,' IROOT=',I4)
10410       II=IROOT
10420       IC=0
10430       LISN(1)=IROOT
10440       CYC=0
10450 %
10460 %     ADELANTE
10470 %
10480    90 JJ=PF(II)
10490 %     PRINT 210,II,JJ
10500 % 210 FORMAT (/,' NODO DE ADELANTE DE',I3,' ES',I3)
10510       IF(JJ.EQ.0)GO TO 10
10520 %
10530 %     LISTA ADICIONAL
10540 %
10550    20 IC=IC+1
10560       LISA(IC)=PB(JJ)
10570       LISN(IC+1)=JJ
10580       II=JJ
10590 %     PRINT 220,IC,LISA(IC),IC+1,LISN(IC+1),II
10600 % 220 FORMAT (/,'LISA(',I3,')=',I3,'LISN(',I3,')=',I3,' II=',I3)
10610 %
10620       IF(II.EQ.IROOT)GO TO 80
10630       GO TO 90
10640 %
10650    10 IF( II.EQ.IROOT)GO TO 40
10660 %
10670 %     DERECHO
10680 %
10690   170 JJ=PR(II)
10700 %     PRINT 230, II,JJ
10710 % 230 FORMAT (/,' APUNTADOR DERECHO DEL NODO ',I3,' IS',I3)
10720       IF(JJ.EQ. 0 ) GO TO 110
10730       GO TO 20
10740 %
10750 %  ATRAS
10760 %
10770   110 K=PB(II)
10780 %     PRINT 240,II,K
10790 % 240 FORMAT (/,' ARCO APUNTADOR HACIA ATRAS DE ',I3,' IS ',I3)
10800       IF(K.GT.0)GO TO 150
10810 %
10820       II=T(-K)
10830 %
10840       GO TO 160
10850   150 II=O(K)
10860   160 CONTINUE
10870 %     PRINT 250,II
10880 % 250 FORMAT (' REGRESO PARA EL NODO ',I3)
10890       IF(II.EQ.IROOT)GO TO 40
10900       GO TO 170
10910    80 CYC=1
10920 %     PRINT 260
10930 % 260 FORMAT (' CYC=1,SE PRESENTA UN CICLO')
10940 %
```

```
10950     40 CONTINUE
10960 %       IF ( IC .EQ. 0) GO TO 43
10970 %       PRINT 41,(LISA(I),I=1,IC)
10980 %    43 PRINT 42,(LISN(I),I=1,IC+1)
10990 %    41 FORMAT (' LISA,LISN',/,(2514))
11000 %    42 FORMAT (2514)
11010 %       PRINT 270
11020 % 270 FORMAT (' FIN DE ROOT',/ )
11030         RETURN
11040         END
11050
11060
11070       SUBROUTINE DELTRE (KL)
11080       COMMON/A/O(0:120)/B/T(0:120)/C/PB(0:120)/D/PF(0:120)/F/PR(0:120)
11090      */G/M,N
11100         INTEGER O,T,PB,PF,PR
11110 %       ************************************************************
11120 %       *                                                          *
11130 %       PROPOSITO : QUITAR UN ARCO DEL ARBOL Y ACTUALIZAR
11140 %       SU REPRESENTACION EN TERMINOS DE LOS APUNTADORES DE LAS
11150 %       TRES ETIQUETAS.
11160 %       *                                                          *
11170 %       ************************************************************
11180 %
11190 %       PRINT 250
11200 % 250 FORMAT (/,' SUBRUTINA DELTRE ')
11210 %       PRINT 260,KL
11220 % 260 FORMAT ('  EL ARCO QUE SALE ES ',I3)
11230         IF( KL.LE.0 )GO TO 20
11240 %
11250 %       ADELANTE
11260 %
11270      10 IL=O(KL)
11280         JL=T(KL)
11290         GO TO 30
11300      20 IL=T(-KL)
11310         JL=O(-KL)
11320      30 CONTINUE
11330 %       PRINT 240,IL,JL,PF(IL)
11340 % 240 FORMAT (' IL=',I3,' JL=',I3,' PF(IL)=',I3)
11350         IF (PF(IL).NE.JL) GO TO 40
11360         PF(IL)=PR(JL)
11370 %       PRINT 230,IL,PF(IL)
11380 % 230 FORMAT (' RESULTADOS PF(',I3')=',I3)
11390         GO TO 80
11400      40 L=PF(IL)
11410 %
11420 %       DERECHO
11430 %
11440      50 CONTINUE
11450 %       PRINT 220,L,PR(L)
11460 % 220 FORMAT (' L=',I3,' PR(L)=',I3)
11470         IF(PR(L).NE.JL) GO TO 90
11480         PR(L)=PR(JL)
11490 %       PRINT 270,L,PR(L)
11500 % 270 FORMAT (' RESULTADOS PR(',I3,')=',I3)
11510 %
11520 %    SALIDA
11530 %
11540      80 PB(JL)=0
11550         PR(JL)=0
11560 %
11570 %    VER INST. 6000-6200 EN LISTADO ORIGINAL
11580 %
11590 %       PRINT 900,(I,PF(I),PB(I),PR(I),I=1,N)
11600 % 900 FORMAT (' RESULTADOS',/,'     I PF(I) PB(I) PR(I)',/,(4I5))
11610 %       PRINT 200
11620 % 200 FORMAT (' FIN DE DELTRE ')
11630         GO TO 60
11640      90 L=PR(L)
11650         GO TO 50
11660      60 RETURN
11670         END
11680
```

```
11690
11700       SUBROUTINE ADDTRE(KE)
11710       COMMON /A/O(0:120)/B/T(0:120)/C/PB(0:120)/D/PF(0:120)/F/PR(0:120)
11720       COMMON /G/PD(0:120)/G/M,N
11730       DIMENSION LISA(0:120),LISN(0:120)
11740       INTEGER O,T,PB,PF,PR,PD,PDADJ
11750 %
11760 %     ************************************************************
11770 %     *                                                          *
11780 %     PROPOSITO : INSERTAR EL ARCO KE(IE,JE) A UN BOSQUE.
11790 %     LOS NODOS (IE) Y (JE) DEBERAN ESTAR EN DIFERENTES ARBOLES
11800 %     Y EL NODO (JE) DEBERA SER RAIZ DE UN ARBOL.
11810 %
11820 %     LA SUBRUTINA REQUERIDA ES EL ALGORITMO : ROOT
11830 %     *                                                          *
11840 %     ************************************************************
11850 %
11860 %     PRINT 270
11870 % 270 FORMAT (/,' SUBRUTINA ADDTRE ')
11880 %
11890 %     ADELANTE
11900 %
11910 %     PRINT 200,KE
11920 % 200 FORMAT (' EL ARCO ENTRANTE ES ',I3)
11930       IF(KE) 15,15,20
11940    15 IE=T(-KE)
11950       JE=O(-KE)
11960       GO TO 30
11970    20 IE=O(KE)
11980       JE=T(KE)
11990    30 CONTINUE
12000 %     PRINT 210,IE,JE,IE,PF(IE)
12010 % 210 FORMAT (/,' NODO ORIGEN ES',I3,//,' NODO TERMINAL ES',I3,/,
12020 %    *' PF(',I3,')=',I3)
12030       IF(PF(IE).EQ.0) GO TO 40
12040       PR(JE) = PF(IE)
12050 %     PRINT 220,JE,PR(JE)
12060 % 220 FORMAT (' RESULTADO PR(',I3,')=',I3)
12070 %
12080 %     ATRAS
12090 %
12100    40 PF(IE) = JE
12110       PB(JE) = KE
12120 %     PRINT 230,IE,JE,JE,KE
12130 % 230 FORMAT ('APTDOR. PF(',I3,')=',I3,/,'APTDOR. PB(',I3,')=',I3)
12140 %
12150 %     PROFUNDIDAD
12160 %
12170       PDADJ = PD(IE) - PD(JE) + 1
12180 %     PRINT 240,IE,PD(IE),JE,PD(JE),PDADJ
12190 % 240 FORMAT (' PD(',I3,')=',I3,' PD(',I3')=',I3,/,
12200 %    * '        ' YIELDS PDADJ= ',I3)
12210       CALL ROOT (JE,LISA,LISN,IC,CYC)
12220       DO 50 I = 1,IC+1
12230       PD(LISN(I)) = PD(LISN(I)) + PDADJ
12240 % 250 FORMAT (' RESULTADO PD(',I3')=',I3)
12250 %     PRINT 250,LISN(I),PD(LISN(I))
12260    50 CONTINUE
12270 %     PRINT 260,(I,PF(I),PB(I),PR(I),PD(I),I=1,N)
12280 % 260 FORMAT (/,' RESULTADOS',//,4X,'I',3X,'PF',3X,'PB',3X,'PR',
12290 %    *3X,'PD',//,(5I5))
12300 %     PRINT 280
12310 % 280 FORMAT (' FIN DE ADDTRE ')
12320       RETURN
12330       END
12340 %
12350 %
12360       SUBROUTINE TRECHG(KL,KE)
12370       COMMON/A/O(0:120)/B/T(0:120)/C/PB(0:120)/D/PD(0:120)
12380       COMMON/G/M,N
12390       DIMENSION LISA(0:120),LISN(0:120),LISND(0:120)
12400       INTEGER O,T,PD,PB
```

```
12410 %
12420 %     ************************************************************
12430 %     *                                                          *
12440 %     PROPOSITO : QUITAR UN ARCO (KL) DEL ARBOL BASE E INSERTAR
12450 %     OTRO ARCO (KE) , ADEMAS DIRIGIR CIERTOS ARCOS EN EL ARBOL
12460 %     PARA MANTENERLO DIRIGIDO.
12470 %
12480 %     LAS SUBRUTINAS REQUERIDAS SON LOS ALGORITMOS :
12490 %     DELTRE Y ADDTRE.
12500 %     *                                                          *
12510 %     ************************************************************
12520 %
12530 %     PRINT 500
12540 % 500 FORMAT(/,'SUBRUTINA TRECHG')
12550 %     PRINT 300,KL,KE
12560 % 300 FORMAT ('EL ARCO QUE SALE ES',I4,'EL ARCO QUE ENTRA ES',I4)
12570 %  DELETE
12580       CALL DELTRE(KL)
12590 %  FIND
12600       IF (KL .GT. 0) GO TO 10
12610       JL = O(-KL)
12620       GO TO 20
12630 %
12640    10 JL = T(KL)
12650 %
12660    20 IF (KE .GT. 0) GO TO 30
12670       JE = O(-KE)
12680       GO TO 40
12690 %
12700    30 JE = T(KE)
12710 %
12720 %     ORTENCION DE LA RUTA INVERSA
12730    40 IC = 1
12740 %     PRINT 310,JL,JE
12750 % 310 FORMAT (' EL NODO TERMINAL DEL ARCO SALIENTE ES',I4,//,
12760 %    *'EL NODO TERMINAL DEL ARCO ENTRANTE ES',I4)
12770       LISA(1) = -KE
12780       LISN(1) = JE
12790       IF ( JE .EQ. JL) GO TO 55
12800    60 IC = IC +1
12810       LISA(IC) = PR(LISN(IC-1))
12820       IF(LISA(IC) .GT. 0 ) GO TO 70
12830       LISN(IC) = T(-LISA(IC))
12840       GO TO 80
12850    70 LISN(IC) = O(LISA(IC))
12860    80 IF (LISN(IC) .NE. JL) GO TO 60
12870 %   INVERSA
12880 %     PRINT 320,(LISA(I),I=1,IC)
12890 %     PRINT 330,(LISN(I),I=1,IC)
12900 % 320 FORMAT (' ARCOS EN AL RUTA REVERSA',/,(20I4))
12910 % 330 FORMAT (' NODOS EN LA RUTA REVERSA',/,(20I4))
12920       DO 1000 I=2,IC
12930       CALL DELTRE (LISA(I))
12940       CALL ADDTRE (-LISA(I-1))
12950  1000 CONTINUE
12960    55 CALL ADDTRE (-LISA(IC))
12970 %     PRINT 400
12980 % 400 FORMAT (' FIN DE TRECHG',/)
12990       RETURN
13000       END
13010 %
13020
13030       SUBROUTINE TREINT(N)
13040       COMMON /A/O(0:120)/B/T(0:120)/C/PR(0:120)/D/PF(0:120)/F/PR(0:120)
13050       COMMON /G/PR(0:120)/R/PP(0:120)
13060       DIMENSION LISA(0:120),LISN(0:120)
13070       INTEGER PR,PD,O,T,PP
13080       INTEGER PF,PR
13090 %.
```

```
C     ************************************************************
C     *                                                          *
C     PROPOSITO : ESPECIFICAR LOS APUNTADORES DE UN ARBOL
C     TENIENDO CONOCIMIENTO DEL APUNTADOR HACIA ATRAS.
C
C     LA SUBRUTINA REQUERIDA ES EL ALGRITMO : ADDTRE.
C     *                                                          *
C     ************************************************************
C
      PRINT 200
  200 FORMAT (' SUBRUTINA TREINT')
C
C  OBTENCION DE PP( O PF Y PR ) E IROOT
C
C     PRINT 90,(PB(I),I=1,N)
C  90 FORMAT (' APUNTADORES (PB) EN EL ARBOL',/,(25I4))
      DO 80 I = 1,N
      PP(I) = I
      PF(I) = 0
      PR(I) = 0
      PB(I) = 0
   80 CONTINUE
C     PRINT 100,(PP(I),I=1,N)
C 100 FORMAT (' LISTA PP INICIAL REQUERIDA',//,(25I4))
      DO 10 I =1,N
      IF (PB(I) .EQ. 0) GO TO 10
      CALL ADDTRE(PB(I))
   10 CONTINUE
C
C     PRINT 300
C 300 FORMAT (' FIN DE TREINT',/)
      RETURN
      END
C
C
      SUBROUTINE FLOCHG(LISA,IC,MF)
      COMMON /D/F(0:120)/B/M,N
      DIMENSION LISA(0:120)
      INTEGER F
C
C     ************************************************************
C     *                                                          *
C     CAMBIAR LOS FLUJOS EN UNA TRAYECTORIA (LISA), PARA CADA ARCO
C     EN LA TRAYECTORIA; SI K>0, INCREMENTA EL FLUJO EN EL ARCO K
C     POR MF, SI K<0, DISMINUYE EL FLUJO EN EL ARCO -K EN MF.
C     *                                                          *
C     ************************************************************
C
      DO 100 L=1,IC
      K=LISA(L)
C     PRINT 130,K
C130  L= FORMAT (' K=',I3)
      IF ( K .GT. 0 ) GO TO 98
      KW=-K
C     PRINT 120,KW,F(KW),MF
      F(-K)=F(-K)-MF
C     PRINT 110,KW,F(KW)
      GO TO 99
   98 CONTINUE
C     PRINT 120,K,F(K),MF
C120 FORMAT (' F(',I3,')=',I5,' Y MF=',I5,' YIELDS')
      F(K)=F(K)+MF
C     PRINT 110,K,F(K)
C110 FORMAT (' F(',I3,')=',I5)
   99 CONTINUE
  100 CONTINUE
C     PRINT 140,(F(I),I=1,M)
C140  FORMAT (' F',/,(10I5))
      RETURN
      END
C
C
      SUBROUTINE NFLO(LISA,IC,MF,KL,ILC)
      COMMON /D/F(0:120)/K/C(0:120)
      DIMENSION LISA(0:120)
      INTEGER F,C
```

```
13860 %
13870 %     ************************************************************
13880 %     * PROPOSITO :                                              *
13890 %     DETERMINAR EL MAXIMO CAMBIO DE FLUJO (MF) EN LA TRAYECTORIA
13900 %     (LISA), EL ALGORITMO TAMBIEN DETERMINA EL ARCO (KL) PARA EL
13910 %     CUAL SE SATURA SU CAPACIDAD, GUARDANDO ESTA INFORMACION EN
13920 %     LA VARIABLE (ILC).
13930 %     *                                                          *
13940 %     ************************************************************
13950 %
13960       MF=9999
13970       KL=0
13980       ILC=0
13990       DO 100 L=1,IC
14000       K=LISA(L)
14010 %     PRINT 140,K
14020 % 140 FORMAT (' K=',I3)
14030       IF ( K .GT. 0 ) GO TO 97
14040 %     PRINT 130,MF,K,F(-K)
14050 % 130 FORMAT (' MF=',I5,' F(',I3,')=',I5)
14060       IF ( MF .GT.F(-K) ) GO TO 96
14070       GO TO 99
14080    96 MF=F(-K)
14090       KL=K
14100       ILC=L
14110 %     PRINT 110,MF,KL,ILC
14120       GO TO 99
14130    97 CONTINUE
14140       KCMF=C(K) - F(K)
14150 %     PRINT 120,MF,KCMF
14160 % 120 FORMAT (' MF=',I5,' KCMF=',I5)
14170       IF ( MF .GT. (C(K)-F(K)) ) GO TO 98
14180       GO TO 99
14190    98 MF=C(K)-F(K)
14200       KL=K
14210       ILC=L
14220 %     PRINT 110,MF,KL,ILC
14230 % 110 FORMAT (' MF=',I5,' KL=',I3,' ILC=',I3)
14240    99 CONTINUE
14250   100 CONTINUE
14260       RETURN
14270       END
14280
14290
14300       INTEGER FUNCTION AD(K)
14310       COMMON /K/C(0:120) /N/CL(0:120) /O/F(0:120)
14320       INTEGER C,CL,F
14330 %
14340       AD=0
14350       IF (K) 20,30,10
14360    10 IF (F(K).LT.C(K)) AD=1
14370       RETURN
14380    20 IF (F(-K).GT.CL(-K)) AD=1
14390       RETURN
14400    30 STOP
14410       END
14420
14430
14440       SUBROUTINE MAXFLO(SN,TN,VR,V,INF)
14450       COMMON/A/O(0:120)/B/T(0:120)/C/PB(0:120)/F/PR(0:120)/G/M,N/
14460      *H/PT(0:120)/I/LT(0:120)/J/PO(0:120)/K/C(0:120)/L/H(0:120)
14470      */M/B(0:120)/N/CL(0:120)/O/F(0:120)/D/PF(0:120)/P/PI(0:120)
14480        DIMENSION LISA(0:120),LISN(0:120)
14490       INTEGER O,T,PB, PF,PR,PT,PO,C,H,B,CL,F,PI ,SN,TN     ,DEL,V,VR
14500       INTEGER DWES
14510 %
14520 %     ************************************************************
14530 %     *                                                          *
14540 %     PROPOSITO : ENCONTRAR EL FLUJO QUE PUEDE PASAR EN LA RED,
14550 %     EL CUAL SE OBTIENE DANDO INICIALMENTE UN FLUJO (VR) DESDE EL
14560 %     EL NODO FUENTE (SN) HASTA EL NODO SUMIDERO (TN). SI EL FLU-
14570 %     JO DADO NO PUEDE SER OBTENIDO, EL MAXIMO FLUJO DEL NODO
14580 %     FUENTE AL NODO SUMIDERO ES OBTENIDO. SE DEBE DAR UN FLUJO
14590 %     INICIAL FACTIBLE.
14600 %     *                                                          *
14610 %     ************************************************************
```

```
14620 %.
14630 %
14640 %  INICIA
14650 %
14660 %      PRINT 110,SN,TN,VR
14670 % 110 FORMAT (/,' SUBRUTINA MAXFLO--SN=',I3,',TN=',I3,',VR=',I3)
14680        V=0
14690        IFIN=0
14700        INF=0
14710        IF=1
14720 %
14730 %  RUTA
14740 %
14750      7  CALL FPATH(IF,SN,TN,NP)
14760        IF ( NP .EQ. 0) GO TO 1
14770        INF = 1
14780 %      PRINT 100,NP,INF
14790 % 100 FORMAT (' NP=',I3,' ,INF=',I3,' NO ADICIONA '
14800 %     *   ,'EXISTE RUTA')
14810         GO TO 3
14820      1 IC = 0
14830        IJ = TN
14840 %
14850 %  LSTARC
14860 %
14870      9 K=PB(IJ)
14880        IC = IC + 1
14890        LISA (IC) = K
14900        IF (K .LT. 0) GO TO 6
14910        IJ = O(K)
14920     10 IF ( IJ .EQ. SN ) GO TO 8
14930        GO TO 9
14940      6 IJ = T(-K)
14950        GO TO 10
14960      8 CONTINUE
14970 %      PRINT 160,(LISA(IW),IW=1,IC)
14980 % 160 FORMAT (' ARCOS EN LA RUTA',/,(10I5))
14990 %
15000 %  FLUJO
15010 %
15020          CALL MFLO(LISA,IC,DEL,KL,ILC)
15030        DWES = DEL + V - VR
15040 %      PRINT 120,DEL,V,VR,DWES
15050 % 120 FORMAT (' DEL=',I4,', V=',I3,',VR=',I3,',DE+V-VR=',I4)
15060        IF ( DEL + V .LT. VR) GO TO 4
15070 %      PRINT 130,VR
15080 % 130 FORMAT (' FLUJO DE  ',I3,' UNIDADES SE HA OBTENIDO')
15090         DEL=VR-V
15100         IFIN =1
15110      4 CALL FLOCHG(LISA,IC,DEL)
15120 %      PRINT 180 ,(IW,O(IW),T(IW),F(IW),C(IW),IW=1,M)
15130    180 FORMAT('     ARC ORIGEN TERMINAL FLUJ.CAPACID.',(I5,I7,I8,I5,I7))
15140          V=V+DEL
15150 %      PRINT 140,DEL,V
15160    140 FORMAT (' FLUJO TOTAL INCREMEN.POR',I3,' UNID.AL NUE.VALOR DE',I3)
15170        IF ( IFIN .EQ. 0) GO TO 7
15180      3 CONTINUE
15190 %      PRINT 150
15200 % 150 FORMAT (' FIN DE MAXFLO ',/)
15210        RETURN
15220        END
15230
15240
15250        SUBROUTINE LABEL1 (SN,TN,NP)
15260        DIMENSION LISA(0:120),LISN(0:120),S(0:120)
15270        COMMON /C/PB(0:120) /G/M,N
15280        INTEGER SN,TN,S,PB,AD
15290 %
15300 %      ************************************************************
15310 %      * PROPOSITO :                                              *
15320 %      ENCONTRAR UNA RUTA DESDE EL NODO (SN) AL NODO (TN), USANDO
15330 %      SOLAMENTE ARCOS ADMISIBLES HACIA ADELANTE O HACIA ATRAS.
15340 %      *                                                          *
15350 %      ************************************************************
15360 %
```

```
15370 %     PRINT 100,SN,TN
15380 % 100 FORMAT (/,' SUBRUTINA LABEL1--SN=',I3,' TN=',I3)
15390 %
15400 %  INICIA
15410 %
15420       NP=0
15430       DO 10 I=1,N
15440       S(I)=0
15450  10   PB(I)=0
15460       S(SN)=1
15470       I=SN
15480       IT=1
15490       IC=1
15500 %
15510 %  ADELANTE
15520 %
15530 %     PRINT 190,(S(IW),IW=1,N)
15540 % 190 FORMAT (' S',/,(10I3))
15550  20   CALL ORIG (I,LISA,LISN,L)
15560 %     PRINT 120,I
15570 %120 FORMAT (1X,'BUSCANDO EL NODO',I3)
15580       IF (L.LE.0) GO TO 40
15590       DO 30 LK=1,L
15600       K=LISA(LK)
15610       J = LISN(LK)
15620 %     PRINT 180,K,J,AD(K),S(J)
15630 % 180 FORMAT(' K=',I3,' J=',I3,' AD(K)=',I2,' S(J)=',I2)
15640       IF (AD(K).NE.1.OR.S(J).NE.0) GO TO 30
15650       IT=IT+1
15660       S(J)=IT
15670       PB(J)=K
15680 %     PRINT 110,PB(J),J,IT
15690 %     PRINT 200,TN
15700 % 200 FORMAT (' TN=',I3)
15710       IF (TN .LE. 0) GO TO 30
15720       IF (S(TN) .GT. 0) GO TO 80
15730    30 CONTINUE
15740 %
15750 %  REFLEJADO
15760 %
15770  40 CALL TERM (I,LISA,LISN,L)
15780       IF (L.LE.0) GO TO 60
15790       DO 50 LK=1,L
15800       K=LISA(LK)
15810       J=LISN(LK)
15820 %     PRINT 170,K,J,AD(-K),S(J)
15830 % 170 FORMAT (' K=',I3,' J=',I3,' AD(-K)=',I2,' S(J)=',I2)
15840       IF (AD(-K).NE.1.OR.S(J).NE.0) GO TO 50
15850       IT=IT+1
15860       S(J)=IT
15870       PB(J)=-K
15880 %     PRINT 110,PB(J),J,IT
15890 %110 FORMAT (2X,'ARCO ADI.',I3,1X,'NODO ETIQUE.',I3,' ITERA.',I3)
15900 %     PRINT 200,TN
15910       IF ( TN .LE. 0) GO TO 50
15920       IF (S(TN) .GT. 0) GO TO 80
15930    50 CONTINUE
15940 %
15950 %  SELECCIONA
15960 %
15970    60 IC=IC+1
15980 %     PRINT 160,IC
15990 % 160 FORMAT (' IC=',I3)
16000       IF(IC.GT.N)GO TO 80
16010       DO 70 I=1,N
16020 %     PRINT 150,I,S(I)
16030 % 150 FORMAT (' I=',I3,' S(I)=',I3)
16040       IF (S(I).EQ.IC) GO TO 20
16050    70 CONTINUE
16060 %     PRINT 140
16070 % 140 FORMAT (' NP=1')
16080       NP=1
16090    80 CONTINUE
16100 %     PRINT 130
16110 % 130 FORMAT (' FIN DE LABEL1 ',/)
16120       RETURN
16130       END
```

```
16140
16150
16160       SUBROUTINE FPATH (IF,SN,TN,NP)
16170       COMMON /C/PB(0:120) /G/M,N
16180       INTEGER PB,SN,TN
16190 %
16200 %     ************************************************************
16210 %     *                                                          *
16220 %     PROPOSITO : PROCEDE A ENCONTRAR LA RUTA PARA EL ALGORITMO DE
16230 %     RUTA MAS CORTA CUANDO SE REQUIERE UNA SOLUCION BASICA.
16240 %     *                                                          *
16250 %     ************************************************************
16260 %
16270 %     PRINT 100,IF,SN,TN
16280 % 100 FORMAT (' SUBRUTINA FPATH2--IF=',I2,',SN=',I3,',TN=',I3)
16290       IF(IF.NE.1)GO TO 10
16300       CALL LABEL1 (SN,0,NP)
16310       IF(NP.EQ.1) GO TO 30
16320       CALL TREINT(N)
16330  30   IF=0
16340 %     IF ( NP .EQ. 1) PRINT 110
16350 % 110 FORMAT(' NO EXISTE ARBOL')
16360 %     PRINT 120
16370 % 120 FORMAT (' END FPATH2')
16380 %
16390 %
16400       RETURN
16410  10   CALL LABEL2 (SN,INF)
16420       IF ( INF .EQ. 1 ) NP=1
16430 %     PRINT 120
16440       RETURN
16450       END
16460
16470
16480       SUBROUTINE LABEL2 (SN,INF)
16490       COMMON /A/D(0:120) /B/T(0:120) /G/M,N
16500       COMMON /C/PB(0:120)
16510       DIMENSION LISN(0:120),LISA(0:120),S(200)
16520       INTEGER AD,D,S,SN,T,CYC
16530 %
16540 %     ************************************************************
16550 %     *                                                          *
16560 %     PROPOSITO : PROPORCIONAR UN NUEVO ARBOL GENERADOR DESPUES DE
16570 %     QUE UNO O MAS ARCOS EN EL ARBOL ORIGINAL SE HACEN INADMISI-
16580 %     BLES.
16590 %     *                                                          *
16600 %     ************************************************************
16610 %
16620 %     PRINT 310
16630 % 310 FORMAT (' SUBRUTINA LABEL2 ')
16640 %
16650 %  INICIA
16660 %
16670       INF=0
16680       JJ=SN
16690  15   CONTINUE
16700 %
16710 %  SALE
16720 %
16730       DO 10 I=1,N
16740       S(I)=0
16750  10   CONTINUE
16760       CALL ROOT (JJ,LISA,LISN,IC,CYC)
16770       IF (IC.LT.1) GO TO 1
16780       DO 20 L=1,IC
16790       K=LISA(L)
16800 %     PRINT 300,K,AD(K)
16810 % 300 FORMAT (' K=',I3,' AD(K)=',I3)
16820       IF(AD(K).NE.0) GO TO 20
16830       KL=K
16840       L1=L+1
16850       J=LISN(L1)
16860 %     PRINT 290,KL,L1,J
16870 % 290 FORMAT (' KL=',I3,' L1=',I3,' J=',I3)
16880       GO TO 40
16890  20   CONTINUE
16900       GO TO 1
```

```
16910 %
16920 %  ARBOL
16930 %
16940      40 CALL ROOT (J,LISA,LISN,IC,CYC)
16950         DO 42 L = 1,IC+1
16960      42 S(LISN(L)) = 1
16970 %       PRINT 260,(S(I),I=1,N)
16980 % 260 FORMAT (' S',/,(10I5))
16990 %
17000 %  ENTRA
17010 %
17020   50    KE=0
17030         DO 54 K=1,M
17040         I=O(K)
17050         J=T(K)
17060 %       PRINT 230,K,I,J,S(I),S(J)
17070 % 230 FORMAT(' K=',I3,',I=',I3,',J=',I3,',S(I)=',I3)
17080         IF (AD(K).EQ.1.AND.S(I).EQ.0.AND.S(J).EQ.1) KE=K
17090         IF (AD(-K).EQ.1.AND.S(I).EQ.1.AND.S(J).EQ.0) KE=-K
17100 %       PRINT 220,KE
17110 % 220 FORMAT (' KE=',I3)
17120         IF (KE.NE.0) GO TO 60
17130   54    CONTINUE
17140         INF=1
17150      1  CONTINUE
17160 %       PRINT 210
17170 % 210 FORMAT (' FIN DE LABEL2')
17180         RETURN
17190   60    CONTINUE
17200 %
17210 %  PIVOTE
17220 %
17230 %       PRINT 130,KL,KE
17240         CALL TRECHG (KL,KE)
17250         IF(KE) 61,95,62
17260   61    JJ=O(-KE)
17270 %       PRINT 200,JJ
17280         GO TO 15
17290   62    JJ=T(KE)
17300 %       PRINT 200,JJ
17310    200 FORMAT (' JJ=',I3)
17320         GO TO 15
17330    95 CONTINUE
17340 %       PRINT 110
17350 % 110 FORMAT (/,1X,'LABEL2 ERROR-- INDICE CERO')
17360 % 130 FORMAT (/,2X,'ARCO SALIENTE',I4,'ARCO ENTRANTE',I4)
17370         STOP
17380         END
17390
17400
17410         SUBROUTINE POUTPU
17420         COMMON/G/N,M,P/PI(0:120)/K/C(0:120)/L/H(0:120)/N/CL(0:120)
17430        */O/F(0:120)
17440         COMMON /C/ PB(0:120) /A/ O(0:120) /B/ T(0:120)
17450         INTEGER PI,C,H,CL,F
17460         INTEGER O,T,PB
17470 %
17480 %       ************************************************************
17490 %       *                                                          *
17500 %       PROPOSITO : IMPRIME UNA INFORMACION DETALLADA DE LA RED.LOS
17510 %       VALORES DE LOS POTENCIALES EN LOS NODOS,LOS APUNTADORES
17520 %       PB(K), EL FLUJO EN LOS ARCOS, EL NODO ORIGEN Y EL NODO
17530 %       TERMINAL DE CADA ARCO, LOS COSTOS POR UNIDAD DE FLUJO PARA
17540 %       CADA ARCO, Y EL COSTO TOTAL DE TODOS LOS FLUJOS EN LA RED.
17550 %       ************************************************************
17560         PRINT 101
17570    101 FORMAT (//,4X,'NODO I',7X,'P(I)',5X,'PB(I)',/)
17580         DO 105 I=1,N
17590         PRINT 102, I,PI(I),PB(I)
17600    102 FORMAT (3I10)
17610    105 CONTINUE
17620         PRINT 103
17630    103 FORMAT (///,6X,'ARC K',6X,'O(K)',6X,'T(K)',6X,'FLUJO',2X,
17640        *   'CAPACIDAD',2X,'COST/UNI.',5X,' COST.FLUJO',/)
17650         ICOST=0
```

```
      DO 110 K=1,M
      KCOST=F(K)*H(K)
      PRINT 106, K,O(K),T(K),F(K),C(K),H(K),KCOST
  106 FORMAT (6I10,5X,I10)
      ICOST=ICOST+KCOST
  110 CONTINUE
      PRINT 107, ICOST
  107 FORMAT (//,30X,'COSTO TOTAL =',I10)
      RETURN
      END


      SUBROUTINE DUALSP (SN,INF)
      COMMON/A/O(0:120)/B/T(0:120)/G/N,N/L/H(0:120)/P/PI(0:120)
     *   /C/PR(0:120)/D/PF(0:120)/F/PR(0:120)
      INTEGER O,T,H,S,SN,D,DEL,PI,PF,PR,PD,CYC
      INTEGER ADC
      DIMENSION S(0:120),LISA(0:120),LISN(0:120)
%
%     ************************************************************
%     *                                                          *
%     PROPOSITO : OBTENER UN NUEVO ARBOL DE TRAYECTORIAS MAS
%     CORTAS CUANDO UNO O MAS ARCOS DEL ARBOL DE TRAYECTORIAS MAS
%     CORTAS SE HACEN INADMISIBLES. EL ALGORITMO COMIENZA CON UN
%     ARBOL DE TRAYECTORIA MAS CORTA ENRAIZADO EN EL NODO FUENTE
%     Y CON NODOS POTENCIALES QUE SATISFACEN LA FACTIBILIDAD DEL
%     PROBLEMA DUAL Y LAS CONDICIONES DE HOLGURA COMPLEMENTARIA.
%     *                                                          *
%     ************************************************************
%
%     PRINT 210
% 210 FORMAT (' SUBRUTINA DUALSP')
%
%  INICIA
%
   27 DO 1 I=1,N
    1 S(I)=0
      CALL ROOT(SN,LISA,LISN,IC,CYC)
%
%  SALE
%
      IF(IC.GT.1) GO TO 6
    7 INF =0
%     PRINT 100
% 100 FORMAT(' INF=0, UN NUEVO OPTIMO EXISTE',/,' FIN DE DUALSP')
      RETURN
    6 DO 2 L=1,IC
      K=LISA(L)
%     IF (ADC(K) .NE. 0) PRINT 110,K
%     IF (ADC(K) .EQ. 0) PRINT 120,K
% 110 FORMAT ('  ARC',I3,' ES ADMISIBLE')
% 120 FORMAT (' ARC',I3,' NO ES ADMISIBLE')
      IF ( ADC(K) .NE. 0) GO TO 2
      KL=K
      J=LISN(L+1)
%     PRINT 160,KL,J
% 160 FORMAT (' KL=',I3,' ,J=',I3)
      GO TO 26
    2 CONTINUE
      GO TO 7
%
%  ARBOL
%
   26 CALL ROOT (J,LISA,LISN,IC,CYC)
      DO 3 L=1,IC+1
    3 S(LISN(L)) = 1
%     PRINT 130,(S(I),I=1,N)
% 130 FORMAT (' S',/,(10I5))
%
%  ENTRA
%
      DEL=9999
      KE=0
      DO 4 K=1,M
      I=O(K)
```

```
      J=T(K)
%     PRINT 140,K,I,J,ADC(K),S(I),S(J)
% 140 FORMAT ('     K     I     J  ADC(K)   S(I)   S(J)',/,6I6)
      IF ( ADC(K) .NE. 1) GO TO 4
      IF ( S(I) .NE. 0) GO TO 4
      IF ( S(J) .NE. 1 ) GO TO 4
      D=PI(I)+H(K)-PI(J)
%     PRINT 150,PI(I),H(K),PI(J),D,DEL
% 150 FORMAT ('  PI(I)  H(K) PI(J)    D   DEL',/,5I6)
      IF ( D .GE. DEL ) GO TO 4
      DEL=D
      KE=K
%     PRINT 170,DEL,KE
% 170 FORMAT (' DEL=',I3,' ,KE=',I3)
    4 CONTINUE
      IF(KE.EQ.0) GO TO 20
%
%  PIVOTE
%
      CALL TRECHG(KL,KE)
      JJ = T(KE)
      CALL ROOT(JJ,LISA,LISN,IC,CYC)
      DO 5 L=1,IC+1
      I=LISN(L)
    5 PI(I)=PI(I)+DEL
%     PRINT 180,(PI(I),I=1,N)
% 180 FORMAT (' PI',/,(10I6))
%     PRINT 190,(PB(I),I=1,N)
% 190 FORMAT (' PB',/,(10I6))
      GO TO 27
   20 INF=1
%     PRINT 200
%  200 FORMAT(' NO ESISTE SOLUCION FACTIBLE',/,' FIN DE DUALSP')
      RETURN
      END


      SUBROUTINE DSHORT(SN,TN,NP)
      COMMON /C/PB(0:120)/P/PI(0:120)/L/H(0:120)/G/M,N
      INTEGER SN,PI,PB,H,TN,AD,S,D
      DIMENSION LISA(0:120),LISN(0:120),S(0:120)
%
%     ************************************************************
%     *                                                          *
%     PROPOSITO : HALLAR LA TRAYECTORIA MAS CORTA DEL NODO (S) AL
%     NODO (T), CUANDO EL COSTO DE TODOS LOS ARCOS ES POSITIVO.
%     SI T=0 , EL ALGORITMO DA EL ARBOL DE TRAYECTORIA MAS CORTA
%     *                                                          *
%     ************************************************************
%
%
%  INICIA
%
      NP = 0
%     PRINT 100,SN,TN,NP
% 100 FORMAT (' SUBRUTINA DSHORT-- SN=',I4,',TN=',I4,',NP=',I3)
      DO 1 I=1,N
      PI(I)=9999
      PB(I)=0
    1 S(I)=0
      PI(SN)=0
      I=SN
      S(SN)=1
%
%  ADELANTE
%
%     PRINT 110,(PI(IW),IW=1,N)
%     PRINT 120,(PB(IW),IW=1,N)
%     PRINT 130,(S(IW),IW=1,N)
% 110 FORMAT (' PI',/,(20I6))
% 120 FORMAT (' PB',/,(20I6))
% 130 FORMAT (' S',/,(20I6))
    2 CALL ORIG(I,LISA,LISN,IC)
```

```
      IF (IC .LE.0) GO TO 4
      DO 6 L = 1,IC
      K=LISA(L)
      J=LISN(L)
%     PRINT 140,J,S(J)
% 140 FORMAT (' S(',I3,')=',I2)
      IF(S(J) .NE. 0) GO TO 6
      D=PI(I) + H(K)
%     PRINT 150,I,PI(I),K,H(K),D,J,PI(J)
% 150 FORMAT (' I,PI(I),K,H(K),D,J,PI(J)',/,7I6)
      IF ( D .GE. PI(J) ) GO TO 6
      PI(J)=D
      PB(J)=K
%     PRINT 160,J,PI(J),J,PB(J)
% 160 FORMAT (' RESULTADOS -- PI(',I3,')=',I4,' ,PB(',I3,')=',I3)
    6 CONTINUE
%
%  SELECCION
%
    4 D=9999
      IENT=0
      IFIN=1
%     PRINT 170,D,IENT,IFIN
% 170 FORMAT (' D,IENT,IFIN',/,3I6)
      DO 8 I=1,N
%     PRINT 180,I,S(I)
% 180 FORMAT (' S(',I3,')=',I2)
      IF ( S(I) .NE. 0 ) GO TO 8
%     PRINT 190,I,PI(I),D
% 190 FORMAT(' IFIN=0',/,' PI(',I3,')=',I5,' Y   D=',I5)
      IFIN=0
      IF ( PI(I) .GE. D) GO TO 8
      D=PI(I)
      IENT=I
%     PRINT 200,D,IENT
% 200 FORMAT(' RESULTADOS -- D=',I5,' ,IENT=',I4)
    8 CONTINUE
%
%  ADICION
%
      IF(D.LT.9999) GO TO 20
      IF(IFIN.EQ.1) GO TO 21
      NP=1
   21 CONTINUE
%     PRINT 210,NP
% 210 FORMAT (' EL VALOR FINAL DE NP ES :',I2)
      GO TO 25
   20 S(IENT)=1
%     PRINT 220,IENT
% 220 FORMAT (' RESULTADOS -- S(',I4,')=1')
%     PRINT 240,(S(I),I=1,N)
% 240 FORMAT (' S',/,(10I2))
      IF(IENT.EQ.TN) GO TO 21
      I=IENT
      GO TO 2
   25 CONTINUE
%     PRINT 56
%  56 FORMAT(///,' RUTA MAS CORTA DEL NODO RAIZ AL RESTO DE NODOS'
%    *,//,9X,1HI,6X,'PI(I)', 5X, 'PB(I)',/  )
%     PRINT 57, (I,PI(I),PB(I),I=1,N)
%  57 FORMAT (3I10)
%     PRINT 230
% 230 FORMAT (/,'FIN DE DSHORT',/)
      RETURN
      END


      SUBROUTINE PSHORT(CYC,J)
      COMMON /C/PB(0:120)/P/PI(0:120)/G/M,N
      DIMENSION LISA(0:120),LISN(0:120)
      INTEGER PB,DEL,PI,CYC
%
```

```
19060 %     ************************************************************
19070 %     *                                                          *
19080 %     PROPOSITO : CONOCIDO EL ARBOL ASOCIADO CON UNA SOLUCION
19090 %     BASICA FACTIBLE Y LOS POTENCIALES EN LOS NODOS,PROCEDE
19100 %     A VERIFICAR SI LA CONDICION DE FACTIBILIDAD DEL PROBLEMA
19110 %     DUAL SE SATISFACE PARA LOS ARCOS NO-BASICOS. SI NO SA-
19120 %     TISFACE, ENTONCES EL ALGORITMO PROCEDE ITERATIVAMENTE
19130 %     A MODIFICAR EL ARBOL (BASICO) Y LOS POTENCIALES EN LOS
19140 %     NODOS HASTA OBTENER UNA SOLUCION OPTIMA O DETECTAR UN
19150 %     CICLO NEGATIVO.
19160 %     *                                                          *
19170 %     ************************************************************
19180 %
19190       PRINT 120
20000 % 120 FORMAT (' SUBRUTINA PSHORT')
20010 %
20020 %  SELECCION
20030 %
20040     20 CALL SMNSP(I,J,KE,DEL)
20050        IF(DEL.EQ.0)GO TO 40
20060 %
20070 %  PIVOTE
20080 %
20090        KL=PB(J)
20100     15 IF(KL.EQ.0)GO TO 100
20110 %      PRINT 59,KL,KE,DEL
20120 %   59 FORMAT(5X,'ARCO SALIENTE=',I5,
20130 %     *5X,'ARCO ENTRANTE=',I5,X ,'DEL=',I5)
20140        CALL DELTRE(KL)
20150       CALL ADDTRE(KE)
20160 %
20170 %  DUAL
20180 %
20190        CALL ROOT(J,LISA,LISN,IC,CYC)
20200        IF(CYC.EQ.0)GO TO 22
20210        GO TO 40
20220     22 DO 30 L=1,IC+1
20230        II=LISN(L)
20240        PI(II)=PI(II)+DEL
20250     30 CONTINUE
20260 %      PRINT 110,(PI(L),L=1,N)
20270 % 110 FORMAT (' POTENCIALES EN LOS NODOS -- PI',/,(10I6))
20280        GO TO 20
20290    100 CYC = 1
20300 %      PRINT 59,KL,KE
20310     40 CONTINUE
20320 %      IF (CYC .EQ. 1) PRINT 52
20330 %   52 FORMAT(5X,'UN CICLO NEGATIVO FUE ENCONTRADO')
20340 %      PRINT 130
20350 % 130 FORMAT (' FIN DE PSHORT ')
20360        RETURN
20370        END
20380
20390
20400        SUBROUTINE SMNSP(I,J,KE,DEL)
20410        COMMON/A/O(0:120)/B/T(0:120)/G/H,N/L/H(0:120)/P/PI(0:120)
20420        INTEGER O,T,D,DEL,PI,H
20430 %
20440 %      ************************************************************
20450 %      *                                                          *
20460 %      PROPOSITO : ENCONTRAR EL ARCO CON EL VALOR MAS NEGATIVO DE
20470 %      PI(I) + H(K) - PI(J)
20480 %      *                                                          *
20490 %      ************************************************************
20500 %
20510 %      PRINT 100
20520 % 100 FORMAT (' SUBROUTINE SMNSP')
20530        DEL=9999
20540        KE=0
20550        DO 20 K=1,H
20560        I=O(K)
20570        J=T(K)
20580        D=PI(I)+H(K)-PI(J)
20590 %      PRINT 110,K,I,J,PI(I),H(K),PI(J),D
20600 % 110 FORMAT (' K,I,J,PI(I),H(K),PI(J),D',/,7I6)
20610        IF ( D .GE. 0) GO TO 20
```

```
      IF (D .GE. DEL ) GO TO 20
      DEL=D
      KE=K
%     PRINT 120,DEL,KE
% 120 FORMAT (' RESULTADOS -- DEL=',I5,' KE=',I4)
   20 CONTINUE
      IF ( DEL .GE. 0 ) GO TO 10
      I=O(KE)
      J=T(KE)
   10 CONTINUE
%     PRINT 150,DEL
      IF (DEL .GT. 0) DEL = 0
%     PRINT 130,I,J,KE,DEL
% 130 FORMAT(' VALORES ',/,'    I     J    KE   DEL',/,4I6)
%     PRINT 140
% 140 FORMAT (' FIN DE SHNSP')
      RETURN
      END
%

      SUBROUTINE STARTD(SN)
      COMMON/G/N,N/A/O(0:120)/P/PI(0:120)/L/H(0:120)/B/T(0:120)
      DIMENSION LISA(0:120),LISN(0:120)
      INTEGER SN,PI,H,T,O
%
%     ************************************************************
%     *                                                          *
%     PROPOSITO : DADO EL ARBOL BASICO  ESTE ALGORITMO ESTABLECE
%     LOS POTENCIALES EN LOS NODOS HACIENDO PI(J)=PI(I) + H(K) ,
%     PARA CADA ARCO BASICO K(I,J).
%     *                                                          *
%     ************************************************************
%
%     PRINT 10,SN
%  10 FORMAT(' SUBRUTINA STARTD CON EL NODO',I4,' COMO LA RAIZ ')
      CALL ROOT (SN,LISA,LISN,IC,CYC)
      PI(SN)=0
      IF (IC.LE.0) GO TO 4
      DO 3 L=1,IC
      K=LISA(L)
      J=LISN(L+1)
       I=O(K)
      PI(J) = PI (I) + H(K)
%     PRINT 40,K,J,I,PI(I),H(K),PI(J)
%  40 FORMAT (' K,J,I,PI(I),H(K),P(J)',/,6I6)
    3 CONTINUE
%     PRINT 30,(PI(L),L=1,N)
%  30 FORMAT (' POTENCIALES EN LOS NODOS -- PI',/,(10I6))
      GO TO 5
    4 CONTINUE
%     PRINT 20,SN
%  20 FORMAT(' IC=0, EL ARBOL ESTA AISLADO EN EL NODO',I4)
    5 RETURN
      END


      SUBROUTINE NDSHOR(SN,CYC)
      COMMON/A/O(0:120)/B/T(0:120)/C/PB(0:120)/D/PF(0:120)
     */F/PR(0:120)/G/M,N/K/C(0:120)/L/H(0:120)/O/F(0:120)/P/PI(0:120)
      DIMENSION LISA(0:120),LISN(0:120)
      INTEGER SN,AD,PI,H,O,T,PB
%
%     ************************************************************
%     *                                                          *
%     PROPOSITO : ENCONTRAR LA LONGITUD DE LA TRAYECTORIA MAS
%     CORTA DEL NUDO FUENTE A TODOS LOS DEMAS A TRAVES DE LOS
%     ARCOS. EL ALGORITMO USA UNA REPRESENTACION NO BASICA.
%     *                                                          *
%     ************************************************************
%
%     PRINT 100,SN
% 100 FORMAT (' SUBRUTINA NDSHOR -- SN=',I4)
      DO 1 I =1,N
    1 PI(I)=9999
      PI(SN) = 0
      CYC = 0
```

```
      IT = 0
    8 IND=0
      IT = IT + 1
      DO 10 K=1,M
      I=O(K)
      J=T(K)
C     PRINT 110,K,I,J,PI(I),H(K),PI(J)
C110 FORMAT (' K,I,J,PI(I),H(K),PI(J) ',/,6I6)
      IF ( PI(I) + H(K) .GE. PI(J) ) GO TO 10
      PI(J)=PI(I) + H(K)
      PB(J) = K
      IND = 1
C     PRINT 120,J,PI(J),J,PB(J)
C120 FORMAT (' RESULTADOS -- PI(',I3,')=',I4,' ,PB(',I3,')=',I4)
   10 CONTINUE
C     PRINT 60
C 60 FORMAT (//,14X,'I',13X,'PI',13X,'PB')
C     PRINT 50,(I,PI(I),PB(I),I=1,N)
C 50 FORMAT(10X,I5,10X,I5,10X,I5)
      IF ( IND .EQ. 0 ) GO TO 99
      IF (IT .LT. N) GO TO 8
      CYC = 1
C     PRINT 130
C130 FORMAT(' LA RED TIENE UN CICLO NEGATIVO ')
   99 RETURN
      END


      SUBROUTINE SHORT(SN,CYC,J)
      COMMON/C/PB(0:120)/D/PF(0:120)/F/PR(0:120)/G/M,N/P/PI(0:120)/NP/NP
      INTEGER CYC,SN,PB,PF,PR,AD
C     ***********************************************************
C     * PROPOSITO :                                             *
C     DETERMINAR EL ARBOL DE TRAYECTORIA MAS CORTA EN UNA RED QUE
C     PUEDE TENER ARCOS CON COSTOS NEGATIVOS. EL ALGORITMO TERMINA
C     CON EL ARBOL DE TRAYECTORIA MAS CORTA (DE SN A LOS DEMAS
C     NODOS) O BIEN ENCUENTRA UN CICLO NEGATIVO (CYC=1) Y PROPOR-
C     CIONA UN NODO J EN DICHO CICLO.
C     *                                                         *
C     ***********************************************************
C
C     PRINT 50
C  50 FORMAT (' SUBROUTINE SHORT ')
C
C  INICIA
C
      CALL DSHORT(SN,0,NP)
      IF(NP.EQ.1)GO TO 20
C
C  ARBOL
C
      CALL TREINT(N)
C
C  PRIMAL
C
      CALL PSHORT(CYC,J)
C     PRINT 60
C  60 FORMAT (' FIN DE  SHORT ')
   20 RETURN
      END

      SUBROUTINE PHASE1(INF)
      COMMON /M/B(0:120) /G/ M,N /O/F(0:120)
      INTEGER B,F,BF,BT,VR,VS,BS
      DIMENSION BF(0:120)
C     ***********************************************************
C     *                                                         *
C      PROPOSITO : ENCONTRAR UN FLUJO FACTIBLE EN LA RED QUE SA-
C      TISFAGA LOS FLUJOS EXTERNOS EN LOS NODOS.
C     *                                                         *
C     ***********************************************************
C
```

```
C     PRINT 100
C 100 FORMAT (' SUBRUTINA PHASE1')
C
C     INICIA
C
      DO 5 K = 1,M
    5 F(K) = 0
      DO 10 I = 1,N-1
   10 BF(I) = B(I)
C     PRINT 94,(BF(I),I=1,N-1)
C  94 FORMAT (' BF(I),I=1,N-1',/,(10I5))
      IS = 1
      BF(N) = 0
C
C     FUENTE
C
   15 NS = N
      BS = 9999
C      PRINT 95,IS
C  95 FORMAT (' IS',IS)
      IT = 1
      IF (IS .GE. N) GO TO 35
      DO 30 I=IS,N-1
      IF ( BF(I) .LE. 0) GO TO 30
      NS = I
      BS = BF(I)
      GO TO 35
   30 CONTINUE
   35 CONTINUE
C      PRINT 96,NS,BS
C  96 FORMAT (' NS,BS',2I7)
C
C     FUENTE
C
      NT=N
      BT = 9999
C      PRINT 98,IT
C  98 FORMAT (' IT',I7)
      IF ( IT .GE. N) GO TO 50
      DO 45 I = IT,N-1
      IF (BF(I) .GE. 0) GO TO 45
      NT = I
      BT = -BF(I)
      IT =I
      GO TO 50
   45 CONTINUE
   50 CONTINUE
C      PRINT 97,NT,BT
C  97 FORMAT (' NT,BT',2I7)
C
C     CHEQUEA
      IF (NS .NE. N .OR. NT .NE. N) GO TO 60
      INF=0
      RETURN
   60 CONTINUE
C
C     FLUJO
C
      IF(BS .GT. BT) VR = BT
      IF(BS .LE. BT) VR = BS
C     PRINT 91,NS,NT,VR
C  91 FORMAT (' NS,NT,VR',3I7)
      CALL MAXFLO(NS,NT,VR,MF,INT)
C      PRINT 92,MF,INT
C  92 FORMAT (' MF,INT',2I7)
C      PRINT 93,(I,F(I),I=1,N)
C  93 FORMAT('    I    F(I)',/,(2I5))
      BF(NS)=BF(NS) - MF
      BS=BF(NS)
      BF(NT)=BF(NT) + MF
      BT=BF(NT)
C      PRINT 99,NS,BF(NS),NT,BF(NT)
C  99 FORMAT (' NS,BF(NS),NT,BF(NT)'4I7)
C
C     CONTROL
C
```

```
      IF ( NS .NE. N) GO TO 70
      BS = 9999
      PRINT 110,N,BS
110   FORMAT (' NS=N=',I3,',LUEGO,BS=',I5)
      IF ( BF(NT) .NE. 0 ) GO TO 90
      IT = IT + 1
      GO TO 35
70    IF(BF(NS) .NE. 0)GO TO 80
      IS = IS + 1
      GO TO 15
80    IF(NT .EQ. N)GO TO 90
      IT = IT + 1
      GO TO 35
90    INF = 1
      RETURN
      END

      SUBROUTINE PRIMA1
      COMMON/A/D(0:120)/B/T(0:120)/C/PB(0:120)/F/PR(0:120)/G/M,N
     */H/PT(0:120)/I/LT(0:120)/J/PD(0:120)/K/C(0:120)/L/H(0:120)
     */M/B(0:120)/N/CL(0:120)/O/F(0:120)/B/PF(0:120)/P/PI(0:120)
     */Q/PP(0:120)
      DIMENSION LISA(0:120),LISN(0:120)
      INTEGER D,T,PB,PF,PR,PT,PD,C,H,B,CL,CYC
      INTEGER SN
C     ************************************************************
C     *                                                          *
C     PROPOSITO : EMPEZANDO CON UNA SOLUCION FACTIBLE PRIMAL,
C     PROCEDE A ENCONTRAR LA SOLUCION OPTIMA DESCUBRIENDO Y SATU-
C     RANDO ITERATIVAMENTE, CICLOS CON COSTO NEGATIVO EN LA
C     RED MARGINAL.
C     *                                                          *
C     ************************************************************
C
      PRINT 110
110   FORMAT (' SUBRUTINA PRIMA1')
C
C  CICLO
C
   10 DO 37 I=1,N
      PP(I) = 0
      PB(I)=0
      PF(I)=0
   37 PR(I)=0
      CYC = 0
      SN=N
      CALL SHORTM (SN,CYC,JJ)
      IC = 0
      IJ = JJ
C     PRINT 100,CYC
100   FORMAT (' CYC=',I3)
      IF(CYC.NE.1) GO TO 20
C
C  LISTA DE ARCOS
C
   15 K=PB(IJ)
      IC = IC+1
      LISA(IC) = K
      IF (K .LT. 0) IJ = T(-K)
      IF ( K .GT. 0) IJ = D(K)
      IF ( IJ .NE. JJ ) GO TO 15
C     PRINT 130,(LISA(L),L=1,IC)
130   FORMAT (' ARCS ON CYCLE',/,(10I3))
C
C  CAMBIO
C
      CALL MFLO(LISA,IC,MF,KL,ILC)
      CALL FLOCHG(LISA,IC,MF)
C     CALL POUTPU
      GO TO 10
```

```
23620     20 CONTINUE
23630 %        PRINT 120
23640 % 120 FORMAT (' FIN DE  PRIMA1',/)
23650        RETURN
23660        END
23670
23680
23690        SUBROUTINE SHORTM(SN,CYC,J)
23700        COMMON/C/PB(0:120)/D/PF(0:120)/F/PR(0:120)/G/M,N/P/PI(0:120)
23710        COMMON /PRNL1/DM,DZ,SPNTRE/SHRT/S(0:120)
23720        INTEGER CYC,SN,PB,PF,PR,AD,S
23730        INTEGER PI
23740        INTEGER DM,DZ,SPNTRE
23750 %
23760 %      ************************************************************
23770 %      *                                                          *
23780 %      PROPOSITO : ENCONTRAR EL ARBOL DE RUTA MAS CORTA EN UNA RED
23790 %      QUE PUEDE TENER COSTOS NEGATIVOS EN LOS ARCOS. CONSIDERANDO
23800 %      LOS ARCOS HACIA ADELANTE Y LOS REFLEJADOS. EL ALGORITMO
23810 %      TERMINA CUANDO SE ENCUENTRA EL ARBOL DE RUTAS MAS CORTAS O
23820 %      CUANDO SE DETECTA UN CICLO.
23830 %      *                                                          *
23840 %      ************************************************************
23850 %
23860  %     PRINT 50
23870  % 50 FORMAT (' SUBROUTINE SHORTM ')
23880  %
23890  %   INICIA
23900  %
23910        DM = 9999
23920        DZ = 0
23930        SPNTRE = 0
23940  %
23950  %   EXPANDE
23960  %
23970     10 CONTINUE
23980        CALL DSHRTM ( SN,0,NP)
23990  %     PRINT 100,NP
24000  %100 FORMAT (' NP =',I4)
24010        IF(NP.NE.1)GO TO 20
24020        DO 30 I = 1,N
24030  %     PRINT 110,S(I),I
24040  %110 FORMAT (' S(I)=',I4,' I=',I4)
24050        IF (S(I) .EQ. 1) GO TO 30
24060        PI(I) = PI(I) + 9999
24070        SN = I
24080  %     PRINT 120,I,PI(I),SN
24090  %120 FORMAT (' I=',I4,' PI(I)=',I6,' SN=',I4)
24100     30 CONTINUE
24110        DM = DM + 9999
24120        DZ = DZ + 9999
24130        SPNTRE = 1
24140  %     PRINT 130,DM,DZ,SPNTRE
24150  %130 FORMAT (' DM=',I6,' DZ=',I6,' SPNTRE=',I3)
24160        GO TO 10
24170     20 CONTINUE
24180  %
24190  %   ARBOL
24200  %
24210        CALL TREINT(N)
24220  %
24230  %   PRIMAL
24240  %
24250        CALL PSHRTM(CYC,J)
24260  %     PRINT 60
24270  % 60 FORMAT (' FIN DE SHORTM ')
24280        RETURN
24290        END
24300
24310
24320        SUBROUTINE DSHRTM(SN,TN,NP)
24330        COMMON /C/PB(0:120)/P/PI(0:120)/L/H(0:120)/G/M,N/SHRT/S(0:120)
24340        COMMON /PRNL1/DM,DZ,SPNTRE
24350        INTEGER SN,PI,PB,H,TN,AD,S,D
24360        INTEGER DM,DZ,SPNTRE
24370        DIMENSION LISA(0:120),LISH(0:120)
```

```
24380 %
24390 %     ************************************************************
24400 %     *                                                          *
24410 %     PROPOSITO : ENCONTRAR LA RUTA MAS CORTA DEL ARBOL EN UNA
24420 %     RED QUE PUEDE TENER ARCOS NEGATIVOS, CONSIDERANDO LOS ARCOS
24430 %     HACIA ADELANTE Y REFLEJADOS. EL ALGORITMO TERMINA CUADO SE
24440 %     ENCUENTRA LA RUTA MAS CORTA DEL ARBOL O UN CICLO.
24450 %     *                                                          *
24460 %     ************************************************************
24470 %
24480 %
24490 %  INICIA
24500 %
24510       NP = 0
24520  %    PRINT 100,SN,TN,NP
24530  %100 FORMAT (' SUBRUTINA DSHRTM-- SN=',I4,',TN=',I4,',NP=',I3)
24540       IF ( SENTRE .EQ.1) GO TO 61
24550       DO 1 I=1,N
24560       PI(I)=DM
24570       S(I) = 0
24580     1 PB(I)=0
24590    61 PI(SN)=DZ
24600       I=SN
24610       S(SN)=1
24620  %
24630  %  ADELANTE
24640  %
24650  %    PRINT 110,(PI(IW),IW=1,N)
24660  %    PRINT 120,(PB(IW),IW=1,N)
24670  %    PRINT 130,(S(IW),IW=1,N)
24680  %110 FORMAT (' PI',/,(20I6))
24690  %120 FORMAT (' PB',/,(20I6))
24700  %130 FORMAT (' S',/,(20I6))
24710     2 CALL ORIG(I,LISA,LISN,IC)
24720       IF (IC .LE.0) GO TO 3
24730       DO 6 L = 1,IC
24740       K=LISA(L)
24750       J=LISN(L)
24760  %    PRINT 260,K,AD(K)
24770  %260 FORMAT (' K=',I3,',AD(K)=',I2)
24780       IF (AD(K) .NE. 1) GO TO 6
24790  %    PRINT 140,J,S(J)
24800  %140 FORMAT (' S(',I3,')=',I2)
24810       IF(S(J) .NE. 0) GO TO 6
24820       D=PI(I) + H(K)
24830  %    PRINT 150,I,PI(I),K,H(K),D,J,PI(J)
24840  %150 FORMAT (' I,PI(I),K,H(K),D,J,PI(J)',/,7I6)
24850       IF ( D .GE. PI(J) ) GO TO 6
24860       PI(J)=D
24870       PB(J)=K
24880  %    PRINT 160,J,PI(J),J,PB(J)
24890  %160 FORMAT (' RESULTADOS -- PI(',I3,')=',I4,' ,PB(',I3,')=',I3)
24900     6 CONTINUE
24910  %
24920  %  REFLEJADO
24930  %
24940     3 CALL TERM(I,LISA,LISN,IC)
24950       IF ( IC .LE. 0) GO TO 4
24960       DO 7 L=1,IC
24970       K=LISA(L)
24980       J=LISN(L)
24990  %    PRINT 250,K,AD(-K)
25000  %250 FORMAT (' K=',I3,',AD(-K)=',I2)
25010  %    PRINT 140,J,S(J)
25020       IF(AD(-K).NE.1) GO TO 7
25030       IF ( S(J) .NE. 0 ) GO TO 7
25040       D=PI(I)-H(K)
25050  %    PRINT 150,I,PI(I),K,H(K),D,J,PI(J)
25060       IF ( D .GE. PI(J) ) GO TO 7
25070       PI(J)=D
25080       PB(J)=-K
25090  %    PRINT 160,J,PI(J),J,PB(J)
25100     7 CONTINUE
25110  %
```

```
25120  %  SELECCIONA
25130  %
25140      4 D=DM
25150        IENT=0
25160        IFIN=1
25170  %     PRINT 170,D,IENT,IFIN
25180  %170 FORMAT (' D,IENT,IFIN',/,3I6)
25190        DO 8 I=1,N
25200  %     PRINT 180,I,S(I)
25210  %180 FORMAT (' S(',I3,')=',I2)
25220        IF ( S(I) .NE. 0 ) GO TO 8
25230  %     PRINT 190,I,PI(I),D
25240  %190 FORMAT(' IFIN=0',/,' PI(',I3,')=',I5,' AND D=',I5)
25250        IFIN=0
25260        IF ( PI(I) .GE. D) GO TO 8
25270        D=PI(I)
25280        IENT=I
25290  %     PRINT 200,D,IENT
25300  %200 FORMAT(' RESULTADOS -- D=',I5,' ,IENT=',I4)
25310     8 CONTINUE
25320  %
25330  %  ADICIONA
25340  %
25350        IF(D.LT.DM) GO TO 20
25360        IF(IFIN.EQ.1) GO TO 21
25370        NP=1
25380    21 CONTINUE
25390  %     PRINT 210,NP
25400  %210 FORMAT ('  VALOR FINAL DE NP ',I2)
25410        GO TO 25
25420    20 S(IENT)=1
25430  %     PRINT 220,IENT
25440  %220 FORMAT (' RESULTADOS -- S(',I4,')=1')
25450  %     PRINT 240,(S(I),I=1,N)
25460  %240 FORMAT (' S',/,(10I2))
25470        IF(IENT.EQ.TN) GO TO 21
25480        I=IENT
25490        GO TO 2
25500    25 CONTINUE
25510  %     PRINT 56
25520  % 56 FORMAT (////,5X,'RUTA MAS CORTA DESDE EL NODO RAIZ AL RESTO'
25530  %   *,///,9X,1HI,6X,'PI(I)', 5X, 'PB(I)',6X,'S(I)',/  )
25540  %     PRINT 57, (I,PI(I),PB(I),S(I),I=1,N)
25550  % 57 FORMAT (4I10)
25560  %     PRINT 230
25570  %230 FORMAT (' FIN DE DSHRTM',/)
25580  %     PRINT 310,N
25590  %310 FORMAT (' N=',I4)
25600        RETURN
25610        END
25620
25630
25640        SUBROUTINE PSHRTM(CYC,J)
25650        COMMON /C/PB(0:120)/P/PI(0:120)/G/M,N
25660        DIMENSION LISA(0:120),LISN(0:120)
25670        INTEGER PB,DEL,PI,CYC
25680  %
25690  %     ************************************************************
25700  %     *                                                          *
25710  %      PROPOSITO : EMPEZANDO CON UN ARBOL BASICO FACTIBLE Y CON
25720  %      POTENCIALES EN LOS NODOS QUE VIOLAN LAS CONDICIONES DE
25730  %      FACTIBILIDAD DEL DUAL PARA ALGUNOS ARCOS NO BASICOS, ESTE
25740  %      ALGORITMO MODIFICA, ITERATIVAMENTE, EL ARBOL BASICO Y LOS
25750  %      POTENCIALES EN LOS NODOS PARA OBTENER UNA SOLUCION OPTIMA.
25760  %      *                                                         *
25770  %      ************************************************************
25780  %
25790   %     PRINT 120
25800   %120 FORMAT (' SUBRUTINA PSHRTM')
25810  %
25820  %  SELECCIONA
25830  %
25840     20 CALL SHNSPN(I,J,KE,DEL)
25850        IF(DEL.EQ.0)GO TO 40
25860  %
```

```
25070 %  PIVOTE
25080 %
25090        KL=PB(J)
25900        IF(KL.EQ.0)GO TO 100
25910  %     PRINT 59,KL,KE,DEL
25920  % 59 FORMAT(5X,'ARCO SALIENTE=',I5,
25930  %    *5X,'ARCO ENTRANTE=',I5,X ,'DEL=',I5)
25940        CALL DELTRE(KL)
25950        CALL ADDTRE(KE)
25960 %
25970 %  DUAL
25980 %
25990        CALL ROOT(J,LISA,LISN,IC,CYC)
26000        IF(CYC.NE.0) GO TO 40
26010     22 DO 30 L=1,IC+1
26020        II=LISN(L)
26030        PI(II)=PI(II)+DEL
26040     30 CONTINUE
26050  %     PRINT 110,(PI(L),L=1,N)
26060  %110 FORMAT (' POTENCIALES EN LOS NODOS -- PI',/,(10I6))
26070        GO TO 20
26080    100 CYC = 1
26090        PB(J) = KE
26100  %     PRINT 59,KL,KE,DEL
26110     40 CONTINUE
26120  %     IF (CYC .EQ. 1) PRINT 52
26130  % 52 FORMAT(5X,'SE ENCONTRO UN CICLO NEGATIVO')
26140  %     PRINT 130
26150  %130 FORMAT (' FIN DE PSHRTN ')
26160        RETURN
26170        END
26180
26190
26200        SUBROUTINE SMNSPM(I,J,KE,DEL)
26210        COMMON/A/O(0:120)/B/T(0:120)/G/M,N/L/H(0:120)/P/PI(0:120)
26220        INTEGER O,T,D,DEL,PI,H,AD
26230 %
26240 %      ************************************************************
26250 %      *                                                          *
26260 %       PROPOSITO : ENCONTRAR EL ARCO ADMISIBLE QUE TENGA EL VALOR
26270 %       MAS NEGATIVO DE PI(I) + H(K) - PI(J).
26280 %      *                                                          *
26290 %      ************************************************************
26300 %
26310  %     PRINT 100
26320  %100 FORMAT (' SUBROUTINE SMNSPM')
26330        DEL=9999
26340        KE=0
26350        DO 20 K=1,M
26360        I=O(K)
26370        J=T(K)
26380        D=PI(I)+H(K)-PI(J)
26390        IF ( D .EQ. 0) GO TO 20
26400  %     PRINT 110,K,I,J,PI(I),H(K),PI(J),D
26410  %110 FORMAT (' K,I,J,PI(I),H(K),PI(J),D',/,7I6)
26420        IF ( D .GE. 0) GO TO 30
26430        IF (D .GE. DEL ) GO TO 20
26440        IF ( AD(K) .EQ. 0) GO TO 20
26450        DEL=D
26460        KE=K
26470  %     PRINT 120,DEL,KE
26480  %120 FORMAT (' RESULTADOS -- DEL=',I5,' KE=',I4)
26490        GO TO 20
26500     30 IF(-D.GE.DEL)GO TO 20
26510        IF(AD(-K).EQ.0)GO TO 20
26520        DEL=-D
26530        KE=-K
26540  %     PRINT 120,DEL,KE
26550     20 CONTINUE
26560        IF(KE.EQ.0)GO TO 55
26570        IF(KE.GT.0)GO TO 50
26580        I=T(-KE)
26590        J=O(-KE)
```

```
      GO TO 40
   50 I=O(KE)
      J=T(KE)
   40 CONTINUE
C     PRINT 150,DEL
C150 FORMAT (' DEL=',I6)
      IF (DEL .GT. 0) DEL = 0
C     PRINT 130,I,J,KE,DEL
C130 FORMAT(' VALORES RETORNADOS',/,'    I     J    KE   DEL'
C    *,/,4I6)
C     PRINT 140
C140 FORMAT (' FIN DE SMNSPM')
      RETURN
      END
C
C
C
      SUBROUTINE ARTIFI
      COMMON/A/O(0:120)/B/T(0:120)/C/PB(0:120)/F/PR(0:120)/G/M,N
     */H/PT(0:120)/I/LT(0:120)/J/PO(0:120)/K/C(0:120)/L/H(0:120)/M/
     *B(0:120)/N/CL(0:120)/O/F(0:120)/D/PF(0:120)/PR/IPRINT/P/PI
     *(0:120)/Q/PP(0:120)
      DIMENSION LISA(0:120),LISN(0:120)
      INTEGER O,T,PB,PF,PR,PT,PO,C,H,B,CL,F
      REAL LOWER
C     PRINT 220
C     ************************************************************
C     *                                                          *
C     PROPORCIONAR UNA SOLUCION BASICA INICIAL ARTIFICIAL PARA
C     UNA RED DE FLUJO A COSTO MINIMO.
C     *                                                          *
C     ************************************************************
C
C220 FORMAT (' SUBRUTINA ARTIFI')
C
C  INICIA
C
      DO 10 K=1,M
   10 F(K)=0
      NN=N-1
C     PRINT 120,NN
C120 FORMAT (' N-1=',I3)
C
C  ARCOS
C
      DO 40 I=1,NN
C     PRINT 100,I,B(I)
C100 FORMAT (' I=',I3,',B(I)=',I3)
      IF(B(I).LT.0) GO TO 30
      LOWER=0
      UPPER=B(I)
      IF (UPPER .EQ. 0.) UPPER=9999.
      II=I
      JJ=N
   20 COST=9999
C     PRINT 110,LOWER,UPPER,II,JJ,COST
      CALL ORIGS(II,JJ,LOWER,UPPER,COST)
      GO TO 40
   30 LOWER=0
      UPPER=-B(I)
      II=N
      JJ=I
C110 FORMAT (' INFE.=',F7.1,',SUPE.=',F7.1,',II=',I3,',JJ=',I3
C    *    ,',COST=',F7.1)
      GO TO 20
   40 CONTINUE
C
C  NUEVA LISTA
C
      LM=M
C     PRINT 130,LM
C130 FORMAT (' LM=',I3)
      M=0
```

```
      DO 50 K=1,LM
      J=T(K)
      M=M+1
%     PRINT 140
% 140 FORMAT (' K=',I3,',J=T(K)=',I3,'M=',I3)
      CALL TERMS(K,J)
   50 CONTINUE
%
%  FLUJOS
%
      CALL ORIG(N,LISA,LISN,L)
%     PRINT 150 ,N,L,(LISA(I),I=1,L)
% 150 FORMAT (' N=',I3,',L=',I3,/,' ARCOS ORIGINA. EN EL NODO N'
%    *         ,/,(10I3))
%     PRINT 160,(LISN(I),I=1,L)
% 160 FORMAT (' NODOS TERMINALES DE LOS ARCOS',/,(10I3))
      IF(L.EQ.0) GO TO 70
      DO 60 I=1,L
      K=LISA(I)
%     PRINT 170,K,H(K)
% 170 FORMAT (' K=',I3,',H(K)=',I5)
      IF ( H(K) .LT. 9999) GO TO 60
      F(K) = C(K)
%     PRINT 180,F(K)
% 180 FORMAT (' F(K)=',I3)
      PB(T(K)) = K
   60 CONTINUE
   70 CALL TERM(N,LISA,LISN,L)
%     PRINT 190,N,L,(LISA(I),I=1,L)
% 190 FORMAT(' N=',I3,',L=',I3,/,
%    *' ARCOS QUE TERMINA EN EL NODO N',/,(10I3))
%     PRINT 200,(LISN(I),I=1,L)
% 200 FORMAT (' ARCOS ORIGINADOS EN LOS NODOS',/,(10I3))
      IF(L.EQ.0) GO TO 90
      DO 80 I=1,L
%     PRINT 170,K,H(K)
      K=LISA(I)
      IF (H(K) .LT. 9999) GO TO 80
      F(K)=C(K)
      IF (F(K) .GE. 9999) F(K)=0
%     PRINT 180,F(K)
      PB(O(K)) = -K
   80 CONTINUE
   90 CONTINUE
      PB(N) = 0
      CALL TREINT(N)
      CALL STARTM(N)
      PRINT 1, N,M
    1 FORMAT(//,10X,'FORMACION DE LA RED CON ARCOS ARTIFICIALES',
     * //,15X,'NRO. DE NODOS--',I5,10X,'NRO. DE ARCOS--',I5)
      PRINT 2
    2 FORMAT (//,5X,'PARAMETROS TRANSFORMADOS DE LOS NODOS',//,
     * 16X,'NODO I',5X,'B(I)',/)
      DO 5 I=1,N
      PRINT 3, I,B(I)
    3 FORMAT (10X,2I10)
    5 CONTINUE
      PRINT 4
    4 FORMAT(//,5X,'PARAMETROS TRANSFORMADOS DE LOS ARCOS',//,6X,
     *'ARCO K',5X,'O(K)',6X,'T(K)',6X,'C(K)',6X,'H(K)',6X,'F(K)',/)
      PRINT 6, (J,O(J),T(J),C(J),H(J), F(J),J=1,M)
    6 FORMAT ( 6I10)
%     PRINT 7
%   7 FORMAT(///,' INDICE I/L',5X,'PO(I)',5X,'PT(I)',5X,'LT(L)')
%     PRINT 8, (I,PO(I),PT(I),LT(I), I=1,M)
%   8 FORMAT (4I10)
%
%
%  OBTIENE LA LISTA DE NODOS ORIGEN Y TERMINAL E IMPRIME
%     PRINT 11
%  11 FORMAT (////,5X,'ORIGEN DE LOS ARCOS-----',//,
%    * 5X,'NODO NRO.',5X,'ARC.ORIG.NRO.',5X,'NODO TERM.NRO.')
      DO 15 I=1,N
      CALL ORIG (I,LISA,LISN,L)
%     IF(L.LE.0) GO TO 16
      DO 15 K=1,L
```

```
28090 %      PRINT 13, I,LISA(K),LISN(K)
28100 %   13 FORMAT (5X,I5,12X,I6,15X,I5)
28110        GO TO 15
28120 %   16 PRINT 13, I
28130     15 CONTINUE
28140 %
28150 %
28160 %      PRINT 14
28170 %   14 FORMAT (////,5X,'TERMINACION DE ARCOS-----',//,
28180 %     * 5X,'NODO NRO.',5X,'ARC.TERM.NRO.',5X,'NODO ORIG.NRO.')
28190        DO 31 I=1,N
28200        CALL TERM(I,LISA,LISN,L)
28210 %      IF(L.LE.0) GO TO 25
28220        DO 31 K=1,L
28230 %      PRINT 13, I,LISA(K),LISN(K)
28240        GO TO 31
28250 %   25 PRINT 13,I
28260     31 CONTINUE
28270 %      PRINT 42
28280 %   42 FORMAT (' ARBOL BASICO INICIAL ARTIFICIAL')
28290 %      PRINT 41
28300 %   41 FORMAT (//,7X,'NODE',7X,'PB',8X,'PF',8X,'PR',8X,'PP',/)
28310 %      DO 45 I = 1,N
28320 %      PRINT 38,I,PB(I),PF(I),PR(I),PP(I)
28330 %   38 FORMAT (4I10)
28340 %   45 CONTINUE
28350 %      PRINT 43
28360 %   43 FORMAT (' POTENCIALES EN LOS NODOS',/,4X,'I',4X,'PI')
28370 %      PRINT 44 ,(I,PI(I),I=1,N)
28380 %   44 FORMAT (2I5)
28390 %      PRINT 230
28400 % 230 FORMAT (' FIN DE ARTIFI',/)
28410        RETURN
28420        END
28430
28440
28450        SUBROUTINE STARTM(SN)
28460        COMMON/G/M,N/A/O(0:120)/P/PI(0:120)/L/H(0:120)/B/T(0:120)
28470        DIMENSION LISA(0:120),LISN(0:120)
28480        INTEGER SN,PI,H,T,O
28490 %
28500 %      ************************************************************
28510 %      *                                                          *
28520 %       PROPOSITO : ENCONTRAR UN ARBOL BASICO, DETERMINA LOS POTEN-
28530 %       CIALES EN LOS NODOS  TAL QUE PI(I) + H(K) = PI(J) , PARA
28540 %       CADA ARCO BASICO K(I,J). CONSIDERANDO TANTO LOS ARCOS HACIA
28550 %       ADELANTE COMO LOS REFLEJADOS.
28560 %       *                                                          *
28570 %      ************************************************************
28580 %
28590 %      PRINT 10,SN
28600 %   10 FORMAT (' SUBRUTINA STARTM CON EL NODO',I4,' COMO RAIZ')
28610        CALL ROOT (SN,LISA,LISN,IC,CYC)
28620        PI(SN)=0
28630        IF  (IC.LE.0) GO TO 4
28640        DO 3 L=1,IC
28650         K=LISA(L)
28660        J=LISN(L+1)
28670        IF ( K .LT. 0) GO TO 1
28680         I=O(K)
28690        PI(J) = PI (I) + H(K)
28700 %      PRINT 40,K,J,I,PI(I),H(K),PI(J)
28710 %   40 FORMAT (' K,J,I,PI(I),H(K),P(J)',/,6I6)
28720        GO TO 3
28730      1 I= T(-K)
28740        PI(J) = PI(I) - H(-K)
28750 %      PRINT 40,K,J,I,PI(I),H(-K),PI(J)
28760     3 CONTINUE
28770 %      PRINT 30,(PI(L),L=1,N)
28780 %   30 FORMAT (' POTENCIALES EN LOS NODOS -- PI',/,(10I6))
28790        GO TO 5
28800      4 CONTINUE
28810 %      PRINT 20,SN
28820 %   20 FORMAT (' IC=0, EL ARBOL ESTA AISLADO EN EL NODO',I4)
28830      5 RETURN
28840        END
```

```
28850 %
28860
28870       SUBROUTINE PRIMAL
28880       COMMON/A/O(0:120)/B/T(0:120)/C/PB(0:120)/D/PF(0:120)/F/PR(0:120)
28890      */G/H,N/H/PT(0:120)/I/LT(0:120)/J/PO(0:120)/K/C(0:120)/L/H(0:120)
28900      */M/B(0:120)/N/CL(0:120)/O/F(0:120)/P/PI(0:120)
28910       COMMON/W/SN,FN
28920       DIMENSION LISA(0:120),LISN(0:120)
28930       INTEGER O,T,PB,PF,PR,PT,LT,PO,C,H,B,CL,F,DEL,PI,SN,FN
28940 %
28950 %     ************************************************************
28960 %     *                                                          *
28970 %     PROPOSITO : EJECUTAR LA TECNICA DEL PRIMAL SIMPLEX PARA LOS
28980 %     PROBLEMAS DE REDES DE FLUJO A COSTO MINIMO. EL ALGORITMO
28990 %     INICIA CON UNA SOLUCION BASICA FACTIBLE.
29000 %     *                                                          *
29010 %     ************************************************************
29020 %
29030  %    PRINT 110
29040  %110 FORMAT (' SUBRUTINA PRIMAL')
29050       IST=1
29060 %
29070 %  SELECCIONA
29080 %
29090   200 CALL SELECT(IST,KE,DEL,IFIN)
29100       IF (IFIN .EQ. 1) GO TO 999
29110 %
29120 %  FLUJO
29130 %
29140       IF ( KE .GT. 0 ) GO TO 30
29150       IS=O(-KE)
29160       IT=T(-KE)
29170       GO TO 40
29180    30 IT=O(KE)
29190       IS=T(KE)
29200    40 CALL TPATH(IS,IT,LISA,LISN,IC,JUNC,NP)
29210       IC=IC+1
29220       LISA(IC)=KE
29230  %    IF (KE .GT. 0) LISN(IC) = O(KE)
29240  %    IF(KE .LT. 0) LISN(IC) = T(-KE)
29250  %    PRINT 81,KE,DEL
29260  % 81 FORMAT ( ' ARC ',I3,' ENTRA , DEL = ',I5)
29270  %    PRINT 84
29280  % 84 FORMAT (' CYCLE ' )
29290  %    PRINT 82 , (LISA(I),I=1,IC)
29300  % 82 FORMAT ( ' ARCS ' ,10I5)
29310  %    PRINT 83,(LISN(I),I=1,IC)
29320  % 83 FORMAT (' NODES ',10I5)
29330       CALL MFLO(LISA,IC,MF,KL,ILC)
29340       CALL FLOCHG(LISA,IC,MF)
29350  %    PRINT 85 ,KL
29360  % 85 FORMAT (' ARC ',I3,' SALE ')
29370  %    PRINT 88,(I,F(I),I=1,M)
29380  % 88 FORMAT (4X,'I',4X,'F(I)',//,(2I5))
29390       IF ( KL .EQ. KE ) GO TO 200
29400 %
29410 %  ARBOL
29420 %
29430       IF ( ILC .LE. JUNC ) GO TO 60
29440       KL=-KL
29450       GO TO 61
29460    60 KE=-KE
29470       DEL=-DEL
29480    61 CONTINUE
29490  %    PRINT 95,KE,KL
29500  % 95 FORMAT (' KE,KL',2I7)
29510       CALL TRECHG (KL,KE)
29520 %
29530 %  POTENCIAL
29540 %
29550       IF ( KE.LT. 0) GO TO 69
29560       IT=T(KE)
29570       GO TO 70
```

```
29580     69 IT=0(-KE)
29590     70 CALL ROOT(IT,LISA,LISN,IC,CYC)
29600        DO 80 L=1,IC + 1
29610        I=LISN(L)
29620     80 PI(I) = PI(I) + DEL
29630   %    CALL POUTPU
29640        GO TO 200
29650    999 CONTINUE
29660   %    PRINT 120
29670   %120 FORMAT (' FIN DE PRIMAL',/)
29680        RETURN
29690        END
29700
29710
29720        SUBROUTINE SELECT(IST,KE,DEL,IFIN)
29730        COMMON/A/0(0:120)/B/T(0:120)/C/PB(0:120)/D/PF(0:120)
29740       */F/PR(0:120)/G/M,N/H/PT(0:120)/I/LT(0:120)
29750        COMMON/J/PD(0:120)/K/C(0:120)/L/H(0:120)/M/B(0:120)
29760       */N/CL(0:120)/O/F(0:120)/P/PI(0:120)
29770        COMMON /W/SN,FN
29780        DIMENSION LISA(0:120),LISN(0:120)
29790        INTEGER 0,T,PB,PF,PR,PT,LT,PD,C,H,B,CL,F,PI,FN,SN,D,DEL
29800   %    ************************************************************
29810   %    *                                                          *
29820   %    PROPOSITO : ENCONTRAR UN ARCO QUE NO SATISFACE LAS CONDI-
29830   %    CIONES DE OPTIMALIDAD, USANDO LA REGLA DEL PRIMER NODO MAS
29840   %    NEGATIVO.
29850   %    *                                                          *
29860   %    ************************************************************
29870    %    PRINT 110
29880    %110 FORMAT ( ' SUBROUTINE SELECT')
29890   %
29900   %      INICIA
29910   %
29920        IF(IST.NE.1)GO TO 10
29930        SN=1
29940        FN=N
29950   %
29960   %      ENCUENTRA
29970   %
29980     10 DO 30 I=SN,FN
29990        DEL=0
30000        KE=0
30010        CALL ORIG(I,LISA,LISN,IC)
30020        IF(IC.LE.0) GO TO 30
30030        DO 20 L=1,IC
30040        K=LISA(L)
30050        J=LISN(L)
30060   %    PRINT 130,I,PI(I),J,PI(J),K,H(K)
30070   %130 FORMAT (' I=',I3,',PI(I)=',I3,',J=',I3,',PI(J)=',I3,
30080   %   *        ',K=',I3,',H(K)=',I3)
30090        D=PI(I)+H(K)-PI(J)
30100   %     PRINT 91 ,I,J,K,D,F(K),C(K)
30110   % 91 FORMAT (' I,J,K,D,F(K),C(K)',6I7)
30120        IF(D.EQ.0) GO TO 20
30130        IF(D.GT.0) GO TO 15
30140        IF(C(K).EQ.F(K)) GO TO 20
30150        IF(D.GT.DEL) GO TO 20
30160        DEL=D
30170        KE=K
30180   %     PRINT 92,DEL,KE
30190   % 92 FORMAT (' DEL,KE',2I7)
30200        GO TO 20
30210     15 IF(F(K).EQ.0) GO TO 20
30220        IF(-D.GT.DEL) GO TO 20
30230        DEL=-D
30240        KE=-K
30250   %     PRINT 92,DEL,KE
30260     20 CONTINUE
30270        IF(KE.EQ.0) GO TO 30
30280        IFIN=0
30290        IST=0
```

```
      SN=I+1
      IF ( SN .GT. N ) SN = 1
      FN=N
%      PRINT 93,SN,FN
% 93 FORMAT (' SN,FN',2I7)
%     PRINT 100
      RETURN
   30 CONTINUE
%
%       COMPLETA
%   .
      IF(SN.EQ.1) GO TO 40
      FN=SN-1
      SN=1
%     PRINT 93,SN,FN
      GO TO 10
   40 IFIN=1
%     PRINT 100
%100 FORMAT (' FIN DE  SELECT',/)
      RETURN
      END


      SUBROUTINE TPATH(IS,IT,LISA,LISN,IC,JUNC,NP)
      COMMON/A/O(0:120)/C/PB(0:120)/B/T(0:120)/G/M,N
     *   /B/PD(0:120)/PR/IPRINT
      DIMENSION LISA(0:120),LISN(0:120)
      INTEGER PB,PD,O,T
      INTEGER DDIF
%
%     ***************************************************************
%     *                                                             *
%      PROPOSITO : ENCONTRAR LA RUTA EN EL ARBOL DESDE EL NODO
%     (IS) AL NODO (IT). OBTENIENDO EN LA RUTA LAS LISTAS DE ARCOS
%     (LISA) Y DE NODOS (LISN),EL NUMERO DE ARCOS EN LA RUTA (IC),
%     LA FUNCION (JUNC) ENTRE LAS PARTES HACIA ADELANTE Y REVERSA
%     DE LA RUTA. LOS ARCOS Y NODOS DE LA RUTA SON LISTADOS EN
%     ORDEN REVERSO. SI NO HAY RUTA DESDE IS A IT , NP REGRESA
%     A 1
%     *                                                             *
%     ***************************************************************
%
%     PRINT 20,IS,IT
% 20 FORMAT (' RUTA',/,' NODO INICIAL=',I3,' NODO TERMINAL'
%    *  ,I3)
% INICIALIZA
      II = IS
      IJ=IT
      ICO = 1
      ICN = N -1
      JUNC = 0
      NP = 0
      DDIF = PD(IS) - PD(IT)
%        PRINT 30,DDIF
% 30 FORMAT (' DIFERENCIA,PD(IS)-PD(IT)=',I3)
    8 IF (DDIF) 1,2,3
% RUTA REVERSA DESDE EL NODO IT
    1 K = PB(IJ)
%     PRINT 40,IJ,K
% 40 FORMAT(' REVERSA DESDE EL NODO',I3,' SOBRE EL ARCO',I3)
      IF (K) 4,5,6
    4 IJ = T(-K)
    7 LISA(ICO) =K
%     PRINT 50,IJ
% 50 FORMAT (' AL NODO',I3)
      LISN(ICO) = IJ
```

```
30960          ICO = ICO +1
30970          DDIF = DDIF +1
30980 %        PRINT 30,DDIF
30990          GO TO 8
31000        6 IJ = O(K)
31010          GO TO 7
31020 %  REVERSA DE LA RUTA
31030        3 K=PR(II)
31040 %        PRINT 40,II,K
31050          LISA(ICN) = -K
31060          LISN(ICN) = II
31070        . ICN = ICN -1
31080        ' IF(K) 9,5,10
31090        9 II= T(-K)
31100       11 DDIF = DDIF -1
31110 %        PRINT 50,II
31120 %        PRINT 30,DDIF
31130          GO TO 8
31140       10 II= O(K)
31150          GO TO 11
31160        2 IF (II .EQ. IJ) GO TO 12
31170 % EJECUTA SIMULTANEAMENTE LA RUTA INVERSA
31180          GO TO 1
31190 % COMBINA RUTAS HACIA ADELANTE Y HACIA ATRAS
31200       12 JUNC = ICO-1
31210          IC = ICO -1
31220 %        PRINT 60,IC,ICN
31230 %  60 FORMAT (' IC=',I3,' ICN=',I3)
31240       13 IF (ICN .EQ. N-1) GO TO 14
31250          IC = IC + 1
31260          ICN = ICN +1
31270          LISA(IC) = LISA(ICN)
31280          LISN(IC) = LISN(ICN)
31290          GO TO 13
31300        5 NP = 1
31310       14 CONTINUE
31320 %        PRINT 300
31330 % 300 FORMAT (' FIN DE TPATH',/)
31340          RETURN
31350          END
31360
31370
31380          INTEGER FUNCTION ADC(K)
31390          COMMON /K/C(0:120) /N/CL(0:120) /O/F(0:120)
31400          INTEGER C,CL,F
31410          ADC = 1
31420          IF ( K .LE. 0 ) ADC=0
31430          IF ( K .EQ. 12) ADC=0
31440          RETURN
31450          END
31460
```

4. EJEMPLOS DE APLICACION

3.1 INTRODUCCION

Los ejemplos de aplicación que se han seleccionado para ilustrar el uso y manejo del programa RFCLSG. Son cortos y sencillos, más que nada con fines pedagógicos. Su objetivo es explicar facilmente la organización y estructura de los datos de entrada, como también la interpretación de los resultados. Se resuelven tres problemas de costo mínimo, tres problemas de flujo máximo y dos problemas de flujo a costo mínimo. Un problema de cada uno de ellos se resuelve por los diferentes métodos (siempre que sea posible), para que el usuario pueda ver las diferencias en los tiempos de proceso, los cuales tienen que ver directamente con los costo para el uso de una computadora.

3.2 PROBLEMAS DE COSTO MINIMO (RUTA MAS CORTA)

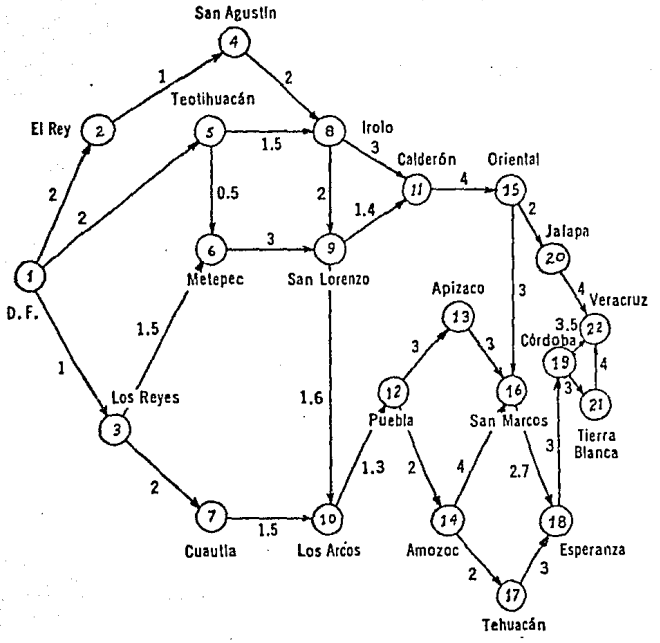
a. Problema de transporte.

Se desea calcular la ruta más rápida desde México a las diferentes ciudades que se encuentran conectadas mediante un sistema vial de ferrocarilles, para lo cual el tiempo que se demora el tren entre una y otra ciudad esta dado en horas; como se puede observar en la figura A-21. Para este caso las horas se transforman en minutos que representan los parámetros costo, asociados a cada arco, en base a los cuales el algoritmo determina el costo mínimo (tiempo mínimo) desde el nodo fuente (México), al resto de nodos (ciudades). Como no se requiere de ningún dato adicional en los nodos, estos se omiten, pero el algoritmo los considera como ceros. Ver cuadros de datos y respuestas que a continuación se presentan. En lo que se refiere a los datos en la lína número 100 se imprime con formato I5, el número clave (NS) que llama a la subrutina principal y resuelve el problema; en la línea 200 se imprime con formato I5, el número del nodo fuente o nodo raíz (SN), aquí no es necesario colocar el número del nodo terminal (TN). No se imprimen los datos correspondien-

tes a los nodos (la subrutina los toma como ceros), los cuales no son necesarios, ya que el algoritmo trabaja solo con parámetros costo correspondientes a cada arco; en la línea - 300 se imprime con formato I5 el número total de nodos de la red (N); la instrucción 400 va en blanco, la misma que indica la terminación de la lectura correspondiente a los datos de los nodos; desde las líneas 500 hasta la 3.500 se imprimen con formato 2I5, el nodo original (I) y el nodo terminal (J) de cada arco y en la última columna se imprime con formato 3F10.0 el tiempo en minutos (COST); note que no aparecen las capacidades mínima (LOWER) y máxima (UPPER) de cada arco las cuales deberían ir en las dos columnas anteriores a la - última (el algoritmo los toma como ceros y no son necesarias para la solución del problema).

Refiriéndose a los resultados, en los parámetros transformados de los nodos, el algoritmo imprime el número de cada nodo I, creando un nodo más que es el auxiliar y el flujo externo fijo en cada nodo B(I), el mismo que es cero en todos los nodos. En los parámetros correspondientes a los arcos - se imprimen ordenadamente en forma creciente, el número del arco K, el nodo origen O(K), el nodo terminal T(K), la capacidad máxima C(K) y el costo H(K), de cada arco. Finalmente en la solución del problema se imprime el número de nodo I; el potencial P(I), que representa el costo acumulado mínimo - (tiempo acumulado mínino) que se necesita para llegar al nodo I partiendo desde el nodo fuente o nodo raíz; el apuntador hacia atrás PB(I), el cual representa el arco que termina en el nodo I y sirve para representar la ruta más corta - entre el nodo fuente al resto de nodos, y el nodo inicial - O(I) y el nodo terminal T(I) de todos los arcos que constituyen la solución óptima. En la figura A-22 se puede observar las rutas más cortas (tiempos mínimos), entre el nodo fuente (México) al resto de nodos (Ciudades), representadas mediante los arcos a doble línea. Los tiempos que se demoran para llegar a cada ciudad, partiendo desde México se determinan -

con el valor de P(I) en cada nodo I, correspondiente a cada Ciudad. Por ejemplo para llegar a Puebla que le corresponde el nodo 12, se demora 348 minutos desde México. Este problema es resuelto mediante el algoritmo Dijkstra.



100	43		
200	1		
300	22		
400			
500	1	2	120.
600	1	3	60.
700	1	5	120.
800	2	4	60.
900	3	6	90.
1000	3	7	120.
1100	4	8	120.
1200	5	8	90.
1300	5	6	30.
1400	6	9	180.
1500	7	10	90.
1600	8	9	120.
1700	8	11	180.
1800	9	10	96.
1900	9	11	84.
2000	10	12	78.
2100	11	15	240.
2200	12	13	180.
2300	12	14	120.
2400	13	16	160.
2500	14	16	240.
2600	14	17	120.
2700	15	16	180.
2800	15	20	120.
2900	16	18	162.
3000	17	18	180.
3100	18	19	180.
3200	19	21	180.
3300	19	22	210.
3400	20	22	240.
3500	21	22	240.
3600			
*			

Figura A-21

Red ferroviaria entre México y otras ciudades.

EL NRO. CLAVE DE ESTA SUBRUTINA ES I 43

PROGRAMA COMI43
DIJKSTRA, COSTO MINIMO CON COSTOS POSITIVOS

EL NODO 1 ES EL NODO RAIZ DE LA RUTA MAS CORTA

REPRESENTACION DE LA RED--

NRO. DE NODOS-- 23 NRO. DE ARCOS-- 31

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0
16	0
17	0
18	0
19	0
20	0
21	0
22	0
23	0

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	O(K)	T(K)	C(K)	H(K)
1	1	2	0	120
2	1	3	0	60
3	1	5	0	120
4	2	4	0	60
5	3	6	0	90
6	3	7	0	120
7	4	8	0	120
8	5	8	0	90
9	5	6	0	30
10	6	9	0	180
11	7	10	0	90
12	8	9	0	120
13	8	11	0	180
14	9	10	0	96
15	9	11	0	84
16	10	12	0	78
17	11	15	0	240
18	12	13	0	180
19	12	14	0	120
20	13	16	0	160
21	14	16	0	240
22	14	17	0	120
23	15	16	0	180
24	15	20	0	120
25	16	18	0	162
26	17	18	0	180
27	18	19	0	180
28	19	21	0	180
29	19	22	0	210
30	20	22	0	240
31	21	22	0	240

LA SOLUCION OPTIMA DEL PROBLEMA ES :

I	P(I)	PB(I)	O(I)	T(I)
1	0	0	0	0
2	120	1	1	2
3	60	2	1	3
4	180	4	2	4
5	120	3	1	5
6	150	5	3	6
7	180	6	3	7
8	210	8	5	8
9	330	10	6	9
10	270	11	7	10
11	390	13	8	11
12	348	16	10	12
13	528	18	12	13
14	468	19	12	14
15	630	17	11	15
16	688	20	13	16
17	588	22	14	17
18	768	26	17	18
19	948	27	18	19
20	750	24	15	20
21	1128	28	19	21
22	990	30	20	22

FIN DEL PROGRAMA RFCLSG

EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES: 0.300000

#ET=3:01.6 PT=0.5 IO=0.3

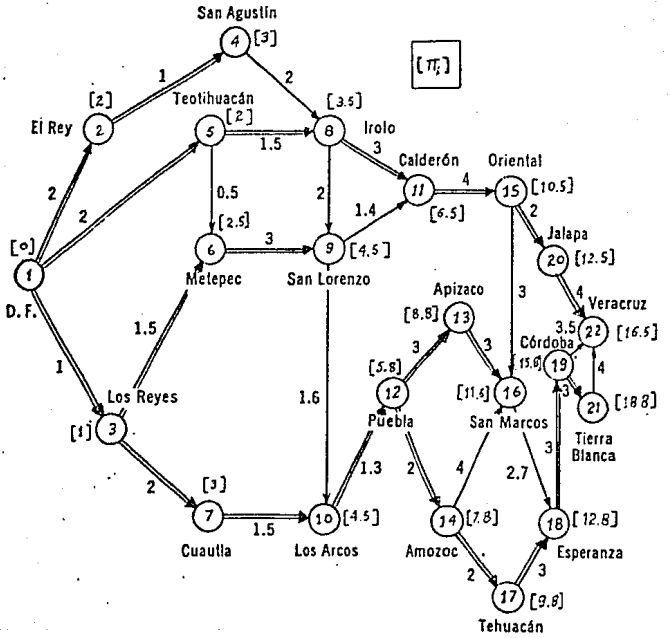


Fig. A-22

b. Problema de reemplazo

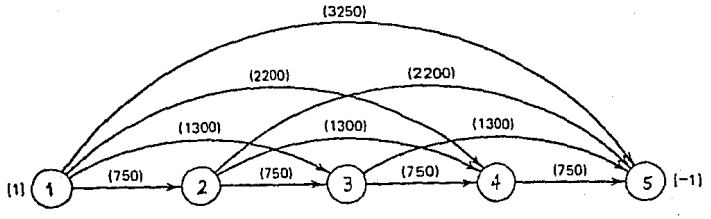
Una compañía constructora tiene como una de sus especialidades, realizar cortes en las capas asfálticas de las calles para luego construir redes de alcantarillado, electricidad o telefónicas. Desea minimizar los costos correspondientes al uso de la sierra cortadora, por el tiempo que dura el contrato que es de un año, para lo cual conoce que el precio de compra de una nueva sierra es de U.S. $1.500 además de acuerdo a sus estadísticas ha establecido los costos de reparación y el valor de rescate, luego de cada periodo de tres meses de uso, mediante la siguiente tabla:

Periodos de uso (3 meses)	Costo de reparación	Valor de rescate
1	300	750
2	600	500
3	875	200
4	1000	25

Considerando que es necesario realizar la repración de la sierra al final de cada período o comprar una nueva, ha establecido la siguiente red de alternativas (Ver figura A-23). Los nodos 1 y 5 representan el inicio y el final del tiempo del contrato y los nodos 2, 3 y 4 representan simultaneamente el inicio y el final de cada período, respectivamente. El costo asociados con el arco (1,3) representa el precio de compra del equipo ($1,500), más el costo de repración al final del primer período ($300), menos el valor de rescate al final del segundo periodo ($500), lo cual da un total de $1,300, y en igual forma para los costos asociados con el resto de arcos.

Como se vió en el ejemplo anterior, la organización y estructura de los datos para este problema son similares, con la única diferencia que en la línea 400 tenemos el número 1 en la primera columna que representa el número del nodo fuente o raíz y el nú-

mero 1 en la segunda columna que representa el flujo externo fijo en el nodo fuente, y en la línea 500 tenemos el número 5 en la primera columna que representa el número del nodo sumidero y el número -1 en la segunda columna es el flujo externo fijo en el nodo sumidero, con esto se supone que una unidad de flujo atraviesa a lo largo de la red desde el nodo fuente hasta el nodo sumidero. A continuación se presentan los datos de entrada y los resultados del problema, obtenidos por los diferentes métodos de ruta más corta (COMI43, COMI45, COMI46 y COMI47). En los cuales en cada nodo I se tiene los correspondientes potenciales P(I) que representa los costos acumulados después de cada período de uso de la sierra cortadora. La solución óptima es fácil de observarla ya que partiendo del nodo 5, con un costo acumulado mínimo de $2,600, se regresa por medio de su arco apuntador hacia atrás 9(3,5) para llegar al nodo 3 y de este por medio del arco apuntador 2(1,3) para llegar al nodo 1 que es el inicio del contrato. En la figura A-24 se puede observar la solución óptima representada mediante arcos a doble línea.




```
100     43
200      1
300      5
400      1          1
500      5         -1
600
700      1     2                    750.
800      1     3                   1300.
900      1     4                   2200.
1000      1    5                   3250.
1100      2    3                    750.
1200      2    4                   1300.
1300      2    5                   2200.
1400      3    4                    750.
1500      3    5                   1300.
1600      4    5                    750.
1700
```

Figura A-23

Red de alternativas

EL NRO. CLAVE DE ESTA SUBRUTINA ES : 43

PROGRAMA COMI43
DIJKSTRA, COSTO MINIMO CON COSTOS POSITIVOS

EL NODO 1 ES EL NODO RAIZ DE LA RUTA MAS CORTA

REPRESENTACION DE LA RED--

NRO. DE NODOS-- 6 NRO. DE ARCOS-- 10

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	1
2	0
3	0
4	0
5	-1
6	0

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	O(K)	T(K)	C(K)	H(K)
1	1	2	0	750
2	1	3	0	1300
3	1	4	0	2200
4	1	5	0	3250
5	2	3	0	750
6	2	4	0	1300
7	2	5	0	2200
8	3	4	0	750
9	3	5	0	1300
10	4	5	0	750

LA SOLUCION OPTIMA DEL PROBLEMA ES :

I	P(I)	PB(I)	O(I)	T(I)
1	0	0	0	0
2	750	1	1	2
3	1300	2	1	3
4	2050	6	2	4
5	2600	9	3	5

FIN DEL PROGRAMA RFCLSG

EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES: 0.133333

#ET=1:19.4 PT=0.3 IO=0.2

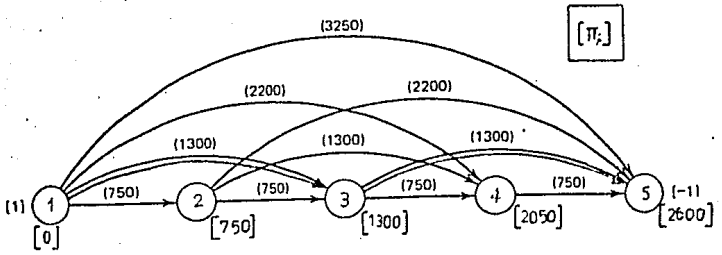


Fig. A-24

EL NRO. CLAVE DE ESTA SUBRUTINA ES : 45

PROGRAMA COMI45
COMBINADO DE RUTA MAS CORTA
CON UN ARBOL INICIAL DADO POR EL ALGORITHO DIJKSTRA

EL NODO RAIZ ES 1

REPRESENTACION DE LA RED--

NRO. DE NODOS-- 6 NRO. DE ARCOS-- 10

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	1
2	0
3	0
4	0
5	-1
6	0

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	O(K)	T(K)	C(K)	H(K)
1	1	2	0	750
2	1	3	0	1300
3	1	4	0	2200
4	1	5	0	3250
5	2	3	0	750
6	2	4	0	1300
7	2	5	0	2200
8	3	4	0	750
9	3	5	0	1300
10	4	5	0	750

RUTA MAS CORTA DESDE EL NODO RAIZ 1
A TODOS LOS DEMAS NODOS

I	PI(I)	PB(I)	O(I)	T(I)
1	0	0	0	0
2	750	1	1	2
3	1300	2	1	3
4	2050	6	2	4
5	2600	9	3	5

FIN DEL PROGRAMA RFCLSG

EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES: 0.116667

#ET=1:20.6 PT=0.3 IO=0.2

EL NRO. CLAVE DE ESTA SUBRUTINA ES : 46

PROGRAMA COMI46
NO BASICO DE RUTA MAS CORTA

EL NODO RAIZ ES 1

REPRESENTACION DE LA RED--

NRO. DE NODOS-- 6 NRO. DE ARCOS-- 10

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	1
2	0
3	0
4	0
5	-1
6	0

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	O(K)	T(K)	C(K)	H(K)
1	1	2	0	750
2	1	3	0	1300
3	1	4	0	2200
4	1	5	0	3250
5	2	3	0	750
6	2	4	0	1300
7	2	5	0	2200
8	3	4	0	750
9	3	5	0	1300
10	4	5	0	750

RUTA MAS CORTA DESDE EL NODO RAIZ 1 A TODOS LOS OTROS NODOS

I	PI(I)	PB(I)	O(I)	T(I)
1	0	0	0	0
2	750	1	1	2
3	1300	2	1	3
4	2050	6	2	4
5	2600	9	3	5

FIN DEL PROGRAMA RFCLSO

EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES: 0.133333

$ET=1:18.0 PT=0.3 IO=0.2

EL NRO. CLAVE DE ESTA SUBRUTINA ES : 47

PROGRAMA COMI47
DUAL DE RUTA MAS CORTA

REPRESENTACION DE LA RED--

NRO. DE NODOS-- 6 NRO. DE ARCOS-- 10

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	1
2	0
3	0
4	0
5	-1
6	0

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	O(K)	T(K)	C(K)	H(K)
1	1	2	0	750
2	1	3	0	1300
3	1	4	0	2200
4	1	5	0	3250
5	2	3	0	750
6	2	4	0	1300
7	2	5	0	2200
8	3	4	0	750
9	3	5	0	1300
10	4	5	0	750

RUTA MAS CORTA DESDE EL NODO RAIZ 1
AL RESTO DE NODOS CYC= 0 J= 5

I	PI(I)	PB(I)	O(I)	T(I)
1	0	0	0	0
2	750	1	1	2
3	1300	2	1	3
4	2050	6	2	4
5	2600	9	3	5

RUTA MAS CORTA DESDE EL NODO RAIZ 1
AL RESTO DE NODOS CYC= 0 J= 5

I	PI(I)	PB(I)	O(I)	T(I)
1	0	0	0	0
2	750	1	1	2
3	1300	2	1	3
4	2050	6	2	4
5	2600	9	3	5

FIN DEL PROGRAMA RFCLSG

EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES: 0.150000

$ET=1:36.1 PT=0.3 IO=0.2

c. Problema de ruta crítica

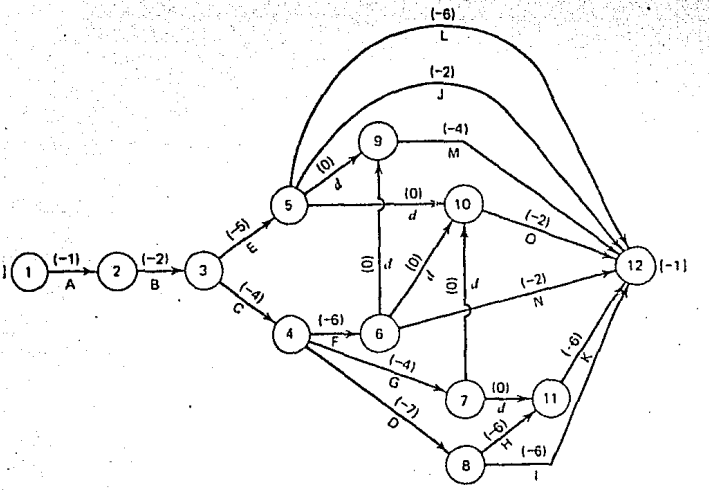
Una compañía constructora obtiene un contrato para construir un nuevo campo de deportes en un colegio. Debido a las multas que debe pagar la compañía si no cumple con las fechas estipuladas en el contrato para entregar la obra, el departamento de planificación ha determinado las partes del proyecto que se consideran más importantes; considerando las actividades mutuamente excluyentes y las actividades que dependen unas de otras y que necesariamente se anteceden o preceden durante la construcción; se obtiene la siguiente información:

Subpto.	Descripción	Días requeridos	Subproyectos que anteceden
A	Medición del terreno	1	-
B	Fundaciones	2	A
C	Levantamiento de paredes	4	B
D	Construcción del techo	7	C
E	Colocación del piso de madera	5	B
F	Acabados interiores de paredes	6	C
G	Instalaciones de plomeria	4	C
H	Instalaciones eléctricas	4	D
I	Montaje de arcos de basketball	1	D
J	Pintar los limites	2	E
K	Instalaciones de sistemas de calefacción/refrigeración	6	G,H
L	Construcción de la pista interior	6	E
M	Construcción de asientos	4	E,F
N	Montaje del control electrónico	2	F
O	Construcción del bar	2	E,F,G

Como se puede ver en la figura A-25, los arcos representan los subproyectos. Los arcos d con costo cero son auxiliares que remueven las ambiguedades en la gráfica. Los costos en los arcos son todos negativos para transformar el problema de ruta más corta, a ruta más larga (camino crítico). Los nodos representan simultaneamente el inicio y el final de los subproyectos y los nodos 1 y 14 represental el inicio y el final del proyecto, respectivamente. El proyecto no tolera ningún retrazo de tiempo en los

subproyectos, ya que demoraría la entrega de la obra.

A continuación se presentan los datos de entrada y los resultados del problema, resuelto mediante el algoritmo no básico de ruta más corta COMI46. En la figura A-26 se puede observar la solución óptima representada mediante arcos a doble linea.




```
100     45
110      1
120     12
130      1          1
140     12         -1
150
160      1      2                -1.
170      2      3                -2.
180      3      4                -4.
190      3      5                -5.
200      4      6                -6.
210      4      7                -4.
220      4      8                -7.
230      5      9                 0.
240      5     10                 0.
250      5     12                -6.
260      5     12                -2.
270      6      9                 0.
280      6     10                 0.
290      6     12                -2.
300      7     10                 0.
310      7     11                 0.
320      8     11                -6.
330      8     12                -6.
340      9     12                -4.
350     10     12                -2.
360     11     12                -6.
370
```

Figura A-25

Red de interconección de los subproyectos

3.3 PROBLEMA DE FLUJO MAXIMO

a. Una compañía desea incrementar su producción de encuadernación de libros, para lo cual establece nueve pasos en el proceso de encuadernación de un libro: (1) abrir las cajas, (2) cotejar las páginas (3) aparejar las hojas, (4) pegar las hojas por la parte de atrás (5) colocar las pastas, (6) Grabar las pastas con los títulos, (7) colocar los libros en las cajas, (9) Pegar -

EL NRO. CLAVE DE ESTA SUBRUTINA ES : 45

PROGRAMA COM145
COMBINADO DE RUTA MAS CORTA
CON UN ARBOL INICIAL DADO POR EL ALGORITMO DIJKSTRA

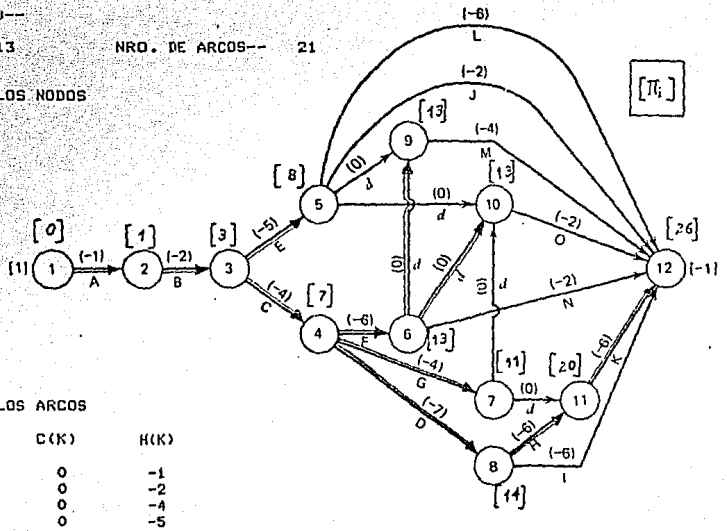
EL NODO RAIZ ES 1

REPRESENTACION DE LA RED--

NRO. DE NODOS-- 13 NRO. DE ARCOS-- 21

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	1
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	-1
13	0



PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	O(K)	T(K)	C(K)	H(K)
1	1	2	0	-1
2	2	3	0	-2
3	3	4	0	-4
4	3	5	0	-5
5	4	6	0	-6
6	4	7	0	-4
7	4	8	0	-7
8	5	9	0	0
9	5	10	0	0
10	5	12	0	-6
11	5	12	0	-2
12	6	9	0	0
13	6	10	0	0
14	6	12	0	-2
15	7	10	0	0
16	7	11	0	0
17	8	11	0	-6
18	8	12	0	-6
19	9	12	0	-4
20	10	12	0	-2
21	11	12	0	-6

RUTA MAS CORTA DESDE EL NODO RAIZ 1
A TODOS LOS DEMAS NODOS

I	PI(I)	PB(I)	O(I)	T(I)
1	0	0	0	0
2	-1	1	1	2
3	-3	2	2	3
4	-7	3	3	4
5	-8	4	3	5
6	-13	5	4	6
7	-11	6	4	7
8	-14	7	4	8
9	-13	12	6	9
10	-13	13	6	10
11	-20	17	8	11
12	-26	21	11	12

FIN DEL PROGRAMA RFCLSG

EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES: 0.200000

#ET=2107.5 PT=0.3 IO=0.2

Fig. A-26

las etiquetas de correo y (9) cargar en carretones.

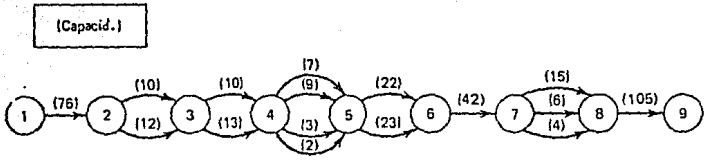
Pasos	Número de estaciones	Capacidad de estación (Libros/día)
1	1	76
2	2	10, 12
3	2	10, 13
4	4	7,9,3,2
5	2	22, 23
6	1	42
7	3	15,6,4
8	2	10,15
9	1	105

Usando la información de arriba el supervisor tiene que determinar el máximo número de libros que pueden ser encuadernados y hacer recomendaciones para incrementar la producción total al menor costo posible. La figura A-27 establece la red que el supervisor usa para su propósito. Los nodos representan los nueve pasos en la encuadernación de un libro y los arcos de la red son las estaciones asociadas con sus nodos origen. Las capacidades de los arcos son el número de libros que pueden hacerse en cada estación por día.

Refiriéndose a los datos de entrada, en la línea número 100 se imprime con formato I5, el número clave (NS) que llama a la subrutina principal que resuelve el problema; en la línea 200 se imprime con formato I5, el número de nodos (N); la línea 300 va en blanco; de la 400 a la 1.900 se imprime con formato 2I5, el nodo original (I) y el nodo terminal (J) de cada arco; con formato 3F10.0 se definen 3 campos de diez espacios para la capacidad mínima (LOWER) que no aparece, la capacidad máxima (UPPER) que si se imprime, ya que con estos datos trabaja el algoritmo y el costo (COST) que tampoco aparece; la línea 2,000 en blanco, y en línea 2,100 se imprime con formato 4I5, el nodo fuente o raíz (SN), el nodo sumidero (TN), el flujo requerido (VR) y el flujo inicial (IFLOW).

Respecto a los resultados, aparecen los parámetros transfor-

mados de los nodos con los números de cada nodo (NODO I) y el - flujo externo fijo (B(I)) que son todos ceros ya que estos datos no son necesarios para la solución del problema; en los paráme-- tros transofmrados de los arcos, constan las capacidades máximas (C(K)), con las que trabaja el algoritmo para obtener el flujo - náximo; obtiene el flujo máximo posible que es de 21 libros; los apuntadores hacia atrás PB(I) que definen el árbol, y finalmente un cuadro de datos adicionales en el que consta el flujo que pasa por cada arco, A continuación se puede observar los datos de entrada y los resultados. El problema es resuelto con el algorit mo no básico y con el algoritmo básico.




```
100    54
200     9
300
400     1     2          76.
500     2     3          10.
600     2     3          12.
700     3     4          10.
800     3     4          13.
900     4     5           7.
1000    4     5           9.
1100    4     5           3.
1200    4     5           2.
1300    5     6          22.
1400    5     6          23.
1500    6     7          42.
1600    7     8          15.
1700    7     8           6.
1800    7     8           4.
1900    8     9         105.
2000
2100    1     9   100     0
2200
$
```

Figura A-27

Proceso de encuadernación de un libro.

b. El Alcade de una ciudad industrial desea conocer la máxima - cantidad de vehículos por hora que pueden transitar desde el cen tro de CHACLOGGO a la pequeña ciudad de OWDDA. La red de carrete ras que se pueden utilizar entre estas 2 ciudades esta dada en - la figura A-28. Los arcos representan las vías con el máximo nú- mero de vehículos por hora (en cientos de vehículos), y los no-- dos son las intersecciones de las vías.

Tanto la entrada de datos como los resultados estan dados a

EL NRO. CLAVE DE ESTA SUBRUTINA ES : 54

PROGRAMA FLMA54
NO BASICO DE FLUJO MAXIMO

REPRESENTACION DE LA RED--

NRO. DE NODOS-- 10 NRO. DE ARCOS-- 16

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	O(K)	T(K)	C(K)	H(K)
1	1	2	76	0
2	2	3	10	0
3	2	3	12	0
4	3	4	10	0
5	3	4	13	0
6	4	5	7	0
7	4	5	9	0
8	4	5	3	0
9	4	5	2	0
10	5	6	22	0
11	5	6	23	0
12	6	7	42	0
13	7	8	15	0
14	7	8	6	0
15	7	8	4	0
16	8	9	105	0

SOLUCION DEL MAX. FLUJO. FLUJO REQUERIDO 100

DESDE EL NODO FUENTE 1 AL NODO SUMIDERO 9

NO EXISTE RUTA ADICIONAL
FLUJO MAXIMO POSIBLE = 21

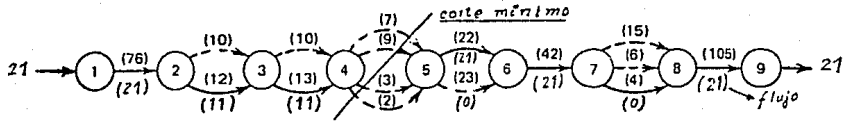
NODO I	P(I)	PB(I)
1	0	0
2	0	1
3	0	3
4	0	5
5	0	9
6	0	10
7	0	12
8	0	14
9	0	16

ARC K	O(K)	T(K)	FLUJO	CAPACIDAD	COST/UNI.	COST.FLUJO
1	1	2	21	76	0	0
2	2	3	10	10	0	0
3	2	3	11	12	0	0
4	3	4	10	10	0	0
5	3	4	11	13	0	0
6	4	5	7	7	0	0
7	4	5	9	9	0	0
8	4	5	3	3	0	0
9	4	5	2	2	0	0
10	5	6	21	22	0	0
11	5	6	0	23	0	0
12	6	7	21	42	0	0
13	7	8	15	15	0	0
14	7	8	6	6	0	0
15	7	8	0	4	0	0
16	8	9	21	105	0	0

COSTO TOTAL = 0

FIN DEL PROGRAMA RFCLSG

EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES: 0.283333



EL NRO. CLAVE DE ESTA SUBRUTINA ES : 55

PROGRAMA FLHA55
BASICO DE FLUJO MAXIMO

REPRESENTACION DE LA RED--

NRO. DE NODOS-- 10 NRO. DE ARCOS-- 16

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	O(K)	T(K)	C(K)	H(K)
1	1	2	76	0
2	2	3	10	0
3	2	3	12	0
4	3	4	10	0
5	3	4	13	0
6	4	5	7	0
7	4	5	9	0
8	4	5	3	0
9	4	5	2	0
10	5	6	22	0
11	5	6	23	0
12	6	7	42	0
13	7	8	15	0
14	7	8	6	0
15	7	8	- 4	0
16	8	9	105	0

SOLUCION DEL MAX. FLUJO. FLUJO REQUERIDO 100

DESDE EL NODO FUENTE 1 AL NODO SUMIDERO 9

TODOS LOS FLUJOS INICIALES SON CERO
NO EXISTE RUTA ADICIONAL
MAXIMO FLUJO POSIBLE = 21

NODO I	P(I)	PB(I)
1	0	0
2	0	1
3	0	3
4	0	5
5	0	9
6	0	10
7	0	12
8	0	14
9	0	16

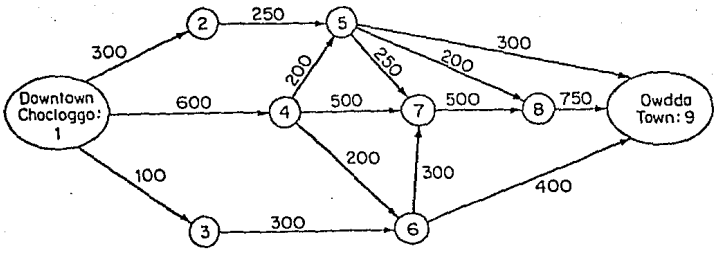
ARC K	O(K)	T(K)	FLUJO	CAPACIDAD	COST/UNI.	COST.FLUJO
1	1	2	21	76	0	0
2	2	3	10	10	0	0
3	2	3	11	12	0	0
4	3	4	10	10	0	0
5	3	4	11	13	0	0
6	4	5	7	7	0	0
7	4	5	9	9	0	0
8	4	5	3	3	0	0
9	4	5	2	2	0	0
10	5	6	21	22	0	0
11	5	6	0	23	0	0
12	6	7	21	42	0	0
13	7	8	15	15	0	0
14	7	8	6	6	0	0
15	7	8	0	4	0	0
16	8	9	21	105	0	0

COSTO TOTAL = 0

FIN DEL PROGRAMA RECLSG

EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES: 0.283333

continuación y tienen la misma organización, estructura y significado, tal como, el problema anterior. El problema se resuelve con el algoritmo no básico de flujo máximo.




```
100     54
200      9
300
400      1     2              300.
500      1     3              100.
600      1     4              600.
700      2     5              250.
800      3     6              300.
900      4     5              200.
1000     4     6               200.
1100     4     7               500.
1200     5     7               250.
1300     5     8               200.
1400     5     9               300.
1500     6     7               300.
1600     6     9               400.
1700     7     8               500.
1800     8     9               750.
1900
2000     1     9 1500     0
2100
2200
$
```

Figura A-28
Red Vial

c. Un organismo de servicio contra las plagas, ha determinado - que la migración anual de una plaga contra el trigo comienza al inicio del año. En base a datos obtenidos se ha logrado establecer una red de caminos que sigue la plaga. La representación esquemática de las rutas de la migración se dan en la figura A-29. Los nodos son los puntos de unión de las rutas de migración, -- mientras los nodos 1 y 18 son los puntos de inicio y término de la migración, respectivamente. En los arcos se señala el número máximo de insectos (en miles) que usarán esas rutas. Estas capacidades se basan en datos históricos. El servicio contra plagas ha formulado este problema como un problema de flujo máximo, que a su vez obtiene la solución del corte mínimo, el cual es un conjunto de rutas de migración de capacidad mínima que son removi--

EL NRO. CLAVE DE ESTA SUBRUTINA ES : 54

PROGRAMA FLMA54
NO BASICO DE FLUJO MAXIMO

REPRESENTACION DE LA RED--

NRO. DE NODOS-- 10 NRO. DE ARCOS-- 15

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	O(K)	T(K)	C(K)	H(K)
1	1	2	300	0
2	1	3	100	0
3	1	4	600	0
4	2	5	250	0
5	3	6	300	0
6	4	5	200	0
7	4	6	200	0
8	4	7	500	0
9	5	7	250	0
10	5	8	200	0
11	5	9	300	0
12	6	7	300	0
13	6	9	400	0
14	7	8	500	0
15	8	9	750	0

SOLUCION DEL MAX. FLUJO. FLUJO REQUERIDO 1500

DESDE EL NODO FUENTE 1 AL NODO SUMIDERO 9

NO EXISTE RUTA ADICIONAL
FLUJO MAXIMO POSIBLE = 950

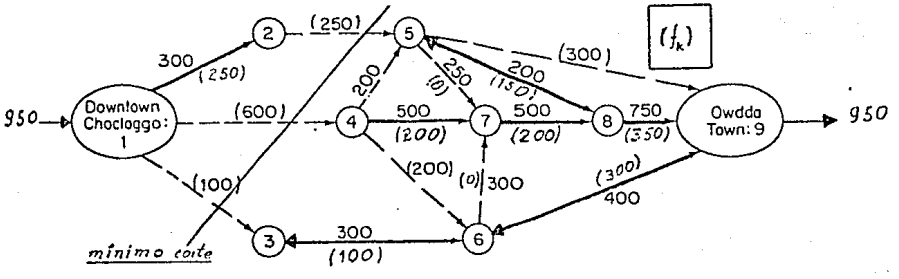
NODO I	P(I)	PB(I)
1	0	0
2	0	1
3	0	-5
4	0	3
5	0	-10
6	0	-13
7	0	8
8	0	14
9	0	15

ARC K	O(K)	T(K)	FLUJO	CAPACIDAD	COST/UNI.	COST.FLUJO
1	1	2	250	300	0	0
2	1	3	100	100	0	0
3	1	4	600	600	0	0
4	2	5	250	250	0	0
5	3	6	100	300	0	0
6	4	5	200	200	0	0
7	4	6	200	200	0	0
8	4	7	200	500	0	0
9	5	7	0	250	0	0
10	5	8	150	200	0	0
11	5	9	300	300	0	0
12	6	7	0	300	0	0
13	6	9	300	400	0	0
14	7	8	200	500	0	0
15	8	9	350	750	0	0

COSTO TOTAL = 0

FIN DEL PROGRAMA RFCLSG

EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES: 0.266667



das desde la red de la figura A-29, impidiendo que lleguen al nodo 18.

Todas las rutas de migración en este corte mínimo recibirán una gran cantidad de insecticida rociado desde un avión. Los datos de entrada y los resultados se presentan a continuación. La figura A-30 presenta la solución del corte mínimo, que lo constituyen los arcos en líneas discontinuas y cuyos flujos suman 24, igual al máximo flujo posible en la red.

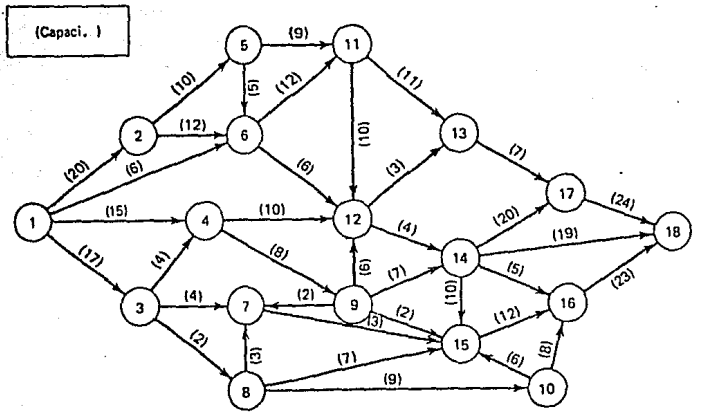


Figura A-29

Red de rutas de migración

```
100     55
200     18
300
400      1     2            20.
500      1     3            17.
600      1     4            15.
700      1     6             6.
800      2     5            10.
900      2     6            12.
1000     3     4             4.
1100     3     7             4.
1200     3     8             2.
1300     4    12            10.
1400     4     9             8.
1500     5     6             5.
1600     5    11             9.
1700     6    11            12.
1800     6    12             6.
1900     7    15             3.
2000     8     7             3.
2100     8    15             7.
2200     8    10             9.
2205     9     7             2.
2300     9    12             6.
2400     9    14             7.
2500     9    15             2.
2600    10    15             6.
2700    10    16             8.
2800    11    12            10.
2900    11    13            11.
3000    12    13             3.
3100    12    14             4.
3200    13    17             7.
3300    14    15            10.
3400    14    16             5.
3500    14    17            20.
3600    14    18            19.
3700    15    16            12.
3800    16    18            23.
3900    17    18            24.
4000
4100     1    18    60      0
4200
```

3.4 PROBLEMA DE FLUJO A COSTO MINIMO

a. Una Compañía suministra empleados de acuerdo a la demanda de varios establecimientos de negocios. La estimación de la demanda de empleados para los próximos cuatro meses son:

mes	1	2	3	4
Demanda	22	15	32	8

Los costos por colocar un empleado al inicio de cada mes y

EL NRO. CLAVE DE ESTA SUBRUTINA ES : 55

PROGRAMA FLMA55
BASICO DE FLUJO MAXIMO

REPRESENTACION DE LA RED--

NRO. DE NODOS-- 19 NRO. DE ARCOS-- 37

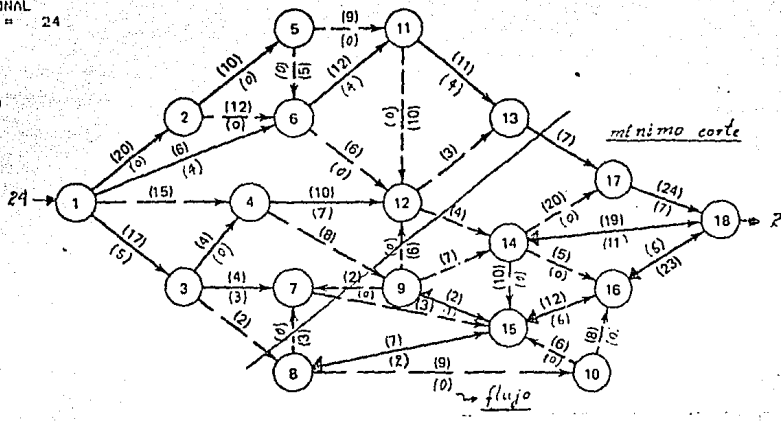
PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0
16	0
17	0
18	0
19	0

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	O(K)	T(K)	C(K)	H(K)
1	1	2	20	0
2	1	3	17	0
3	1	4	15	0
4	1	6	6	0
5	2	5	10	0
6	2	6	12	0
7	3	4	4	0
8	3	7	4	0
9	3	8	2	0
10	4	12	10	0
11	4	9	8	0
12	5	6	5	0
13	5	11	9	0
14	6	11	12	0
15	6	12	6	0
16	7	15	3	0
17	8	7	3	0
18	8	15	7	0
19	8	10	9	0
20	9	7	2	0
21	9	12	6	0
22	9	14	7	0
23	9	15	2	0
24	10	15	6	0
25	10	16	8	0
26	11	12	10	0
27	11	13	11	0
28	12	13	3	0
29	12	14	4	0
30	13	17	7	0
31	14	15	10	0
32	14	16	5	0
33	14	17	20	0
34	14	18	19	0
35	15	16	12	0
36	16	18	23	0
37	17	18	24	0

```
   SOLUCION DEL MAX. FLUJO. FLUJO REQUERIDO  60
DESDE EL NODO FUENTE  1 AL NODO SUMIDERO 18

TODOS LOS FLUJOS INICIALES SON CERO
        NO EXISTE RUTA ADICIONAL
        MAXIMO FLUJO POSIBLE =   24
```



NODO I	P(I)	PB(I)
1	0	0
2	0	1
3	0	2
4	0	7
5	0	5
6	0	4
7	0	8
8	0	-18
9	0	-23
10	0	19
11	0	14
12	0	10
13	0	27
14	0	-34
15	0	-35
16	0	-36
17	0	30
18	0	37

ARC K	O(K)	T(K)	FLUJO	CAPACIDAD	COST/UNI.	COST.FLUJO
1	1	2	0	20	0	0
2	1	3	5	17	0	0
3	1	4	15	15	0	0
4	1	6	4	6	0	0
5	2	5	0	10	0	0
6	2	6	0	12	0	0
7	3	4	0	4	0	0
8	3	7	3	4	0	0
9	3	8	2	2	0	0
10	4	12	7	10	0	0
11	4	9	8	8	0	0
12	5	6	0	5	0	0
13	5	11	0	9	0	0
14	6	11	4	12	0	0
15	6	12	0	6	0	0
16	7	15	3	3	0	0
17	8	7	0	3	0	0
18	8	15	2	7	0	0
19	8	10	0	9	0	0
20	9	7	0	2	0	0
21	9	12	0	6	0	0
22	9	14	7	7	0	0
23	9	15	1	2	0	0
24	10	15	0	6	0	0
25	10	16	0	8	0	0
26	11	12	0	10	0	0
27	11	13	4	11	0	0
28	12	13	3	3	0	0
29	12	14	4	4	0	0
30	13	17	7	7	0	0
31	14	15	0	10	0	0
32	14	16	0	5	0	0
33	14	17	0	20	0	0
34	14	18	11	19	0	0
35	15	16	6	12	0	0
36	16	18	6	23	0	0
37	17	18	7	24	0	0

```
                    COSTO TOTAL =         0

FIN DEL PROGRAMA RFCLSG

EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES:  0.483333

#ET=4:25.4 PT=0.6 IO=0.2
```

Fig. A-30

los costos de indemnización por salir un empleado al final de cada mes son:

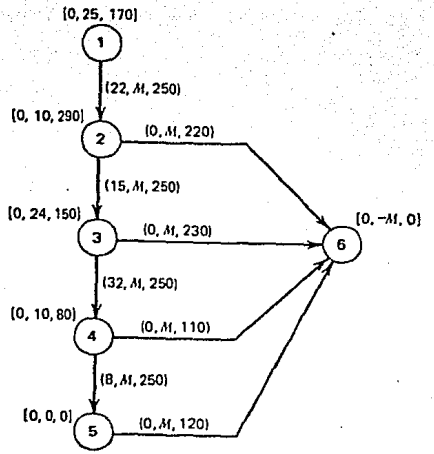
mes	1	2	3	4
Dolar/entrar	170	290	150	80
Dolar/salir	220	230	110	120

El sueldo mensual para todos los empleados es de $250 al mes. La estimación de obtener nuevos empleados en los próximos cuatro meses son: 25,10,24 y 10, respectivamente. Se asume que todos los empleados entran o salen durante este período. La compañía desea determinar la política optima de entradas y salidas de empleados al inicio y al final de cada mes, tal que su costo total de suministro de empleados sea mínimo. Para lo que ha formulado un problema de redes que se presenta en la figura A-31, en la cual los nodos 1 y 4 representan el inicio y el final de cada mes. Los flujos de holgura externos ofrecen la posibilidad de emplearse en estos puntos en el tiempo. Los flujos en los arcos (1,2), (2,3), (3,4) y (3,5) indican el número de empleados durante los 4 meses. Los límites superiores aseguran que los requerimientos de demanda sean satisfechos. Los flujos en los arcos que entran en el nodo 6 representan los empleados que salen al final de cada mes.

Respecto a los datos de entrada, en la línea 100, se imprime con formato I5, el número de clave de la subrutina que resuelve el problema; en la línea 200 se imprime con formato I5, el número total de nodos; de la 300 a la 800 se imprimen con formato I5, en la primera columna el número de cada nodo I y en las 3 columnas siguientes se imprimen con formato 3F10.0 el flujo externo fijo (BF), el flujo externo de holgura (BS), y el costo de flujo de holgura por cada unidad (CS) de cada nodo, respectivamente, la línea 900 va en blanco; de la 1.000 a la 1,700 se imprimen con formato 2I5, en las 2 primeras columnas los números del nodo origen (I) y el nodo terminal (J) de cada arco, y en las 3 siguientes se imprimen con formato 3F10.0, la capacidad mínima (LOWER), la capacidad máxima (UPPER) y el costo (COST) de cada arco, respectivamente.

Dentro de los resultados tenemos B(I), el cual es el flujo ex-

terno fijo en cada nodo, P(I) es el costo unitario acumulado en el nodo I, en este caso se interpreta como el costo que representa al salir un empleado al final del mes; PB(I) es el arco apuntador hacia atrás, que llega a cada nodo y constituye el árbol básico óptimo, finalmente en la última tabla constan entre otros resultados el flujo en cada arco, la capacidad máxima tranformada, el costo unitario y el costo del flujo. Cabe anotar que el algoritmo trabaja en base a los datos transformados, y para representar la red en base a la red original es necesario reestructurar los resultados que presenta el programa. Por ejemplo entre los nodos 1 y 2 no hay flujo, sin embaro entrean 22 unidades (empleados) que se compensan con -22, como flujo externo fijo en el nodo, luego es conveniente transformar el flujo externo fijo -22, en flujo que sale del nodo 1 al nodo 2; en el nodo 2, salen 7 hacia el nodo 6 y los 15 sobrantes necesariamente tienen que ir al nodo 3 y asi sucesivamente. Finalmente en la figura A-32 se puede observar la red reestructurada en base a los resultados, en la cual en el primer mes deben entrar 22 unidades (empleados), al final del primer mes deben salir 7, y los 15 restantes continúan en el segundo mes, al final de este entran 17 empleados, más los 15 anteriores, quedan 32 durante el tercer mes, al final de este salen 24 y quedan 8 para el último mes, y al final de este salen los 8, con lo cual se obtiene los empleados que deben entrar y salir a lo largo de los 4 meses, para que los costos sean mínimos. El costo total que aparece en los resultados, lo constituye solo la cantidad correspondiente a los empleados que salen, faltando agregar el costo de los que entran, más los salarios mensuales de cada uno durante los meses que permanecen trabajando indistintamente. Este problema se resuelve con los algoritmos primal básico y primal no básico.




```
100    65
200     6
300     1          0.          25.         170.
400     2          0.          10.         290.
500     3          0.          24.         150.
600     4          0.          10.          80.
700     5          0.           0.           0.
800     6          0.        -200.           0.
900
1000    1    2        22.        200.        250.
1100    2    3        15.        200.        250.
1200    3    4        32.        200.        250.
1300    4    5         8.        200.        250.
1400    2    6         0.        200.        220.
1500    3    6         0.        200.        230.
1600    4    6         0.        200.        110.
1700    5    6         0.        200.        120.
1800
*
```

Figura A-31

Red de empleados que entran y salen temporalmente.

b. Se supone una Empresa exportadora de cereales, con diferentes oficinas en varias partes del mundo. Recientes pérdidas en las cosechas en la URSS, han aumentado la oportunidad para exportar trigo a este país, solamente por los próximos cinco meses. Debido a la gran competencia del mercado, mes a mes los precios cambian y también los espacios disponibles para el almacenamiento del trigo.

Los administradores de la compañía han obtenido la siguiente información:

EL NRO. CLAVE DE ESTA SUBRUTINA ES : 65

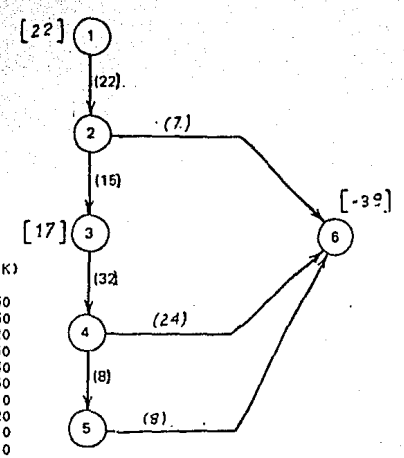
PROGRAMA FCMI65
PRIMAL NO BASICO PARA FLUJO A COSTO MINIMO

REPRESENTACION DE LA RED--

NRO. DE NODOS-- 7 NRO. DE ARCOS-- 13

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	-22
2	7
3	-17
4	24
5	8
6	0
7	0



PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	O(K)	T(K)	C(K)	H(K)
1	1	2	178	250
2	2	3	185	250
3	2	6	200	220
4	3	4	168	250
5	3	6	200	230
6	4	5	192	250
7	4	6	200	110
8	5	6	200	120
9	6	7	200	0
10	7	1	25	0
11	7	2	10	0
12	7	3	24	0
13	7	4	10	0

SOLUCION OPTIMA DEL PROBLEMA

NODO I	P(I)	PB(I)
1	0	10
2	-220	-3
3	0	12
4	-110	-7
5	-120	-8
6	0	-9
7	0	0

ARC K	O(K)	T(K)	FLUJO	CAPACIDAD	COST/UNI.	COST.FLUJO
1	1	2	0	178	250	0
2	2	3	0	185	250	0
3	2	6	7	200	220	1540
4	3	4	0	168	250	0
5	3	6	0	200	230	0
6	4	5	0	192	250	0
7	4	6	24	200	110	2640
8	5	6	8	200	120	960
9	6	7	39	200	0	0
10	7	1	22	25	0	0
11	7	2	0	10	0	0
12	7	3	17	24	0	0
13	7	4	0	10	0	0

COSTO TOTAL = 5140

FIN DEL PROGRAMA RFCLSG

EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES: 0.233333

Fig. A-32

$ET=2:07.5 PT=0.4 IO=0.2

EL NRO. CLAVE DE ESTA SUBRUTINA ES : 66

PROGRAMA FCMI66
PRIMAL BASICO PARA FLUJO A COSTO MINIMO

REPRESENTACION DE LA RED--

NRO. DE NODOS-- 7 NRO. DE ARCOS-- 13

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	-22
2	7
3	-17
4	24
5	8
6	0
7	0

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	O(K)	T(K)	C(K)	H(K)
1	1	2	178	250
2	2	3	185	250
3	2	6	200	220
4	3	4	168	250
5	3	6	200	230
6	4	5	192	250
7	4	6	200	110
8	5	6	200	120
9	6	7	200	0
10	7	1	25	0
11	7	2	10	0
12	7	3	24	0
13	7	4	10	0

FORMACION DE LA RED CON ARCOS ARTIFICIALES

NRO. DE NODOS-- 7 NRO. DE ARCOS-- 19

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	-22
2	7
3	-17
4	24
5	8
6	0
7	0

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	O(K)	T(K)	C(K)	H(K)	F(K)
1	1	2	178	250	0
2	2	3	185	250	0
3	2	6	200	220	0
4	2	7	7	9999	7
5	3	4	168	250	0
6	3	6	200	230	0
7	4	5	192	250	0
8	4	6	200	110	0
9	4	7	24	9999	24
10	5	6	200	120	0
11	5	7	8	9999	8
12	6	7	200	0	0
13	6	7	9999	9999	0
14	7	1	25	0	0
15	7	2	10	0	0
16	7	3	24	0	0
17	7	4	10	0	0
18	7	1	22	9999	22
19	7	3	17	9999	17

SOLUCION OPTIMA DEL PROBLEMA

NODO I	P(I)	PB(I)
1	0	14
2	-220	-3
3	0	16
4	-110	-8
5	-120	-10
6	0	-12
7	0	0

ARC K	O(K)	T(K)	FLUJO	CAPACIDAD	COST/UNI.	COST.FLUJO
1	1	2	0	178	250	0
2	2	3	0	185	250	0
3	2	6	7	200	220	1540
4	2	7	0	7	9999	0
5	3	4	0	168	250	0
6	3	6	0	200	230	0
7	4	5	0	192	250	0
8	4	6	24	200	110	2640
9	4	7	0	24	9999	0
10	5	6	8	200	120	960
11	5	7	0	8	9999	0
12	6	7	39	200	0	0
13	6	7	0	9999	9999	0
14	7	1	22	25	0	0
15	7	2	0	10	0	0
16	7	3	17	24	0	0
17	7	4	0	10	0	0
18	7	1	0	22	9999	0
19	7	3	0	17	9999	0

COSTO TOTAL = 5140

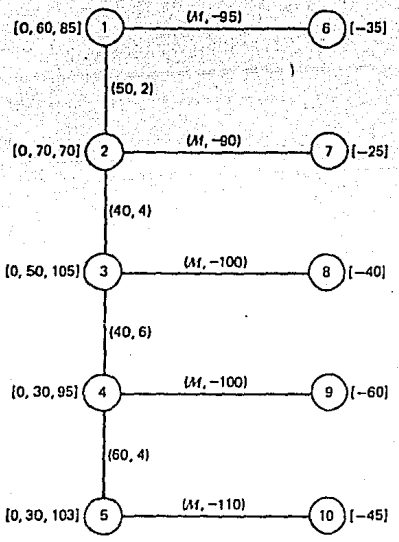
FIN DEL PROGRAMA RFCLSG

EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES: 0.350000

#ET=3:37.5 PT=0.5 IO=0.2

Mes	Precio de compra ($/Ton)	Precio de venta ($/Ton)	Costo de mantener ($/Ton)	Disponib. de compra (Tons.)	Disponib. de almacen (Tons.)	Demanda (Tons.)
1	85	95	2	6000	5000	3500
2	70	90	4	7000	4000	2500
3	105	100	6	5000	4000	4000
4	95	100	4	3000	6000	6000
5	103	110	4	3000	7000	4500

La figura A-33 indica la red que la empresa usa para minimizar el costo de suministro de trigo a la URSS, en la cual los nodos del 1 al 5 representan los puntos de suministro de trigo en cada mes. Los nodos del 6 al 10 representan los puntos de demanda de la - URSS en cada mes. Los arcos (1,2), (2,3), (3,4) y (4.5) reflejan el costo de almacenamiento de los sobrantes de trigo en cada mes, finalmente los arcos (1,6), (2,7), (3,8), (4,9) y (5,10) reflejan el flujo de trigo desde los puntos de suministro a los puntos de demanda. Los datos de entrada y resultados se organizan y estructuran en la misma forma que en el problema anterior. La solución del problema se presenta en la figura A-34. Este problema se resuelve con el algoritmo primal no básico.




```
100     65
200     10
300      1          0.       60.        85.
400      2          0.       70.        70.
500      3          0.       50.       105.
600      4          0.       30.        95.
700      5          0.       30.       103.
800      6        -35.
900      7        -25.
1000     8        -40.
1100     9        -60.
1200    10        -45.
1300
1400     1     6    0.     1000.       -95.
1500     1     2    0.       50.         2.
1600     2     7    0.     1000.       -90.
1700     2     3    0.       40.         4.
1800     3     8    0.     1000.      -100.
1900     3     4    0.       40.         6.
2000     4     9    0.     1000.      -100.
2100     4     5    0.       60.         4.
2200     5    10    0.     1000.       -45.
2300
$
```

Fig. A-33

EL NRO. CLAVE DE ESTA SUBRUTINA ES : 65

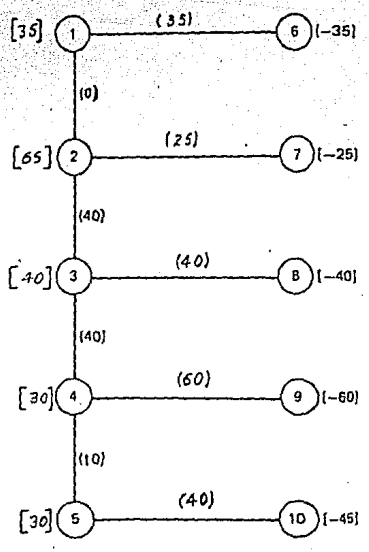
PROGRAMA FCM165
PRIMAL NO BASICO PARA FLUJO A COSTO MINIMO

REPRESENTACION DE LA RED--

NRO. DE NODOS-- 11 NRO. DE ARCOS-- 14

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO I	B(I)
1	0
2	0
3	0
4	0
5	0
6	-35
7	-25
8	-40
9	-60
10	-45
11	0



PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO K	D(K)	T(K)	C(K)	H(K)
1	1	6	1000	-95
2	1	2	50	2
3	2	7	1000	-90
4	2	3	40	4
5	3	8	1000	-100
6	3	4	40	6
7	4	9	1000	-100
8	4	5	60	4
9	5	10	1000	-45
10	11	1	60	85
11	11	2	70	70
12	11	3	50	105
13	11	4	30	95
14	11	5	30	103

-- SOLUCION OPTIMA DEL PROBLEMA

NODO I	P(I)	PB(I)
1	85	10
2	70	11
3	105	12
4	10040	-8
5	10044	-9
6	-10	1
7	-20	3
8	5	5
9	9940	7
10	9999	0
11	0	0

ARC K	D(K)	T(K)	FLUJO	CAPACIDAD	COST/UNI.	COST.FLUJO
1	1	6	35	1000	-95	-3325
2	1	2	0	50	2	0
3	2	7	25	1000	-90	-2250
4	2	3	40	40	4	160
5	3	8	40	1000	-100	-4000
6	3	4	40	40	6	240
7	4	9	60	1000	-100	-6000
8	4	5	10	60	4	40
9	5	10	40	1000	-45	-1800
10	11	1	35	60	85	2975
11	11	2	65	70	70	4550
12	11	3	40	50	105	4200
13	11	4	30	30	95	2850
14	11	5	30	30	103	3090

COSTO TOTAL = 730

FIN DEL PROGRAMA RFCLSG

EL TIEMPO DE PROCESO DE ESTE PROGRAMA ES: 0.350000

Fig. A-34

REFERENCIAS

1. Jensen, P.A. y Barnes, J.W., "Network Flow Programming", John Wiley, 1980

2. Bazaraa, M.S. y Jarvis, J.J. "Linear Programming and Network Flows", John Wiley, 1977

3. Kenninton, J.L. y Helgason, R.V., "Algorithms for Network Programming", John Wiley, 1980.

4. Murray, M.A. y Chicurel, U. E., "Aplicaciones de computación a la Ingeniería", Limusa, 1975

5. Prawda, W.J., "Método y Modelos de Investigación de Operaciones", Limusa, 1979

6. Wagner, M.H., "Principles of Operations Research", Prentice Hall. 1975