UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

EL PROBLEMA DE FLUJO MAXIMO EN UNA RED Y SU SOLUCION POR MEDIO DE ALGORITMOS IMPLANTADOS EN COMPUTADORA

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A :

RICARDO JOSE HERNANDEZ LOYOLA

México, D. F. 1983

C O N T E N I D O

Pág.

1. CONCEPTOS SOBRE REDES

1.1 Definición y estructura de una red 1

1.2 Notación y parámetros utilizados en redes 5

1.3 Redes expandida y marginal 7

2. EL PROBLEMA DE FLUJO MAXIMO EN UNA RED

2.1 Descripción y planteamiento del problema de flujo máximo 9

2.2 Conceptos sobre flujo factible y mínima cortadura 11

2.3 Interpretación física del problema dual 12

2.4 Resultados teóricos 14

2.5 Soluciones básicas y no básicas 16

2.6 Conceptos sobre aumento de flujo 17

3. REPRESENTACION Y MANEJO DEL PROBLEMA POR COMPUTADORA

3.1 Representación de redes 19

3.2 Lectura y almacenamiento de una red
Algoritmos utilizados 24

3.3 Construcción del Arbol. Algoritmos utilizados 30

3.4 Algoritmos empleados para una solución no básica 44

3.5 Algoritmos empleados para una solución básica 50

4. ESTRUCTURA Y MANEJO DEL PROGRAMA MAXFLUJO

4.1 Consideraciones generales 56

4.2 Ejemplo ilustrativo 57

4.3 Preparacion y ejecución del programa 58

4.4 Solución del ejemplo ilustrativo 61

5. EJEMPLOS DE APLICACION 66

APENDICE A

A.1 Listado del programa MAXFLUJO 86

BIBLIOGRAFIA 105

CAPITULO I

CONCEPTOS SOBRE REDES

1.1 DEFINICION Y ESTRUCTURA DE UNA RED

Una gráfica dirigida denotada G = (N,A) consiste de un conjunto finito N cuyos elementos se denominan nodos y un conjunto A formado por pares ordenados de nodos, denominados arcos. La forma de dibujar una gráfica dirigida, es dibujar círculos pequeños que no se intersecten, para caracterizar cada nodo $i,j \in N$, y dibujar para cada arco $(i,j) \in A$, una línea o flecha dirigida del nodo i al nodo j. Note que en una gráfica dirigida podemos tener arcos (i,i), y que el arco (i,j) es diferente al (j,i). Por ejemplo, la gráfica dirigida de la fig. 1.1 consiste de tres nodos y cinco arcos, esto es: G = (N,A) donde N = {1,2,3} y A = {(1,2), (1,3), (2,1), (2,2), (3,2)}.

En una gráfica dirigida G = (N,A) se define una cadena del nodo i al nodo j como la sucesión de nodos distintos de N, denotados por $i=i_1,i_2,\ldots,i_r=j$, y arcos de A, 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 existe ambiguedad, sólo se especifican los nodos que forman la cadena. Si en la definición de cadena se permite que cada arco sea de la forma $a_t = (i_t,i_{t+1})$ ó bien $a_t = (i_{t+1},i_t)$ donde $t=1,\ldots,r-1$ entonces la sucesión resultante se denomina trayectoria del nodo i al j.

CAPITULO I

CONCEPTOS SOBRE REDES

1.1 DEFINICION Y ESTRUCTURA DE UNA RED

Una gráfica dirigida denotada G = (N,A) consiste de un conjunto finito N cuyos elementos se denominan nodos y un conjunto A formado por pares ordenados de nodos, denominados arcos. La forma de dibujar una gráfica dirigida, es dibujar círculos pequeños que no se intersecten, para caracterizar cada nodo $i,j \in N$, y dibujar para cada arco $(i,j) \in A$, una línea o flecha dirigida del nodo i al nodo j. Note que en una gráfica dirigida podemos tener arcos (i,i), y que el arco (i,j) es diferente al (j,i). Por ejemplo, la gráfica dirigida de la fig. 1.1 consiste de tres nodos y cinco arcos, esto es: G = (N,A) donde N = {1,2,3} y A = {(1,2), (1,3), (2,1), (2,2), (3,2)}.

En una gráfica dirigida G = (N,A) se define una cadena del nodo i al nodo j como la sucesión de nodos distintos de N, denotados por $i=i_1,i_2,\ldots,i_r=j$, y arcos de A, denotados por $a_1,a_2,\ldots,a_r$ tales que $a_t=(i_t,i_{t+1})$, donde t=1,...,r-1. Si no existe ambiguedad, sólo se especifican los nodos que forman la cadena. Si en la definición de cadena se permite que cada arco sea de la forma $a_t = (i_t,i_{t+1})$ O bien $a_t = (i_{t+1},i_t)$ donde t=1,...,r-1 entonces la sucesión resultante se denomina trayectoria del nodo i al j.

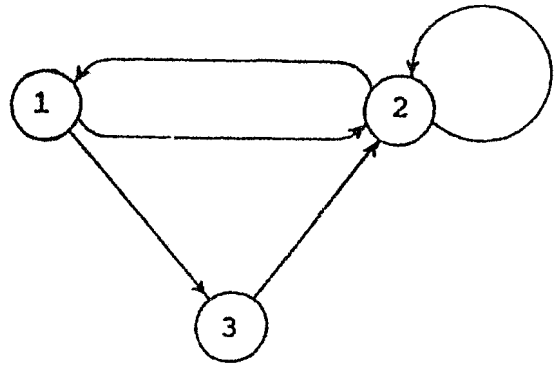


Fig. 1.1 Gráfica dirigida

En una gráfica dirigida G=(N,A) se define un circuito como una cadena en que el nodo inicial es igual al nodo final. Asimismo, un ciclo es una trayectoria con el mismo nodo inicial y final. Una cadena aumentada, es una cadena o un circuito o la unión de ambos mediante un arco. Observe que cadenas y circuitos necesitan que todos los arcos tengan un mismo sentido. También observe que toda cadena es una trayectoria y que todo circuito es un ciclo.

Ejemplo 1. En la gráfica dirigida de la fig. 1.1, se tiene

a. cadena del nodo 2 al 3; nodos 2,1,3; arcos (2,1) y (1,3)

b. trayectoria de 3 a 1; nodos 3,1; arco (1,3)

c. circuito de 1 a 1; nodos 1,3,2,1; arcos (1,3), (3,2) y (2,1)

d. ciclo de 1a 1: nodos 1,2,3,1; arcos (1,2), (3,2) y (1,3).

La gráfica dirigida más importante es la red. Una red es una gráfica dirigida G=(N,A) en donde no existen arcos de la forma $(i,i)\in A$. Es común asociar a los elementos de una red ciertos parámetros. Específicamente, dado el nodo $i\in N$, se denota por d_i la disponibilidad en este nodo y se dice que el nodo es fuente, sumidero o traspaso, si la disponibilidad es positiva, negativa o cero, respectivamente.

Existen diferentes tipos de redes. Sólo mencionaremos las más comunes. Una red, G=(N,A), es <u>bipartita</u> si el conjunto de nodos N puede dividirse en dos subconjuntos N_1, N_2, tales que si $(i,j)\in A$ entonces, $i\in N_1$ y $j\in N_2$. Una red, G=(N,A), es <u>simple</u> si tiene un sólo nodo fuente s y un sólo nodo sumidero t, y no existen arcos de la forma (i,s) o (t,j) donde $i,j\in N$. Una red es <u>circulatoria</u> si todos sus nodos son de traspaso. (fig. 1.2).

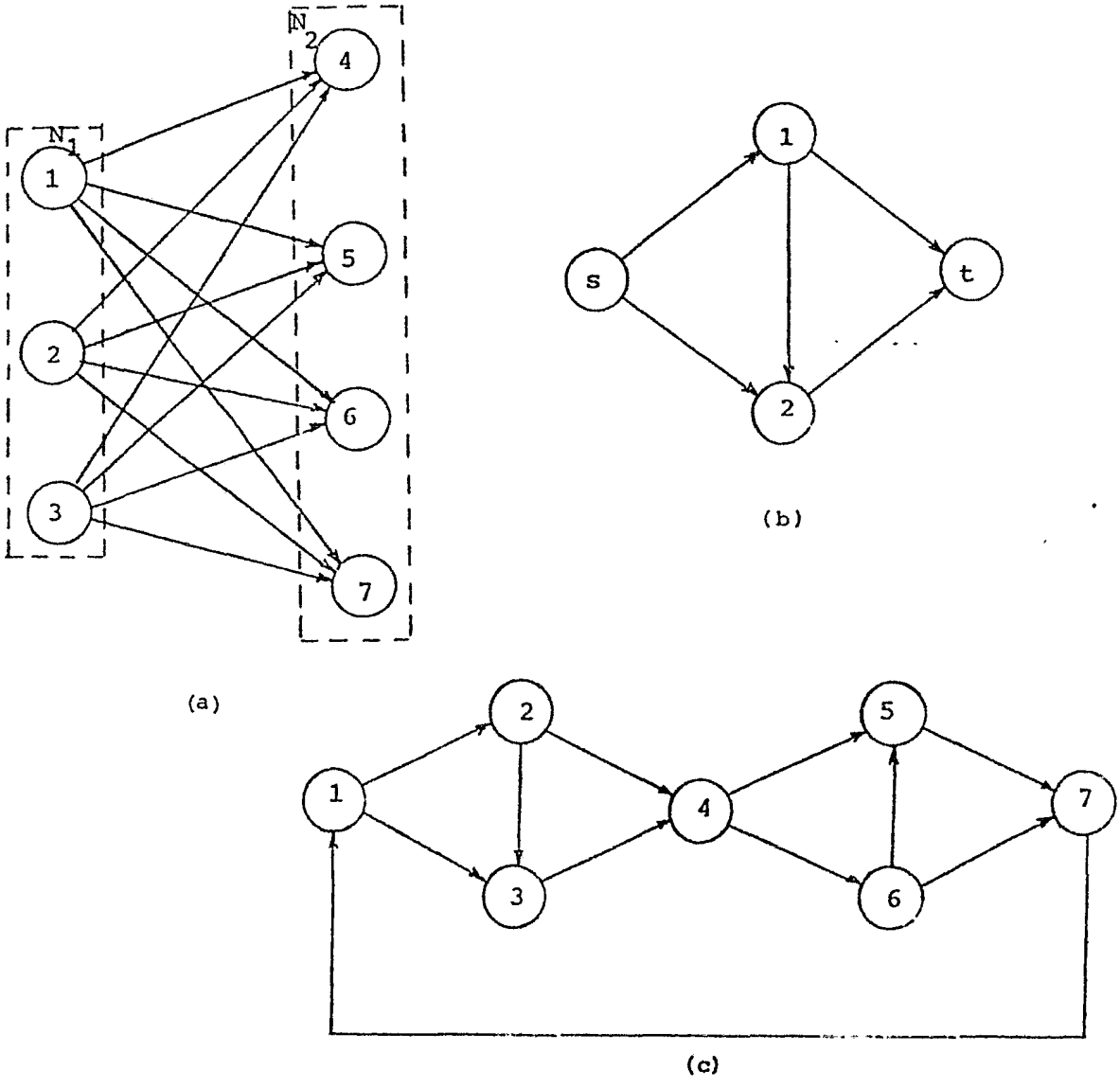


fig. 1.2 (a) red bipartita, (b) red simple
(c) red circulatoria

Una forma alternativa de representar una red es mediante la matriz de incidencia nodos-arcos $A = [a_{ij}]$ donde

$$a_{ij} = \begin{cases} +1 & \text{si el arco } j \text{ sale del nodo } i \\ -1 & \text{si el arco } j \text{ llega al nodo } i \\ 0 & \text{otro caso} \end{cases}$$

La matriz A tiene la característica de que en cada una de sus columnas sólo existen dos elementos diferentes de cero, es decir, las columnas de A están dadas por $a_j = e_i - e_j$, donde e_i, e_j son vectores unitarios en R^m y m es el número de arcos.

Se puede ver que la matriz A no tiene rango total pues la suma de sus renglones es el vector cero, si se selecciona una submatriz (m-1) x (m-1) de P que no sea singular se puede demostrar que A tiene precisamente rango m-1. Por las características que presentan las redes, se pueden formular como modelos de programación lineal y ya que en el método simplex utilizado en problemas de programación lineal se inicia con una matriz de restricciones de rango total se requiere de una varible artificial de tal forma que el rango de la nueva matriz sea m. Las soluciones básicas deben contener m columnas linealmente independientes y en consecuencia la variable artificial debe aparecer en cada solución básica.

Se dice que una gráfica es conectada si existe una cadena que una cada par de vértices distintos en la gráfica. Un <u>árbol</u> es una gráfica G = [N,A] que satisface dos condiciones; a) es conectada y b) no tiene ciclos.

Un <u>árbol generador</u> es un árbol que incluye todos los nodos de la red original.

1.2 NOTACION Y PARAMETROS UTILIZADOS EN REDES

Otra forma de definir a un arco ademas de un par ordenado de nodos es como un elemento k del conjunto de arcos M(1,2,...,k,...,m). El arco k ó k (i,j), se origina en el nodo i, nodo origen, y termina el j denominado nodo terminar o destino. Es posible definir todas las conexiones presentadas en la red por medio de las listas origen y destino dadas como:

$$O = (o_1, o_2, \ldots\ldots, o_m)$$

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

donde o_k y t_k son los nodos origen y destino, respectivamente, del arco k. Un parámetro muy importante que se contempla en una red es el flujo en el arco denotado por f_k o f(i,j) el cual usualmente representa cantidades físicas como flujo de un fluido, flujo de personas, etc. La principal característica del flujo en la red, es que se conserva en los nodos, esto es que el flujo que entra es igual al flujo que sale en un nodo.

El flujo en los arcos puede o no conservarse igual; en el caso de que el flujo varie se trata de una red con ganancias cuyo flujo en cada arco esta determinado por

$$f'_k = a_k f_k$$

Donde a_k es la ganancia en el arco k y siempre será positiva; si a_k=1 el flujo se conserva, si $a_k < 1$ hay pérdida de flujo y si $a_k > 1$ hay incremento de flujo en cada arco. Para nuestros propósitos la ganancia siempre será unitaria, esto es, que el flujo también se conserva-

varía en los arcos.

Otro parámetro importante es el costo el cual siempre va asociado con el flujo en el arco. El costo en el arco $h_k(f_k)$ es una función solo del flujo en el arco k y es independinte del flujo en otros arcos. La capacidad es un parámetro que define el limite superior de flujo en cada arco y se le denota por C_k donde $f_k \leq C_k$. También se presenta en algunos casos la necesidad de incluir un limite inferior denotado por $\underline{C}_k$, donde $f \geq \underline{C}_k$.

1.3 REDES EXPANDIDA Y MARGINAL

Los conceptos de redes expandida y marginal son utilizados en el desarrollo de los algoritmos para la obtención de soluciones óptimas de los problemas de redes, por lo que resulta conveniente definirlos.

Primero definiremos la existencia de un arco reflejado, -k, por cada arco $k \in M$ tal que el arco -k conecta los mismos nodos que k pero en dirección opuesta.

La red expandida, $G_E = (N, M_E)$, tiene el mismo conjunto de nodos que la red original G y su conjunto de arcos contiene además que los de G, a todos los arcos reflejados, o sea, si $M = (1,2,\ldots,m)$ entonces $M_E = (1,2,\ldots,m, -1, -2,\ldots,-m)$ (fig. 1.3)

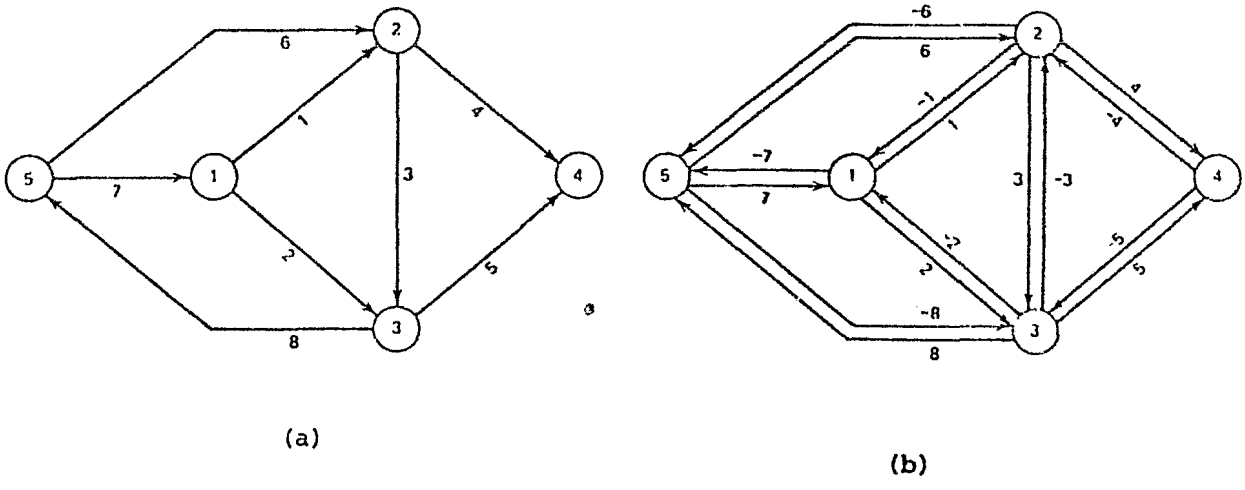


fig. 1.3 (a) Red original (b) Red expandida

Los arcos de la red original son llamados arcos hacia adelante; si la red es conectada, entonces existe una trayectoria dirigida entre cada par de nodos en la red expandida. Es claro que el arco reflejado de un arco reflejado es precisamente su asociado arco hacia adelante. Por otro lado se observa que si h_k es el costo asociado del arco hacia adelante, entonces $-h_k$ será el costo para el arco reflejado -k.

La red marginal, G* = (N, M*), tiene el mismo conjunto N de nodos que G y G_E, y su conjunto de arcos M* consiste en un subconjunto de M_E que incluye a todos los arcos admisibles. (fig. 1.4). Un arco hacia adelante es admisible en la red marginal si no ha alcanzado su capacidad; un arco reflejado es admisible si el flujo es su correspondiente arco hacia adelante es mayor que cero, o sea que pueda ser posible disminuir el flujo en el arco asociado de la red original. Al incrementar el flujo en un arco reflejado se decrementa en la misma proporción en su correspondiente arco hacia adelante.

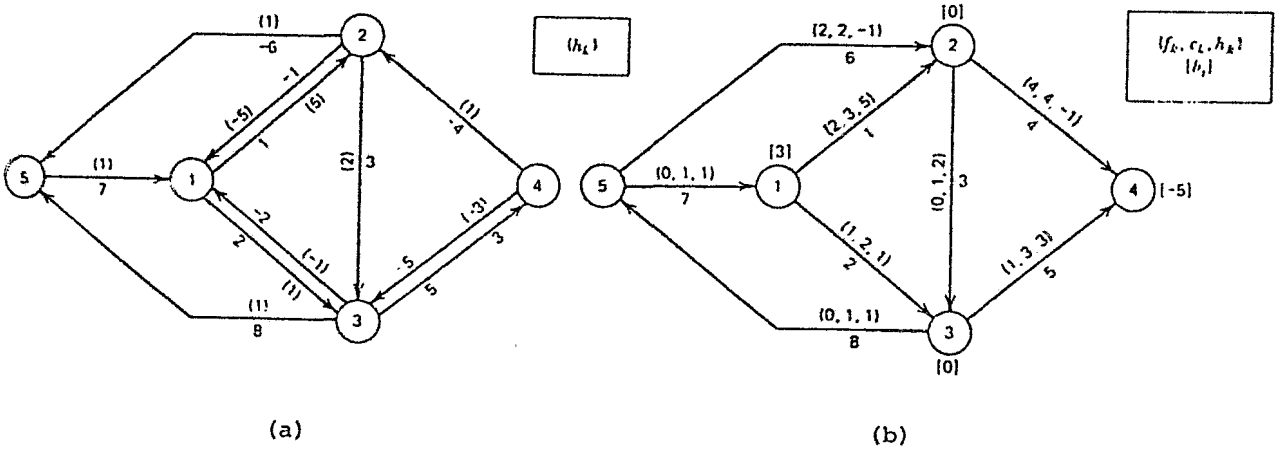


Fig. 1.4 (a) Red original (b) Red marginal asociada

CAPITULO II

2.1 EL PROBLEMA DE FLUJO MAXIMO EN UNA RED

Un caso importante de problemas de flujos en redes es el problema de flujo máximo el cual se puede resolver ya sea por el metodo simplex, o bién, mediante algoritmos especializados mas eficientes para manejar este problema.

Considérese una red con n nodos y m arcos a través de la cual fluye un solo tipo de bien o unidad. Con cada arco k se asocia sobre el flujo una cota inferior c_k = o y una cota superior C_k. En el problema de flujo máximo no intervienen los costos. En la red, se desea encontrar la cantidad máxima de flujo, desde el nodo fuente s al nodo sumidero t (fig. 2.1). Los algoritmos que serán introducidos son usados para encontrar el flujo máximo o un flujo requerido v_r, que puede pasar desde un nodo fuente hasta un nodo sumidero.

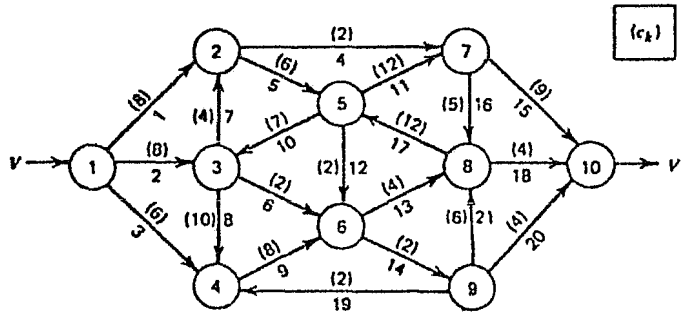


Fig. 2.1 Ejemplo de un problema de flujo máximo

La representación como un problema de programación lineal del problema de flujo máximo en una red es:

$$\max \quad v \tag{1}$$

sujeto a

$$\sum_{k\epsilon M_{O_i}} fk - \sum_{k\epsilon M_{T_i}} f_k = 0 \qquad i\epsilon N - \{s,t\} \tag{2a}$$

$$\sum_{k\epsilon M_{O_s}} f_k - \sum_{k\epsilon M_{T_s}} f_k = v \tag{2b}$$

$$\sum_{k\epsilon M_{O_t}} f_k - \sum_{k\epsilon M_{T_t}} f_k = -v \tag{2c}$$

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

$$0 \leq f_k \leq C_k \tag{4}$$

Por el momento consideramos en la ecuación 3 que v_r puede ser un número muy grande (esto implica un flujo inalcanzable). Por otra parte la ecuación de conservación de flujo para el nodo s(2b) puede salir de las restricciones ya que es redundante.

2.2 CONCEPTOS SOBRE FLUJO FACTIBLE Y MINIMA CORTADURA.

Antes de continuar con el desarrollo del problema de flujo máximo es necesario introducir los conceptos de flujo factible y mínima cortadura.

El flujo que circula por un arco k es factible si cumple con las restricciones de capacidad del arco, o sea $c_k \leq f_k \leq C_k$ y será <u>factible</u> en la red si cumple con esta condición en todos los arcos de la red. Por otro lado sea X cualquier conjunto de nodos en la red tal que X contiene el nodo 1 pero no al nodo m sea $\bar{X}$=N-X. Entonces $(X,\bar{X}) = \{(i,j): i\epsilon X,\ j\epsilon\bar{X}\}$ se llama <u>cortadura</u> que separa al nodo m del nodo 1.

Sea $(X,\bar{X})$ cualquier cortadura en una red. Entonces $C(X,\bar{X})=\Sigma\ C(i,j)$ donde $(i,j)\epsilon(X,\bar{X})$ se llama <u>capacidad</u> de la cortadura. En otras palabras la capacidad de una cortadura es la suma de las capacidades de los arcos que van de X a $\bar{X}$ y la menor capacidad de todas las cortaduras será la capacidad de mínima cortadura.

Por otro lado se puede demostrar que el valor v de cualquier flujo factible es menor o igual que la capacidad $C(X,\bar{X})$ de cualquier cortadura (que separa al nodo m del nodo 1).

2.3 INTERPRETACION FISICA DEL PROBLEMA DUAL

El problema Dual de la representación del problema de flujo máximo como uno de programación lineal es:

$$\text{Min} \quad \sum_{k=1}^{m} C_k \delta_k + v_r \delta_{m+1} \tag{5}$$

sujeto a:

$$\Pi i - \Pi j + \delta_k \geq 0 \qquad \text{si } k(i,j) \in M \tag{6}$$

$$\Pi_t - \Pi_s + \delta_{m+1} \geq 1 \tag{7}$$

$$\delta_k \geq 0 \qquad K = 1,2,\ldots,m+1 \tag{8}$$

Es conveniente desde el 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 la mínima cortadura y se puede demostrar que el valor del flujo máximo en una red es igual a la capacidad de la mínima cortadura. Los algoritmos que serán utilizados resuelven los dos problemas simultáneamente, el de corte mínimo y el de flujo máximo. A continuación se presenta la demostración del teorema que establece que el flujo máximo es igual al valor del costo mínimo.

Teorema 1. (Flujo máximo-corte mínimo). En toda red el flujo máximo es igual al corte mínimo.

Prueba. Observe que el flujo permitido es siempre menor o igual a la capacidad de cualquier corte. Entonces solo es necesario establecer un flujo cuyo valor sea igual a la capacidad de un corte. De esta manera quedará demostrado que dicho corte es mínimo. Suponga que denotamos por f el flujo máximo del nodo origen s al nodo sumidero t y defina XCN como: a). $s\in X$; b) si $x\in X$ y $f(x,y) < c(x,y)$ entonces $y\in X$; c) si $x\in X$ y $f(y,x) > 0$ entonces $y\in X$. Existen dos casos a considerar:

Caso 1. $t\in\bar{X}$. Entonces el corte $(X,\bar{X})$ separa s de t y se tiene, por definición del conjunto X que $f(X,\bar{X}) = C(X,\bar{X})$ y $f(\bar{X},X) = 0$. Esto implica que el corte propuesto es mínimo.

Caso 2. $t\in X$. Entonces existe una trayectoria en X que conecta el nodo s con t. Específicamente, existen nodos en la red simple G = [N , A], denotados $s = x_1, x_2,\ldots,x_n = t$ tales que

$\varepsilon_i = c(x_i, x_{i+1}) - f(x_i, x_{i+1})$ $(x_i, x_{i+1})\in A$ ó $\varepsilon_i = f(x_i, x_{i+1})$ $(x_{i+1},x_i)\in A$

donde ε_i es positivo (i=1,...,n-1). Denote por ε el mínimo de estos escalares positivos e incremente el flujo de s a t como sigue: si $(x_i,x_{i+1})\in A$ el nuevo flujo es $f(x_i,x_{i+1})+\varepsilon$ y si $(x_{i+1},x_i)\in A$ el flujo es $f(x_{i+1},x_i)-\varepsilon$. Es sencillo verificar que las ecuaciones de continuidad en cada nodo $x\neq s,t$ se satisfacen y que el flujo de s a t se ha incrementado en un valor ε. Esto contradice la suposición que f es el flujo máximo y se tiene necesariamente que $t\notin X$ o equivalentemente $t\in\bar{X}$. Esto termina la prueba.

2.4 RESULTADOS TEORICOS

Se dispone de resultados interesantes de las soluciones de los problemas primal y dual.

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 límite 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 Πi y δk tienen valores enteros).
3. Hay una solución óptima dual para la cual todos los potenciales en los nodos son 0 ó 1 ($\Pi i = 0$ ó $\Pi i = 1$).
4. De las condiciones de holgura complementaria, se obtiene las condiciones de optimalidad.

Específicamente para cada arco $k(i,j) \in M$, 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 límite 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 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 $s \in N_1$, $t \in N_2$, $N_1 \cup N_2 = N$ y $N_1 \cap N_2 = \phi$ tal que:

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

para una solución dual óptima. los arcos que se originan en N_1 y terminal en N_2 forman la mínima cortadura.

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

$f_k(i,j) = C_k(i,j)$ para $i \in N_1$ y $j \in N_2$ y para los arcos que pasan desde N_2 a N_1, el flujo es cero, $f_k(i,j) = 0$ para $i \in N_2$, $j \in N_1$. El valor del máximo flujo es igual a la capacidad de la mínima cortadura.

$$\sum_{\substack{i \in N_1 \\ j \in N_2}} C_{k(i,j)} = v$$

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

2.5 SOLUCIONES BASICAS Y NO BASICAS

En los problemas de flujo máximo, existe la alternativa de obtener soluciones básicas o soluciones no básicas. Una solución no básica se caracteriza por tener un ciclo de arcos cuyos flujos estan estrictamente entre sus limites superior e inferior ($0 < f_k < C_k$), o sea que no forma un árbol. En cambio, una solución básica consiste de un conjunto de variables no básicas en una de sus cotas inferiores o superiores más un conjunto de variables que forman un árbol de expansión enraizado, o sea que no contiene un ciclo.

Los algoritmos que aquí se propondrán contemplan ambos casos, para obtener una solución básica (algoritmos básicos) y para obtener una solución no básica (algoritmos no básicos). Los algoritmos básicos requieren de un proceso computacional más completo para el manejo y almacenamiento de la información, en cambio los algoritmos no básicos son procesos de iteraciones más simples y requieren de menos recursos.

2.6 CONCEPTOS SOBRE AUMENTO DE FLUJO

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

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

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

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:

$$\begin{aligned} f'_k &= f_k \qquad \text{Si} \quad k \notin M_p \quad y \quad -k \notin M_p \\ f'_k &= f_k + \Delta \text{ para } k \in M_p, \qquad k > 0 \\ f'_{-k} &= f_{-k} - \Delta \text{ para } k \in M_p \qquad k < 0 \\ V &= V + \Delta \end{aligned} \qquad (10)$$

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 su respectivo arco reflejado disminuye. El nuevo flujo es factible para el problema de flujo máximo si se cumple:

$$0 \leq f_k' \leq C_k$$

Lo cual se asegura con:

$$\Delta = \text{Min} \left\{ \underset{k\in\{M_p | k \geq 0\}}{\text{Min}} (C_k - f_k), \underset{k\in\{M_p | k < 0\}}{\text{Min}} f_k \right\}$$

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 llamada 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 la 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 continúa hasta que se obtiene el flujo v_r o hasta cuando ya no se encuentra una trayectoria.

CAPITULO III

REPRESENTACION Y MANEJO DEL PROBLEMA POR COMPUTADORA

3.1 REPRESENTACION DE REDES.

La estructura de una red y sus parámetros se representan por medio de constantes, listas de variables y matrices. La representación que se use en computadora adquiere gran importancia debido a que un algoritmo utilizado para la manipulacion de la red será más o menos eficiente dependiendo de la cantidad de memoria y del tiempo consumido para lograr su cometido. Tiempo y espacio de memoria generalmente son variables complementarias en el diseño de procedimientos computacionales; La disminución de uno es acompañada por un incremento en el otro.

Existen dos enfoques para la representación de una red: Por nodos orientados y por arcos orientados. En el primer caso se utilizan las matrices origen-destino, en las cuales se almacenan las variables y parámetros tal que en el cruce de la fila i con la columna j contenga un valor asociado con el arco (i, j).

Generalmente esta representación se utiliza solo cuando las matrices son densas, como en los problemas de transporte; de otra forma este enfoque resulta costoso en memoria como en tiempo requerido.

La presentación por arcos-orientados se realiza mediante las listas de arcos $O = [O_k]$ y $T = [t_k]$, vistas en el primer capítulo; sus parámetros f_k, c_k y h_k se representan en forma similar.

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

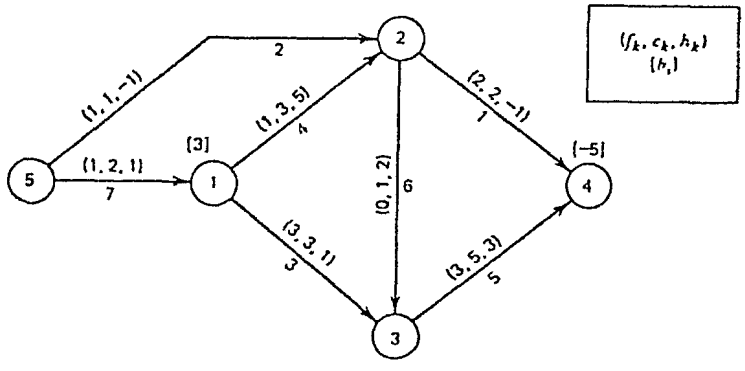
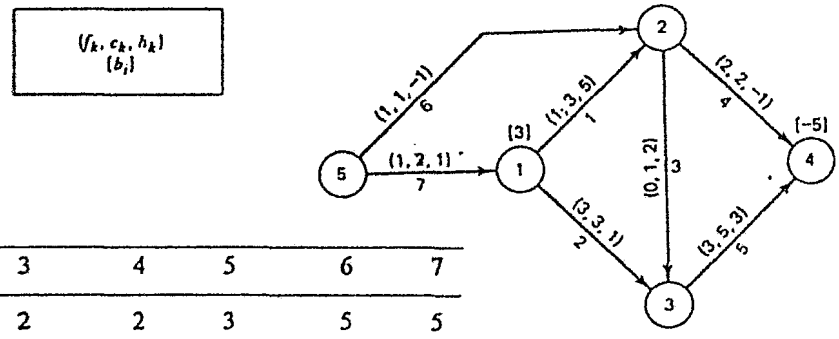


Fig. 3.1 Ejemplo de una red y su correspondiente lista de arcos

Es necesario realizar algunas modificaciones al almacenar la lista de arcos: La primera es ordenarla de acuerdo al crecimiento del nodo origen. En la figura 3.2 se indican los resultados de esta modificacion luego de reordenar la numeración de los arcos de la Fig. 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

Fig. 3.2 La Red y su correspondiente lista de arcos reordenada

Además se manejaran ciertas listas de apuntadores tales como la lista $P_0 = [p_o(i)]$, la cual contiene el arco con el menor numero indice que se origina en el nodo i; Si no se originan arcos en el nodo i, se hace $p_0(i) = p_0(i+1)$. Esta lista de arcos es ordenada en tal forma que:

$$0(k_1) < 0(k_2) \qquad \text{si } k_1 < k_2$$

$$0(k_1) = 0(k_2) \qquad \text{si el orden de } k_1 \text{ y } k_2 \text{ es arbitrario}$$

posteriormente, para $i \leq m$

$$p_o(1) = 1$$

$$p_o(i) = \{k | o(k) \geq i,\ o(k-1) < i\}\ 1 < i \leq n$$

y

$$p_0(n+1) = m + 1$$

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

$$Mo_i = \{k | p_o(i) \leq k < p_o(i+1)\}$$

aplicando estos conceptos a la red de la figura 3.2 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 $L_T = \{\ell_T(i)\}$, es otra lista que ordena los arcos de acuerdo al crecimiento del nodo terminal. Así, si k'_w es el indice del arco k_w en L_T, la lista de arcos es ordenada tal que:

$$t(k_w) < t(k_y) \quad \text{si } k'_w < k'_y$$

$$t(k_w) = t(k_y) \quad \text{si el orden de } k_w \text{ y } k_y \text{ es arbitrario.}$$

La lista $P_T = \{p_T(i)\}$, es la que contiene el número de arcos con el menor valor que termina en el nodo i, pero en base a la lista L_T, en la siguiente forma:

$$P_T(1) = 1$$

$$P_T(i) = \{k' \mid t(\ell_T(k')) \geq i,\ t(\ell_T(k'-1)) < i\}$$

para $i \leq n$

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

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

$$M_{T_i} = \{\ell_T(K') \mid P_T(i) \leq K' < P_T(i+1)\}$$

Aplicando estos conceptos a la red de la figura 3.2 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

3.2 LECTURA Y ALMACENAMIENTO DE UNA RED. ALGORITMOS UTILIZADOS

Es importante la lectura y el almacenamiento de los datos, al iniciar el proceso de cálculo en una computadora. Primeramente se lee los datos de cada uno de los nodos en la siguiente secuencia: 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 un espacio en blanco y se continúa con los datos de los arcos, los cuales son leídos por cada uno de los arcos en el siguiente orden: Nodo origen, nodo terminal, capacidad mínima, capacidad máxima y costo, luego se deja un espacio en blanco.

La entrada de datos puede ser diferente para cada tipo de problema pero la lógica de la entrada en forma general se mantiene.

Algoritmos utilizados.

En esta parte se presentan los algoritmos básicos que se utilizan en el almacenamiento de la red. Estos algoritmos y sus funciones se describen a continuación:

Algoritmo READ - Lee los datos de una red y usa el nodo de holgura para modificar la red, teniendo solamente flujos externos fijos.

Algoritmo 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.

Algoritmo ORIG - Encuentra la lista de arcos originados en el nodo i (M_{o_i}).

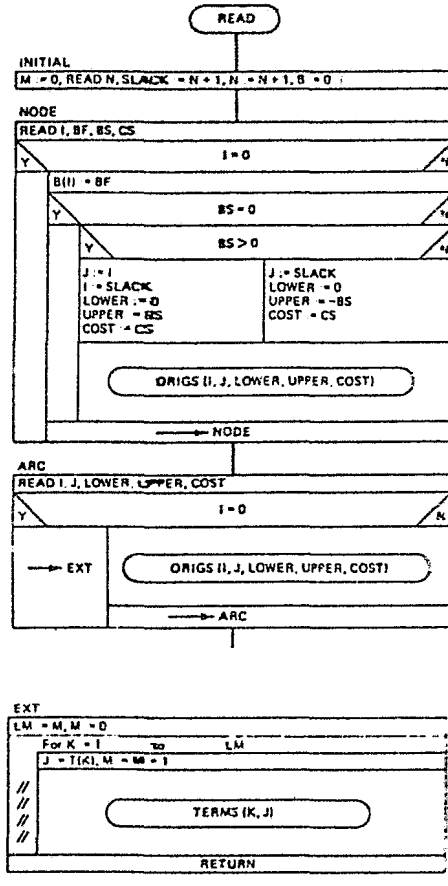
Algoritmo TERMS - Elabora la lista auxiliar L_T de los arcos ordenados de acuerdo al incremento del nodo terminal, y la lista de los apuntadores P_T.

Algoritmo TERM - Encuentra la lista de arcos que terminan en el nodo i (M_{T_i}), en base a la lista auxiliar L_T.

En escencia, estos cinco algoritmos simplemente implementan los conceptos contenidos en las ecuaciones vistas en este capítulo. La Presentación detallada de estos algoritmos aparece a continuación.

Algoritmo READ

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



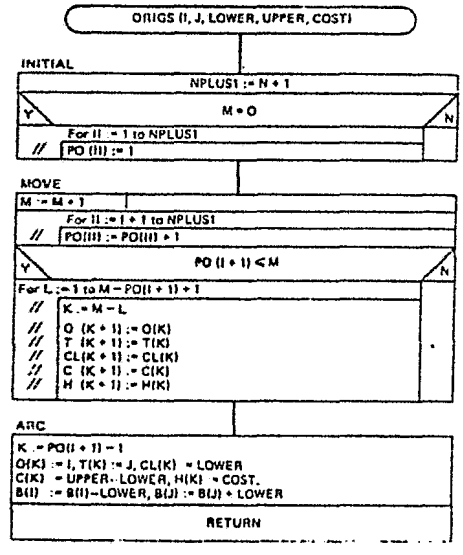
1. (INICIAR) Inicializa el número del arco a cero. Lee el número de nodos, crea el nodo de holgura (SLACK). Fija todos los flujos externos a cero.
2. (NODO) Lee los datos del nodo i (flujo externo B, capacidad superior BS y el costo de holgura CS). Si el renglón de datos está en blanco, va al paso 3; de lo contrario almacena los flujos fijados. 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 las listas de arcos (algoritmo ORIGS).
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.

(Algoritmo Origs). Repite el paso 3.

4. (EXT) Coloca cada dato de los arcos en la posición correcta en la lista terminal correspondiente (Algoritmo Terms)

Algoritmo ORIGS.

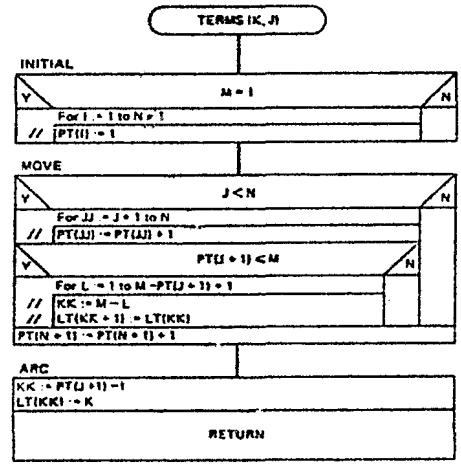
Propósito: Acepta el conjunto de datos correspondientes a un arco y los almacena en una lista ordenada incrementando el índice del nodo origen.



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.

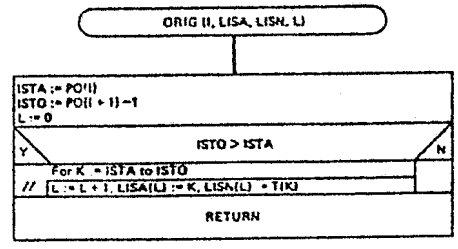
Propósito: Elabora la lista LT de los índices de los arcos en orden de acuerdo al incremento del nodo terminal. También produce la lista de apuntadores PT hacia los nodos terminales, tal que la lista LT puede ser referenciada rápidamente. Este algoritmo es llamado una vez por cada arco en la red.



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.

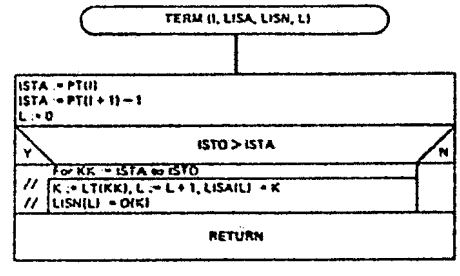
Algoritmo ORIG.

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 terminar de cada uno de estos arcos y los almacena en una lista de nodos LISN.



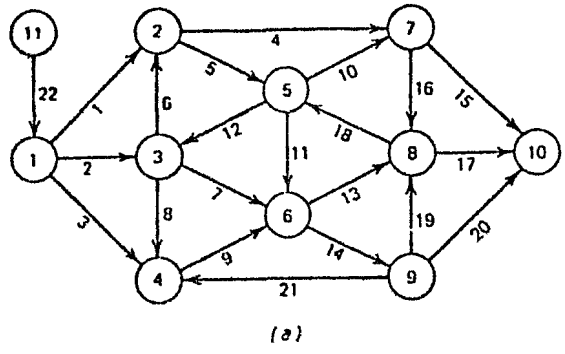
Algoritmo TERM

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.

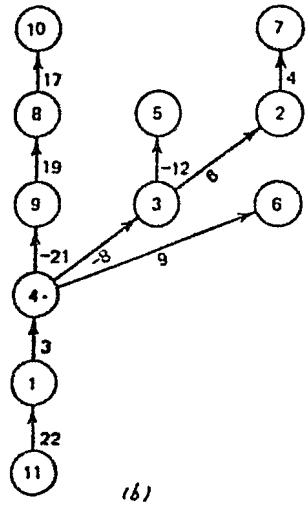


3.3 CONSTRUCCION DEL ARBOL. ALGORITMOS UTILIZADOS.

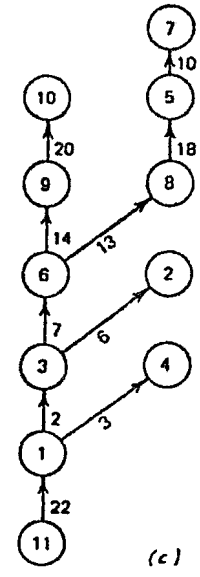
En esta sección se presentan los algoritmos para la construcción de un árbol dirigido. ROOT encuentra el subárbol enraizado a un nodo dado; DELTRE quita un arco desde un árbol básico; ADDTRE adiciona un arco a un bosque (donde un bosque es dos o más árboles dirigidos); TRECHG cambia de base, y TREINT inicializa un árbol. Varios de estos algoritmos usan una base (el árbol dirigido enraizado en un nodo de holgura), la misma que debe estar almacenada en tal forma que sea fácil de localizarla y modificarla. En la figura 3.3 se puede ver la red y sus correspondientes dos árboles expandidos. Como se puede notar cada árbol representa una única ruta desde el nodo raíz 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 (álbol dirigido). El árbol $D_T = [N_T, M_T]$, es una subred del árbol expandido y puede ser representado usando tres etiquetas, que es el método más usado. 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. Note que a cada nodo sólo se le puede asignar un sólo apuntador hacia atrás P_B, pero la asignación de



(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

(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

(e)

Fig. 3.3 (a) red, (b) y (c) árboles, (d) y (e) apuntadores de (b) y (c) respectivamente.

de p_F y p_R no necesariamente es única.

En la figura 3.3 se muestra la representación de apuntadores asociada con cada árbol de expansión. $p_D(i)$ es la altura o nivel de cada nodo.

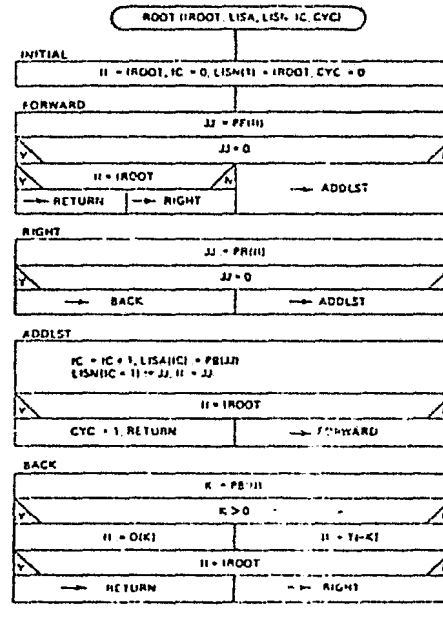
Dado un par de nodos i y j, existe una única ruta desde i a j definida por los arcos del árbol. Una ruta puede tener un nodo unión ℓ y además una ruta hacia adelante o una ruta inversa o ambas.

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.

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 3.3b,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 de 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 3.4a se ilustra el primer caso, al salir el arco -21(4,9) y en la figura 3.4b se ilustra el segundo caso al salir el arco -8(4,3); a partir de la red original de la figura 3.3b. 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.

Nodo	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

Nodo	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	3	3	5	0

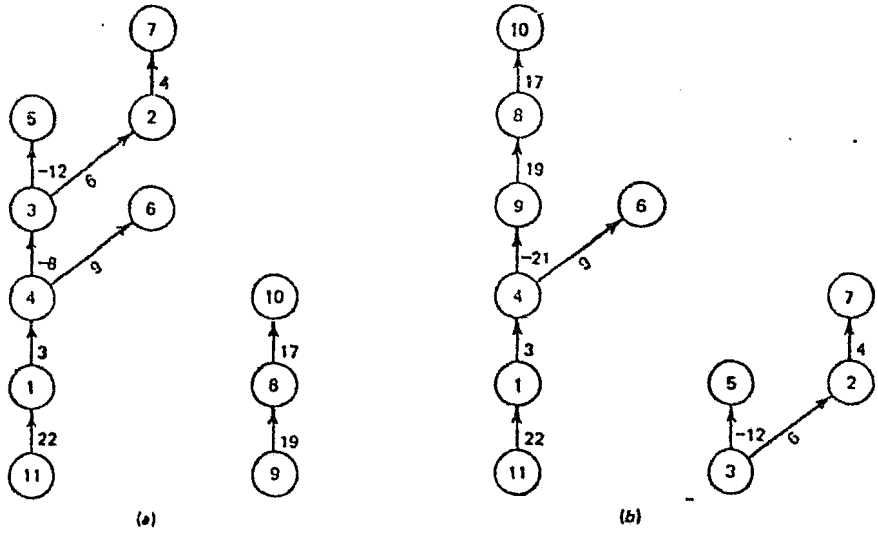
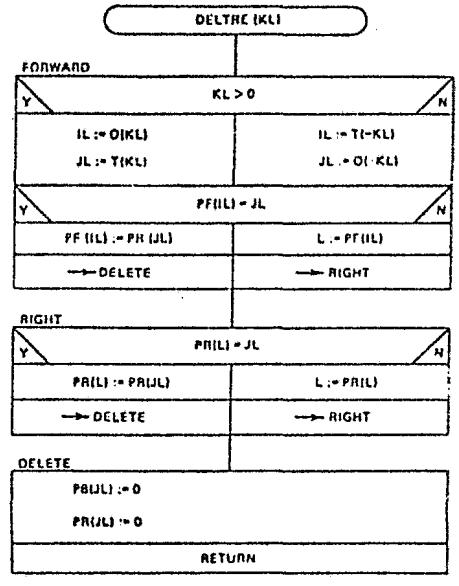


Figura 3.4

(a) Subárboles al salir del arco -21(4.9) del árbol de la figura 3.3b
(b) Subárboles al salir del arco -8(4.3) del árbol de la figura 3.3b

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 la paso 3. Si $P_F(i_L) \neq j_L$, va al paso 2.
2. (DERECHA) Encuentra el nodo ℓ para el cual $P_R(\ell):=j_L$. Hace $P_R(\ell):=P_R(j_L)$ va al paso 3.
3. (QUITAR) Hace al nodo j_L un nodo raiz; esto es, $P_B(j_L):=0$, $P_R(j_L):=0$.

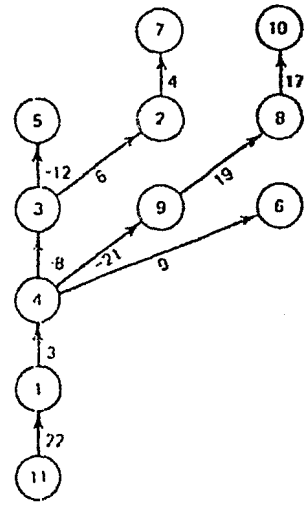
Para adicionar un arco a un bosque usando las tres etiquetas, Se define primeramente un bosque como dos o más árboles dirigidos, D_{T_1}, D_{T_2},... Cuando se añade un arco $k_E(i_E,j_E)$ a un bosque, tal que el nodo i_E se encuentre en un árbol y el nodo j_E sea la raiz del otro, estos dos árboles 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 $p_F(i_E) := \ell$, entonces $p_R(j_E) := \ell$

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 3.5 se puede observar un árbol al añadir el arco -8(4.3) partiendo del árbol de la figura 3.4. Las operaciones anteriores lo realiza el algoritmo ADDTRE.

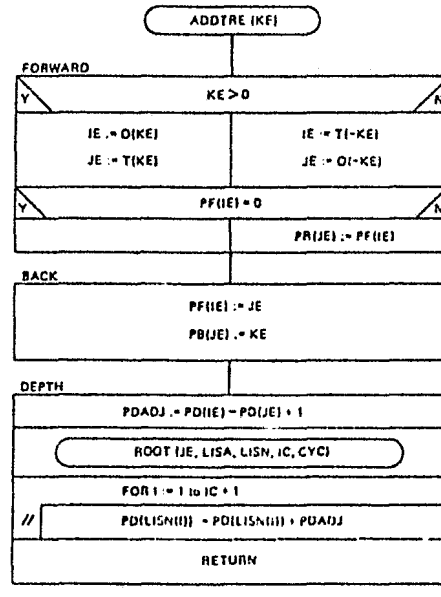


Nodo	1	2	3	4	5	6	7	8	9	10	11
P_B	22	6	8	3	-12	9	4	19	-21	17	0
P_F	4	7	5	3	0	0	0	10	8	0	1
P_R	0	0	9	0	2	0	0	0	6	0	0
P_D	1	4	3	2	4	3	5	4	3	5	0

Figura 3.5 Arco -8(4.3) añadido al bosque de la fig. 3.4 (b)

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 $M'_T = M_1\ U\ k_e\ U\ M'_2$

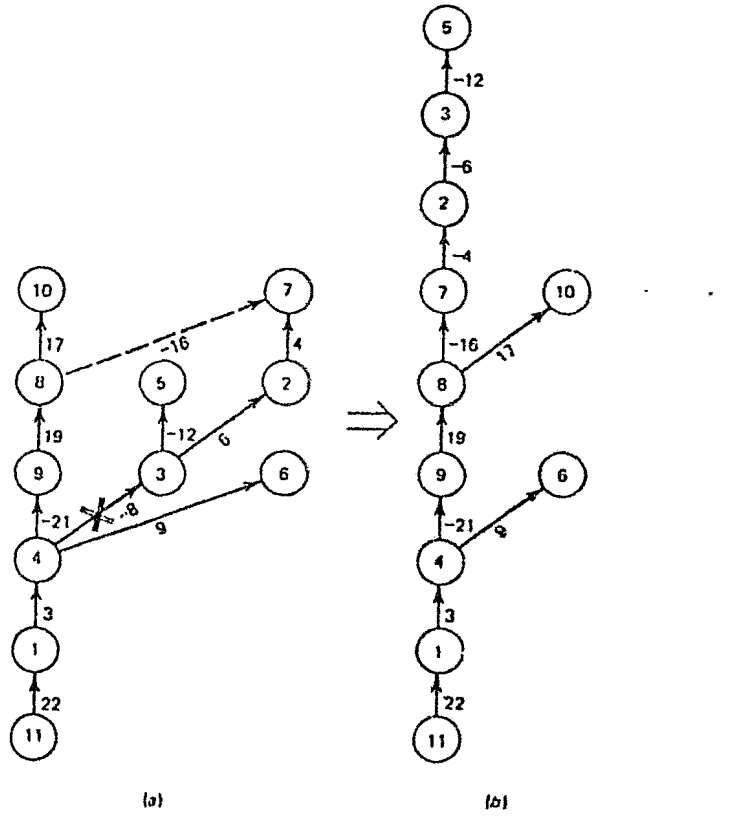
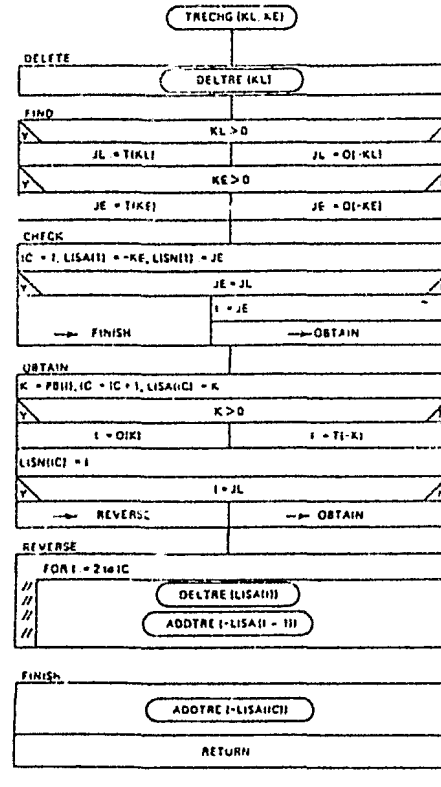


Figura 3.6 (a) Base original. (b) Base luego de sacar el arco -8(4.3) y añadir el arco -16(8,7).

La figura 3.6 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 mantener el árbol dirigido. El algoritmo asume que el nodo terminal del arco entrante está en N_2.



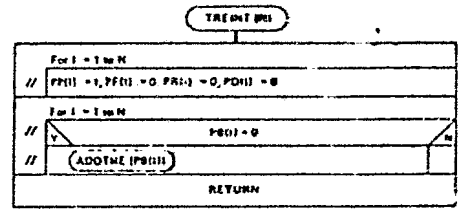
1. (QUITAR) Quita el arco k_L de la base.
2. (ENCONTRAR) Encuentra los nodos terminales para k_E y k_L.
3. (CHEQUEAR) Inicializa el índice IC y la lista de arcos y nodos. Si el nodo terminal de los arcos que entra y sale es el mismo, va al paso 6. De otra manera va al paso 4.
4. (OBTENER) Obtiene la 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 algoritmo 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.

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 3.7, 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 can-

tidad 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:

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

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

Esta operación es realizada por el algoritmo FLOCHG.

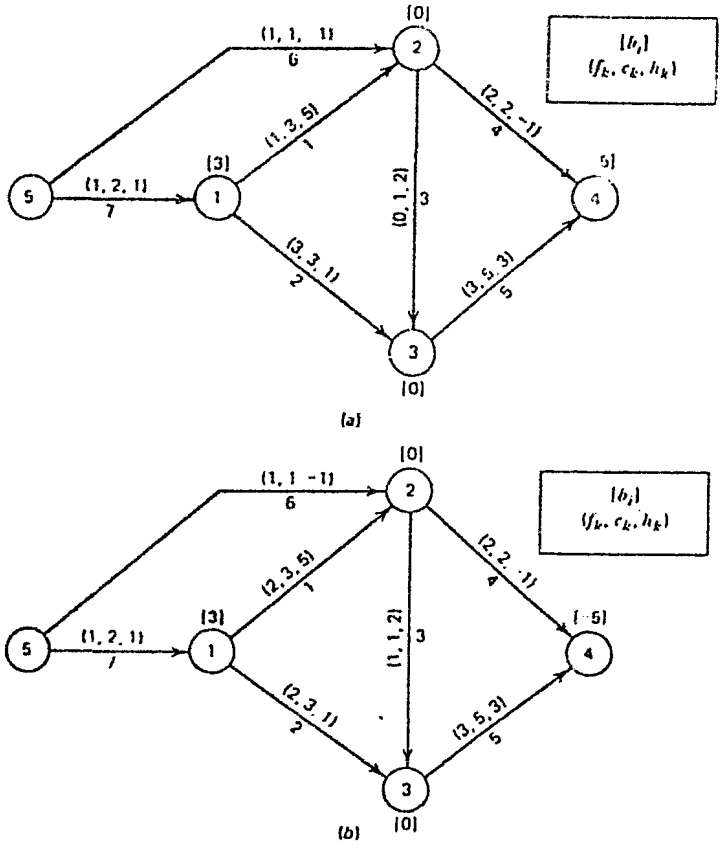
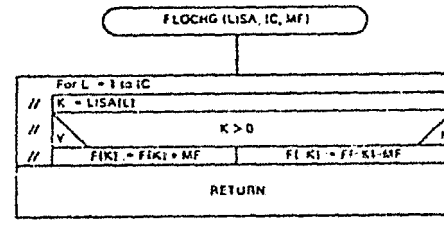


Figura 3.7 (a) Flujos antes del cambio, (b) Flujos después del cambio en el ciclo, M_p = (1, 3,-2), por Δ = 1.

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

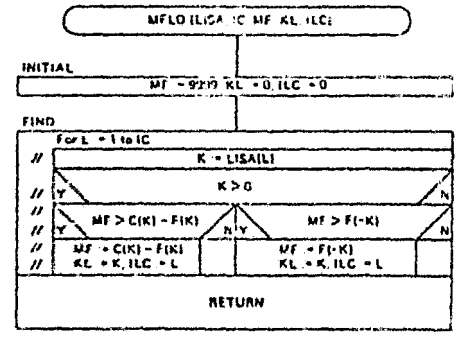
$$\Delta_m = \text{Min.} \left[\underset{k>0}{\text{Min.}} \; C_k - f_k, \; \underset{k<0}{\text{Min}} \; f_{-k} \right]$$

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

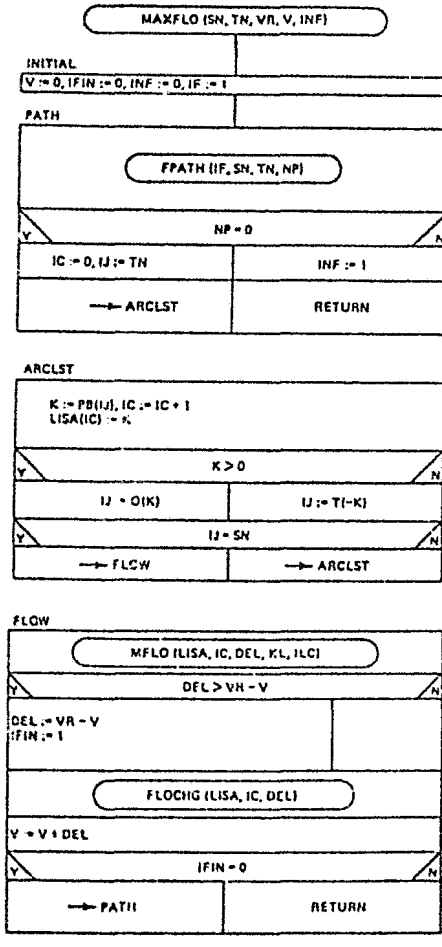
$$M_F = Min \left[\underset{k>0}{Min} \; C_k - f_k \; , \; \underset{k<0}{Min} \; f_{-k} \right]$$

3.4 ALGORITMOS EMPLEADOS PARA UNA SOLUCION NO BASICA.

Los problemas de un flujo especifico y de flujo máximo pueden ser resueltos por algoritmos que no requieren de un árbol básico y se caracterizan por tener un ciclo de arcos cuyos flujos se encuentran estrictamente entre los límites ($0<f_k<C_k$), en la solución no básica óptima. Estos algoritmos sor LABEL1, llamado por el algoritmo FPATH1, 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.

Propósito: 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 puede ser enviado en la trayectoria. Si DEL excede a VR-V, incrementa el flujo

por esta cantidad y termina. De lo contrario incrementa el flujo en la trayectoria y retorna al paso 2.

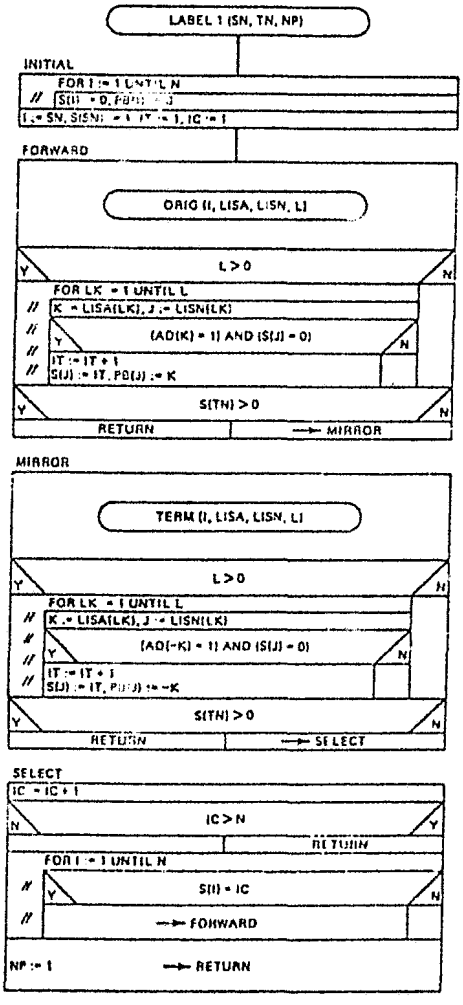
Los principales pasos en este algoritmo son:

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 capacidad, o arcos reflejados con flujo positivo sobre su correspondiente arco hacia adelante.



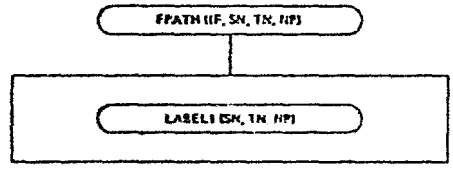
1. (INICIAR) El conjunto S de nodos etiquetados es vacio; s_i=0. Los apuntadores hacia atrás son inicializados en cero; p_{Bi}:=0, luego el nodo fuente SN es incluido en S con la etiqueta S(SN):=1. Los contadores de etiquetas e iteraciones son inicializados a 1: I_c:=1, I_t:=1.
2. (ADELANTE) Encuentra 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 etiquedado, 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 3.8 muestra las iteraciones realizadas por el algoritmo no básico MAXFLO, partiendo desde la red original de la figura 2.1 del capítulo II 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 fueron etiquetados.

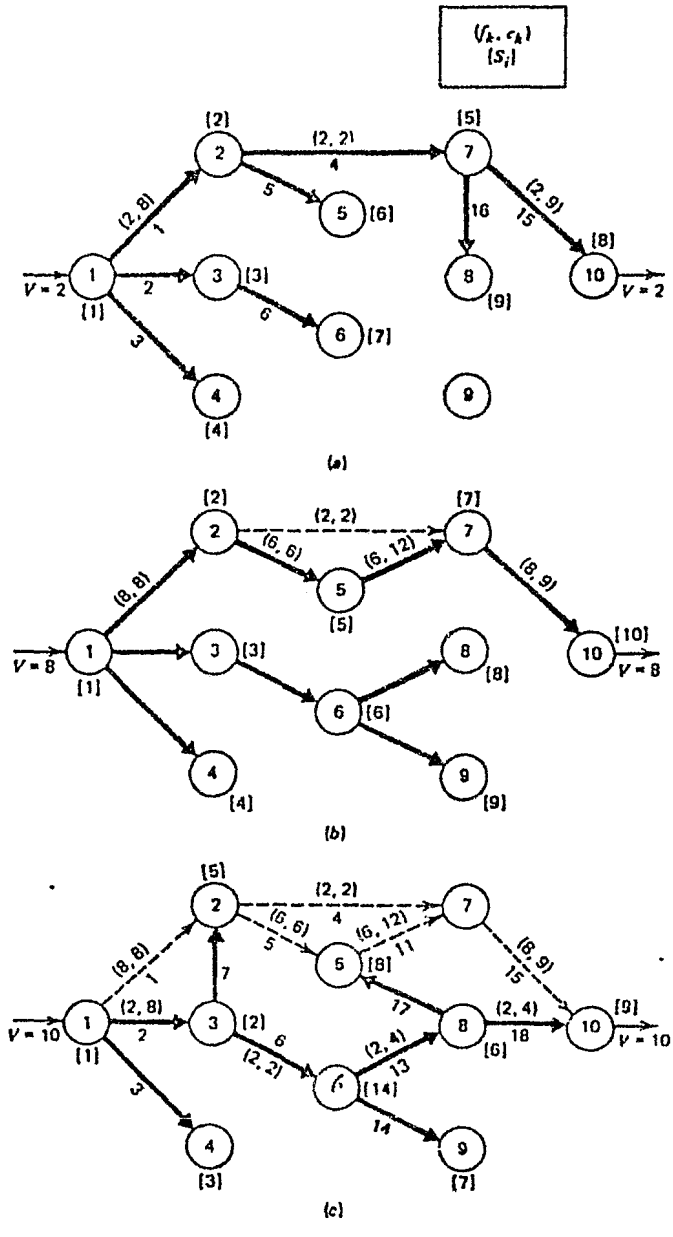


Fig. 3.8 Red con s = 1, t = 10 y v_r = 10

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 3.3. 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

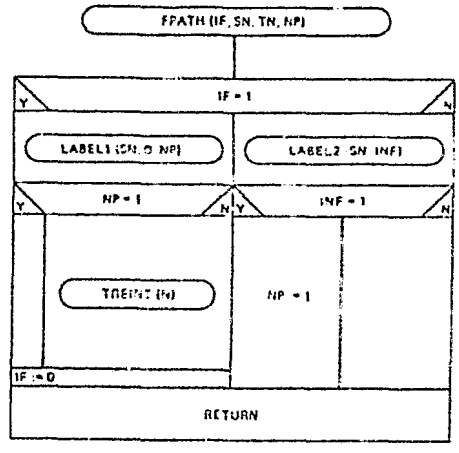
adicional de flujo a dos unidades. Con dos iteraciones más es suficiente para obtener el flujo requerido de 10; como se puede observar en las figuras 3.8 b y 3.8 c.

3.5 ALGORITMOS EMPLEADOS PARA UNA SOLUCION BASICA.

Durante el proceso de una busqueda de una solución básica es necesario mantener un árbol básico en cada iteración. Los algoritmos que son utilizados en este propósito se describen a continuación:

ALGORITMO FPATH2

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



La primera vez el algoritmo usa el procedimiento de etiquetación para encontrar 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.

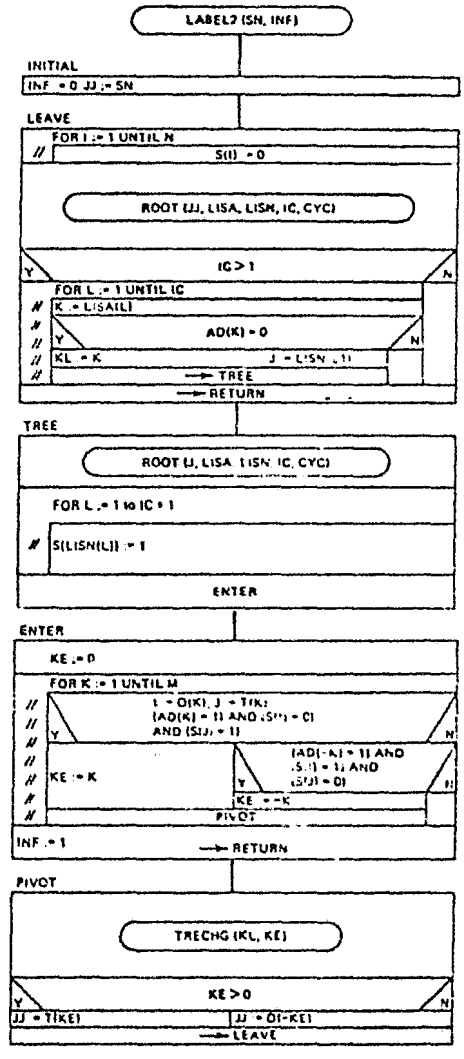
En la primera iteración el algoritmo LABEL1 es usado para encontrar un árbol generador de arcos admisibles enraizado en el nodo fuente. Con el aumento del flujo desde el nodo fuente al nodo sumidero en una trayectoria definida por el árbol, uno o más arcos se vuelven inadmisibles.

Estos son descubiertos y salen del árbol, mientras los arcos admisibles entran para formar el nuevo árbol básico. El proceidmiento 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 $i \in \bar{S}$ y $j \in S$, entonces el arco k entra a la base



y va al paso 5. Si un arco reflejado -k(i,j) es admisible, tal que $j \in \bar{S}$ e $i \in S$, 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 pue dan entrar a la base, termina; el máximo flujo ha sido encontrado.

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 3.9 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.

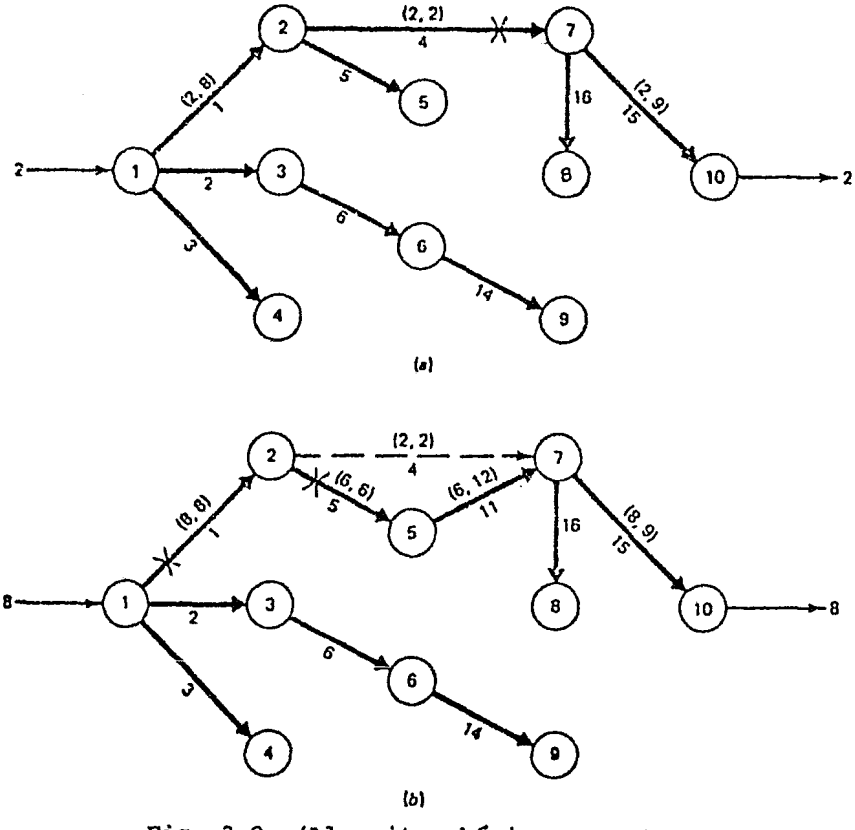
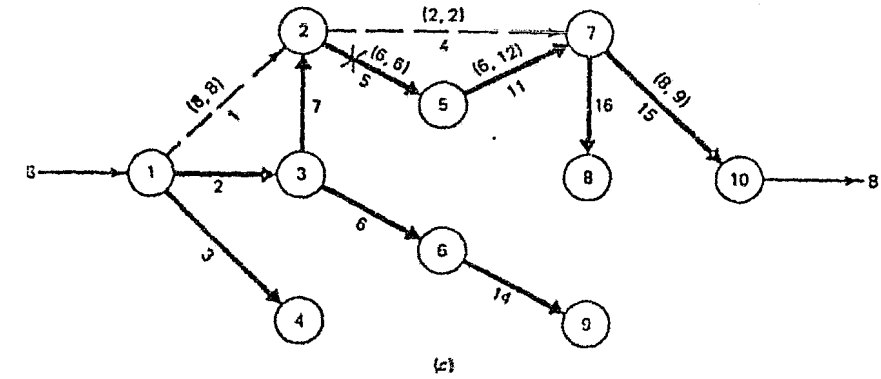
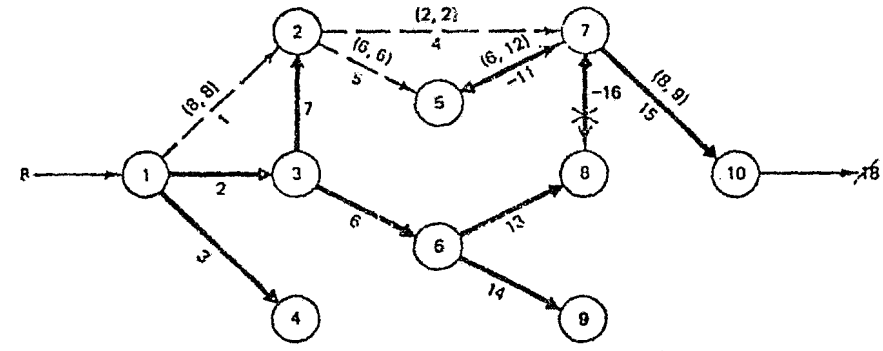


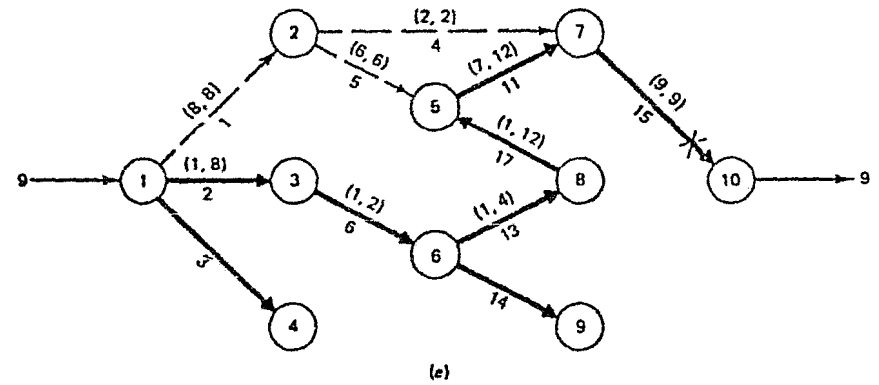
Fig. 3.9 (Algoritmo básico $v_r = 99$



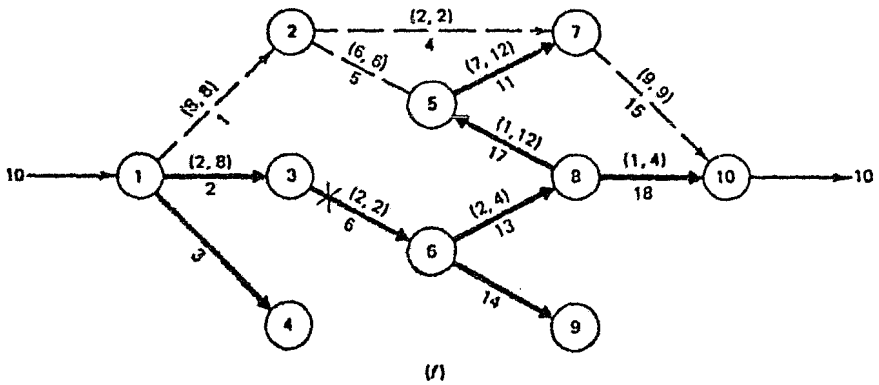
(c)



(d)



(e)



(f)

Figura 3.9 (continuación)

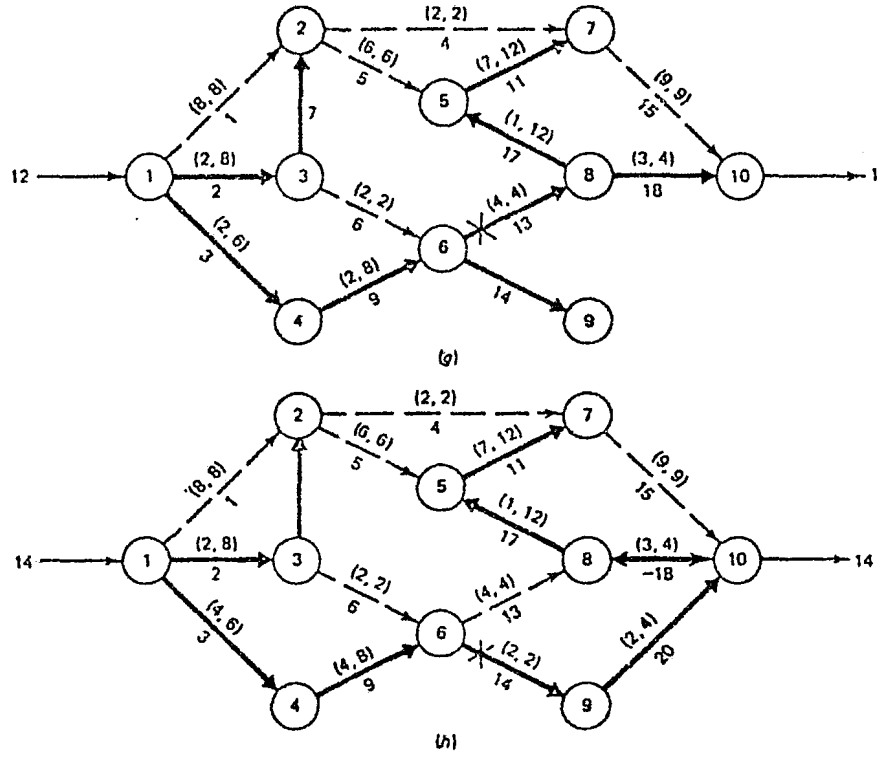


Figura 3.9 (continuación)

Las cabezas gruesas de las flechas indican la orientación de los arcos en la base, las líneas 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 3.9 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 3.9, 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 3.9. Despúes 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 3.9. 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 3.9. 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 indamisible 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 3.9. El arco 13 sale por volverse indamisible 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 admisibles, y el máximo flujo desde el nodo 1 al nodo 10 ha sido encontrado.

CAPITULO IV

ESTRUCTURA Y MANEJO DEL PROGRAMA MAXFLUJO

4.1 CONSIDERACIONES GENERALES

En este capítulo se estudia la forma de encontrar el flujo máximo o la trayectoria de un flujo requerido que puede pasar desde un nodo fuente a un nodo sumidero en una red dada, utilizando un programa de computadora llamado MAXFLUJO, el cual ha sido implantado en el sistema digital VAX11/780 del Centro de Cálculo de la Facultad de Ingeniería.

Este programa fué desarrollado en lenguaje Fortran IV y está integrado por subrutinas que ejecutan los algoritmos empleados para la obtención de soluciones básicas o no-básicas, dependiendo de las necesidades del usuario.

La subrutina que realiza las funciones de lectura y almacenamiento de datos de la red (algoritmo READ), funciona en forma conversacional, esto es, que establece una conversación entre el usuario y la computadora por medio de la terminal utilizada permitiendo de esta manera el acceso al programa de personas que desconoscan el manejo de sistemas computacionales.

El programa puede encontrar soluciones básicas o no básicas de algún problema específico, según lo indique el usuario. Para esto el programa llama a las subrutinas necesarias y entrega los resultados por impresión, pantalla o ambos.

4.2 EJEMPLO ILUSTRATIVO

Con el propósito de mostrar en forma objetiva el funcionamiento del programa MAXFLUJO, se presenta a continuación un ejemplo ilustrativo de un problema de flujo máximo, el cual sera resuelto por medio del programa mostrando la secuencia que este sigue para captar la información de la red, especificando cuando sea conveniente la forma de accesar los datos e interpretando los tipos de soluciones que nos entrega.

Dada una red dirigida, con un parámetro de capacidad en cada arco, se desea encontrar el flujo máximo total en la red, desde un nodo fuente (1) a un nodo sumidero (10). Para accesar la información de la red al programa es necesario etiquetar con numeración consecutiva a todos los nodos y arcos de la red, indicando las capacidades de los arcos entre paréntesis tal como se indica en la red de ejemplo (fig. 4.1)

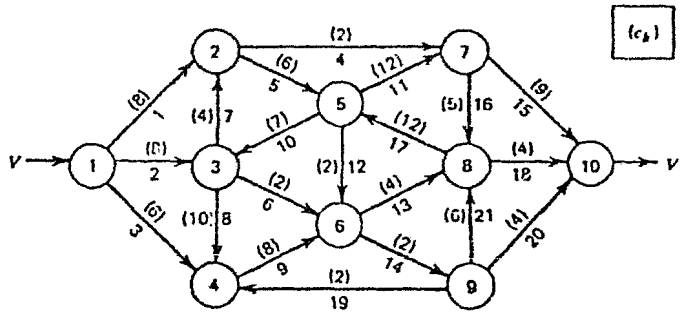


Fig. 4.1 Ejemplo de una red etiquetada

4.3 PREPARACION Y EJECUCION DEL PROGRAMA

Para ejecutar el programa es necesario que el usuario cuente con una clave autorizada por el centro de cálculo de la Facultad de Ingeniería para trabajar en la computadora; una vez que se encuentre en el sistema debe teclear RUN SYS$BIB: MAXFLUJO. EXE Entonces aparece en la pantalla la leyenda "PROGRAMA MAXFLUJO" y una breve explicación de los objetivos que persigue, después solicitara la siguiente información:

"INDIQUE LA SOLUCION QUE DESEA"

1.- Solución no básica

2.- Solución básica

El usuario deberá teclear el número de la opción que requiera y posteriormente apretar la tecla RETURN; de esta forma el programa solicitará la información subsiguiente

"INDIQUE EL NUMERO DE NODOS DE LA RED ORIGINAL
(Debe ser un número entero, menor o igual a 120)"

"EXISTEN DATOS DE LOS NODOS

1.- Si 2.- No "

Si uno o varios nodos tienen flujo externo, cota superior en el flujo de holgura y costo asociado a ese flujo debe de teclear 1 y -- RETURN:; de esa manera le sera preguntado el número de nodos que - contienen datos y la información por cada nodo. Si no existen da-- tos de los nodos debe de teclear 2 y continuar con la información de los arcos.

"INDIQUE EL NUMERO DE ARCOS DE LA RED ORIGINAL"
(Debe ser un número entero, menor o igual a 120)

A continuación el programa pregunta la información para cada arco de la red de la siguiente manera:

"ENTRADA DE INFORMACION REFERENTE AL ARCO NUMERO K
INDIQUE EL NUMERO DEL NODO ORIGINAL DEL ARCO K"

"INDIQUE EL NUMERO DEL NODO TERMINAL DEL ARCO K"

"INDIQUE LA COTA INTERIOR DEL ARCO K"

"INDIQUE LA COTA SUPERIOR DEL ARCO K"

"INDIQUE EL COSTO UNITARIO DEL ARCO K"

Los datos de los límites o cotas y el costo unitario deberán ser números enteros de 5 dígitos máximo; cuando el problema no especifique cota inferior se debe entender como cero.

A partir de este momento, el programa investiga información general en la red.

"INDIQUE EL NUMERO DE NODO FUENTE O RAIZ"

"INDIQUE EL NUMERO DE NODO TERMINAL O SUMIDERO"

"INDIQUE EL FLUJO DESEADO EN LA RED"

El usuario debe dar un flujo requerido que sea entero de 5 dígitos máximo, si este flujo es menor o igual al máximo que puede pasar por la red del nodo fuente al sumidero, entonces el programa encontrará una ruta factible para ese flujo y proporciona el flujo que maneja cada arco. Si por el contrario el flujo requerido es mayor al máximo factible, el programa encontrará el flujo máximo que puede pasar por la red e informará el flujo que corre por cada arco; también

proporcionará una lista de apuntadores hacia atrás indicando los arcos que forman el árbol para el caso de una solución básica.

Si se desea encontrar el flujo máximo de cualquier red basta con indicar como flujo inicial alguno mayor al máximo permisible, esto es, algun número mayor a la suma de los flujos de los arcos que salen del nodo origen.

"EXISTEN FLUJOS INICIALES EN LOS ARCOS?
0-no 1-si "

Si uno o varios arcos tienen un flujo inicial debe de teclear uno, de esta manera el programa preguntará el flujo inicial por cada uno de los arcos. Si el usuario teclea un cero el programa asume que todos los flujos iniciales son cero.

"COMO DESEA SUS RESULTADOS? "

1.- Por pantalla

2.- Por impresora

3.- Por los dos

De esta forma el usuario escoge la forma de salida de sus resultados. Después de proporcionar esta clave el programa procesa los datos y despliega la información por el dispositivo optado. Por pantalla muestra los resultados en la terminal utilizada, por impresora manda a imprimir un archivo llamada SALIDA-LIS por la impresora disponible, si se requiere la opción 3, los resultados son proporcionados por ambos dispositivos.

4.4 SOLUCION DEL EJEMPLO ILUSTRATIVO

Una vez introducida la información de la red en el programa se procesa y posteriormente se encuentra la solución escogida entregándola por el dispositivo indicado. A continuación se muestran los resultados obtenidos por impresora para una solución no básica y una solución básica del ejemplo ilustrado.

En la impresión aparece en primer término la información dada por el usuario de los arcos, acomodada en orden ascendente de acuerdo a su nodo origen. En seguida se indica el máximo flujo que puede pasar por la red y una tabla con los apuntadores hacia atrás de los nodos siendo esta significativa sólo para la solución básica ya que cada apuntador señala el arco que llega al nodo y que a su vez pertenece al árbol básico.

Por último se presenta una tabla que contiene a todos los arcos de la red con información propia de cada uno de ellos, incluyenco la cantidad de flujo que les fué asignado en la solución encontrada.

PROGRAMA MAXFLUJO SOLUCION NO BASICA

REPRESENTACION DE LA RED

NUMERO DE NODOS: 10
NUMERO DE ARCOS: 21

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO NO.	FLUJO EXTERNO
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 NO.	ORIGEN	TERMINAL	CAPACIDAD	COSTO P/U	FLUJO INI.
1	1	2	8	0	0
2	1	3	8	0	0
3	1	4	6	0	0
4	2	7	2	0	0
5	2	5	6	0	0
6	3	6	2	0	0
7	3	2	4	0	0
8	3	4	10	0	0
9	4	6	8	0	0
10	5	3	7	0	0
11	5	7	12	0	0
12	5	6	2	0	0
13	6	8	4	0	0
14	6	9	2	0	0
15	7	10	9	0	0
16	7	8	5	0	0
17	8	5	12	0	0
18	8	10	4	0	0
19	9	4	2	0	0
20	9	10	4	0	0
21	9	8	6	0	0

SOLUCION DEL PROBLEMA DE MAXIMO FLUJO EN LA RED

FLUJO REQUERIDO: 20

DESDE EL NODO FUENTE 1

AL NODO SUMIDERO 10

TODOS LOS FLUJOS INICIALES SON CERO

NO EXISTE RUTA ADICIONAL

MAXIMO FLUJO POSIBLE = 14

NODO NO.	APUNTADOR HACIA ATRAS
1	0
2	7
3	2
4	3
5	0
6	9
7	0
8	0
9	0
10	0

ARCO NO.	ORIGEN	TERMINAL	FLUJO	CAPACIDAD	COSTO P/U	COSTO 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	6	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	8	9	0	0
16	7	8	0	5	0	0
17	8	5	0	12	0	0
18	8	10	4	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

PROGRAMA MAXFLUJO SOLUCION BASICA

REPRESENTACION DE LA RED

NUMERO DE NODOS: 10
NUMERO DE ARCOS: 21

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO NO.	FLUJO EXTERNO
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 NO.	ORIGEN	TERMINAL	CAPACIDAD	COSTO P/U	FLUJO INI.
1	1	2	8	0	0
2	1	3	8	0	0
3	1	4	6	0	0
4	2	7	2	0	0
5	2	5	6	0	0
6	3	6	2	0	0
7	3	2	4	0	0
8	3	4	10	0	0
9	4	6	8	0	0
10	5	3	7	0	0
11	5	7	12	0	0
12	5	6	2	0	0
13	6	8	4	0	0
14	6	9	2	0	0
15	7	10	9	0	0
16	7	8	5	0	0
17	8	5	12	0	0
18	8	10	4	0	0
19	9	4	2	0	0
20	9	10	4	0	0
21	9	8	6	0	0

SOLUCION DEL PROBLEMA DE MAXIMO FLUJO EN LA RED

FLUJO REQUERIDO: 20

DESDE EL NODO FUENTE 1

AL NODO SUMIDERO 10

TODOS LOS FLUJOS INICIALES SON CERO

NO EXISTE RUTA ADICIONAL

MAXIMO FLUJO POSIBLE = 14

NODO NO.	APUNTADOR HACIA ATRAS
1	0
2	7
3	2
4	3
5	17
6	9
7	11
8	-18
9	14
10	20

ARCO NO.	ORIGEN	TERMINAL	FLUJO	CAPACIDAD	COSTO P/U	COSTO 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

CAPITULO V

EJEMPLOS DE APLICACION

En este capítulo se presentan tres ejemplos de aplicación con la finalidad de mostrar algunas áreas en donde puede utilizarse el programa, además de aclarar la manera de interpretar los datos de los problemas e introducirlos a la computador. Para cada ejemplo se obtuvo una solución no básica en la que se encuentra una descripción detallada de la red, y se indica el flujo máximo que se puede enviar del nodo fuente al nodo sumidero así como proporcionar la información del flujo que pasa por cada uno de los arcos.

También se presenta una solución básica, la cual requiere de más tiempo de máquina pero obteniendo como resultado un árbol formado por los arcos indicados en la lista de apuntadores hacia atrás.

De esta forma se pretnede proporcionar suficiente información para utilizar el programa MAXFLUJO como una herramienta más para la solución de los problemas de redes.

1. OPERACION CUARENTENA

Un servicio contra plagas a determinado que la emigración anual del gusano volador que barrena el maíz empezará nuevamente este año. Al menos que se tomen medidas exhaustivas, los granjeros sufrirán serias pérdidas económicas. En los últimos años las rutas de emigración del gusano barrenador de maíz han sido extensamente delineadas. Usando esta información el servicio determinará la forma menos costosa para segurarse que cada gusano barrenador de maíz se someterá a una aplicación de insecticida.

En vista de que el costo de "espolvorear" cualquier ruta de emigración es directamente proporcional al número máximo de gusanos barrenadores que pudieron usar la ruta, el esquema que representa las rutas de emigración señalado en la figura 5.1 fué usado para contestar la pregunta formulada anteriormente.

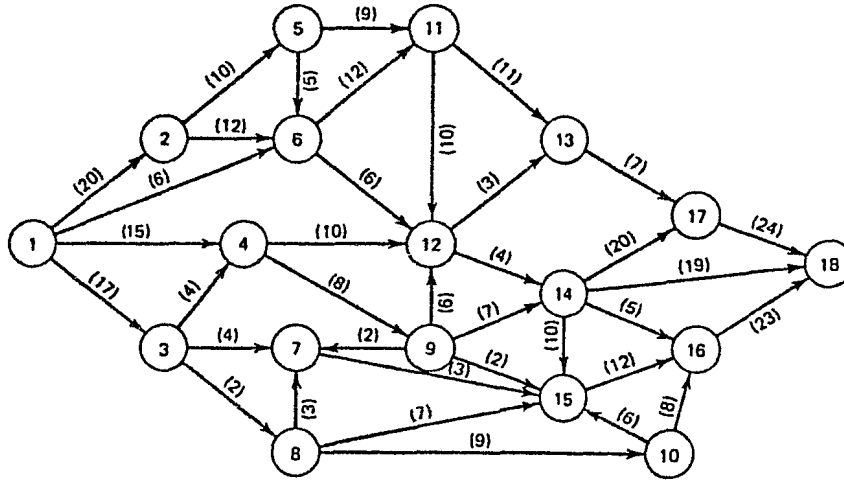


Fig. 5.1 Rutas de emigración del gusano barrenador

En la figura 5.1 los nodos son puntos de unión para los arcos, que representan los senderos de emigración. Los nodos 1 y 18 son los puntos del principio y final de la emigración respectivamente. Las capacidades de arcos son el número máximo de barrenadores (en miles) que se espera usen determinada ruta. Estas capacidades se basan principalmente en información histórica.

El servicio contra plagas formuló este problema como un problema de máximo flujo, que al solucionarse redituó una mínima cortadura o sea un juego de rutas de emigración de mínima capacidad total, que si se retiraran de la red de información de la figura 5.1 impedirían a cualquier gusano barrenador llegar al nodo 18.

Todas las rutas de emigración en esta mínima cortadura recibieron un amplio espolvoréo de insecticida efectuado por el avión del servicio contra plagas en las cosechas.

PROGRAMA MAXFLUJO SOLUCION NO BASICA

REPRESENTACION DE LA RED

NUMERO DE NODOS: 18
NUMERO DE ARCOS: 37

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO NO.	FLUJO EXTERNO
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

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO NO.	ORIGEN	TERMINAL	CAPACIDAD	COSTO P/U	FLUJO INI.
1	1	2	20	0	0
2	1	3	17	0	0
3	1	4	15	0	0
4	1	6	6	0	0
5	2	5	10	0	0
6	2	6	12	0	0
7	3	4	4	0	0
8	3	7	4	0	0
9	3	8	2	0	0
10	4	12	10	0	0
11	4	9	8	0	0
12	5	6	5	0	0
13	5	11	9	0	0
14	6	11	12	0	0
15	6	12	6	0	0
16	7	15	3	0	0
17	8	7	3	0	0
18	8	15	7	0	0
19	8	10	9	0	0
20	9	7	2	0	0
21	9	12	6	0	0
22	9	14	7	0	0
23	9	15	2	0	0
24	10	15	6	0	0
25	10	16	8	0	0
26	11	12	10	0	0
27	11	13	11	0	0
28	12	13	3	0	0
29	12	14	4	0	
30	13	17	7	0	

31	14	15	10	0		0
32	14	16	5	0	70	0
33	14	17	20	0		0
34	14	18	19	0		0
35	15	16	12	0		0
36	16	18	23	0		0
37	17	18	24	0		0

SOLUCION DEL PROBLEMA DE MAXIMO FLUJO EN LA RED

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 NO.	APUNTADOR HACIA ATRAS
1	0
2	1
3	2
4	7
5	5
6	4
7	8
8	0
9	0
10	0
11	14
12	15
13	27
14	0
15	0
16	0
17	0
18	0

ARCO NO.	ORIGEN	TERMINAL	FLUJO	CAPACIDAD	COSTO P/U	COSTO 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	2	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	71	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

PROGRAMA MAXFLUJO SOLUCION BASICA

REPRESENTACION DE LA RED

NUMERO DE NODOS: 18
NUMERO DE ARCOS: 37

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO NO.	FLUJO EXTERNO
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

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO NO.	ORIGEN	TERMINAL	CAPACIDAD	COSTO P/U	FLUJO INI.
1	1	2	20	0	0
2	1	3	17	0	0
3	1	4	15	0	0
4	1	6	6	0	0
5	2	5	10	0	0
6	2	6	12	0	0
7	3	4	4	0	0
8	3	7	4	0	0
9	3	8	2	0	0
10	4	12	10	0	0
11	4	9	8	0	0
12	5	6	5	0	0
13	5	11	9	0	0
14	6	11	12	0	0
15	6	12	6	0	0
16	7	15	3	0	0
17	8	7	3	0	0
18	8	15	7	0	0
19	8	10	9	0	0
20	9	7	2	0	0
21	9	12	6	0	0
22	9	14	7	0	0
23	9	15	2	0	0
24	10	15	6	0	0
25	10	16	8	0	0
26	11	12	10	0	0
27	11	13	11	0	0
28	12	13	3	0	0
29	12	14	4	0	0
30	13	17	7	0	0

31	14	15	10	0	0
32	14	16	5	0	0
33	14	17	20	0	0
34	14	18	19	0	0
35	15	16	12	0	0
36	16	18	23	0	0
37	17	18	24	0	0

SOLUCION DEL PROBLEMA DE MAXIMO FLUJO EN LA RED

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 NO.	APUNTADOR HACIA ATRAS
1	0
2	1
3	2
4	7
5	5
6	4
7	8
8	-18
9	-23
10	19
11	14
12	10
13	27
14	-34
15	-35
16	-36
17	30
18	37

ARCO NO.	ORIGEN	TERMINAL	FLUJO	CAPACIDAD	COSTO P/U	COSTO 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	
18	8	15	2	7	0	

19	8	10	0	9	0		0
20	9	7	0	2	0	74	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

2. PROBLEMA DE ENCUADERNACION

El presidente de una editorial esta interesado en la productividad de su empresa. Existen nueve pasos para encuadernar un libro: (1) abrir cajas, (2) cotejar las páginas, (3) aparejar las hojas, (4) pegar dorsos, (5) pegar las pastas, (6) realzar las pastas con títulos, (7) colocar los libros en las cajas, (8) pegar etiquetas de envío, y (9) cargar en los camiones.

Usando la información anterior, el supervisor de la planta ha sido encargado con la responsabilidad de determinar el número máximo de libros que pueden ser encuadernados al día, y a la vez, hacer recomendaciones que aumenten esta producción al menor costo posible, una vez determinada la fase donde se satura la línea de producción.

La figura 5.2 proporciona la red de información alcanzada por el supervisor de acuerdo con las sugerencias hechas por él a su jefe. Los nodos representan los nueve pasos seguidos para encuadernar un libro y los arcos son estaciones de trabajo asociadas con sus nodos de origen.

Las capacidades de los arcos son las capacidades de las estaciones de trabajo en libros por día.

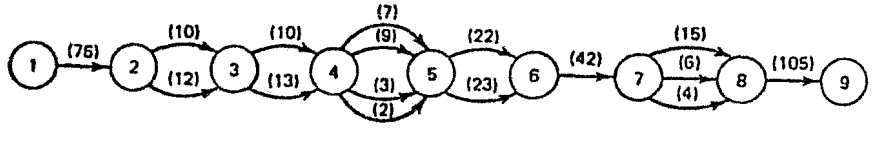


Fig. 5.2 Proceso de encuadernación de un libro

PROGRAMA MAXFLUJO SOLUCION NO BASICA

REPRESENTACION DE LA RED

NUMERO DE NODOS: 9
NUMERO DE ARCOS: 16

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO NO.	FLUJO EXTERNO
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO NO.	ORIGEN	TERMINAL	CAPACIDAD	COSTO P/U	FLUJO INI.
1	1	2	76	0	0
2	2	3	10	0	0
3	2	3	12	0	0
4	3	4	10	0	0
5	3	4	13	0	0
6	4	5	7	0	0
7	4	5	9	0	0
8	4	5	3	0	0
9	4	5	2	0	0
10	5	6	22	0	0
11	5	6	23	0	0
12	6	7	42	0	0
13	7	8	15	0	0
14	7	8	6	0	0
15	7	8	4	0	0
16	8	9	105	0	0

SOLUCION DEL PROBLEMA DE MAXIMO FLUJO EN LA RED

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 NO.	APUNTADOR HACIA ATRAS
1	0
2	1
3	3
4	5

5
6
7
8
9

0
0
0
0
0

77

ARCO NO.	ORIGEN	TERMINAL	FLUJO	CAPACIDAD	COSTO P/U	COSTO 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

PROGRAMA MAXFLUJO SOLUCION BASICA

REPRESENTACION DE LA RED

NUMERO DE NODOS: 9
NUMERO DE ARCOS: 16

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO NO.	FLUJO EXTERNO
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO NO.	ORIGEN	TERMINAL	CAPACIDAD	COSTO P/U	FLUJO INI.
1	1	2	76	0	0
2	2	3	10	0	0
3	2	3	12	0	0
4	3	4	10	0	0
5	3	4	13	0	0
6	4	5	7	0	0
7	4	5	9	0	0
8	4	5	3	0	0
9	4	5	2	0	0
10	5	6	22	0	0
11	5	6	23	0	0
12	6	7	42	0	0
13	7	8	15	0	0
14	7	8	6	0	0
15	7	8	4	0	0
16	8	9	105	0	0

SOLUCION DEL PROBLEMA DE MAXIMO FLUJO EN LA RED

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 NO.	APUNTADOR HACIA ATRAS
1	0
2	1
3	3
4	5

5
6
7
8
9

9
10
12
14
16

79

ARCO NO.	ORIGEN	TERMINAL	FLUJO	CAPACIDAD	COSTO F/U	COSTO 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

3. DESALOJO DE AUTOS EN UNA RED

En la figura 5.3 se tiene una red donde los arcos representan calles en un solo sentido que salen desde un estacionamiento de un estadio. Cada calle tiene como se indica una capacidad en carros por minuto. Es necesario desarrollar un modelo de flujo máximo de tráfico para desconges-tionar el estacionamiento después de algún juego y en base a los resultados se tomen las medidas pertinentes para agilizar la salida de autos.

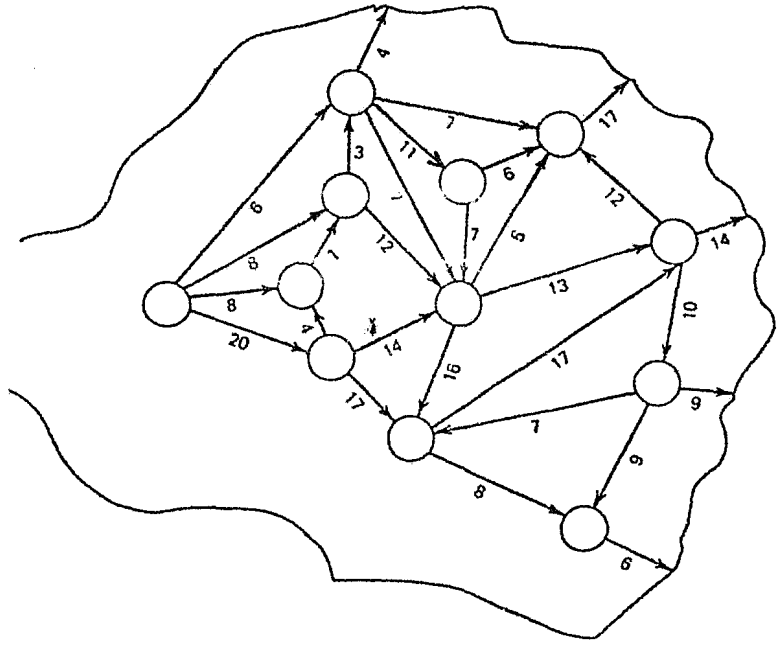


Fig. 5.3 Red de calles

PROGRAMA MAXFLUJO SOLUCION NO BASICA

REPRESENTACION DE LA RED

NUMERO DE NODOS: 13
NUMERO DE ARCOS: 29

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO NO.	FLUJO EXTERNO
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

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO NO.	ORIGEN	TERMINAL	CAPACIDAD	COSTO P/U	FLUJO INI.
1	1	2	6	0	0
2	1	3	8	0	0
3	1	4	8	0	0
4	1	5	20	0	0
5	2	13	4	0	0
6	2	7	7	0	0
7	2	9	17	0	0
8	2	6	11	0	0
9	3	2	3	0	0
10	3	7	12	0	0
11	4	3	7	0	0
12	5	4	4	0	0
13	5	7	14	0	0
14	5	8	17	0	0
15	6	9	6	0	0
16	6	7	7	0	0
17	7	8	16	0	0
18	7	9	5	0	0
19	7	10	13	0	0
20	8	10	17	0	0
21	8	12	8	0	0
22	9	13	17	0	0
23	10	9	12	0	0
24	10	11	10	0	0
25	10	13	14	0	0
26	11	8	7	0	0
27	11	12	9	0	0
28	11	13	9	0	0
29	12	13	6	0	0

SOLUCION DEL PROBLEMA DE MAXIMO FLUJO EN LA RED

FLUJO REQUERIDO: 50

DESDE EL NODO FUENTE 1

AL NODO SUMIDERO 13

TODOS LOS FLUJOS INICIALES SON CERO

NO EXISTE RUTA ADICIONAL

MAXIMO FLUJO POSIBLE = 41

NODO NO.	APUNTADOR HACIA ATRAS
1	0
2	0
3	0
4	3
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0

ARCO NO.	ORIGEN	TERMINAL	FLUJO	CAPACIDAD	COSTO P/U	COSTO FLUJO
1	1	2	6	6	0	0
2	1	3	8	8	0	0
3	1	4	7	8	0	0
4	1	5	20	20	0	0
5	2	13	4	4	0	0
6	2	7	0	7	0	0
7	2	9	5	17	0	0
8	2	6	0	11	0	0
9	3	2	3	3	0	0
10	3	7	12	12	0	0
11	4	3	7	7	0	0
12	5	4	0	4	0	0
13	5	7	13	14	0	0
14	5	8	7	17	0	0
15	6	9	0	6	0	0
16	6	7	0	7	0	0
17	7	8	7	16	0	0
18	7	9	5	5	0	0
19	7	10	13	13	0	0
20	8	10	8	17	0	0
21	8	12	6	8	0	0
22	9	13	17	17	0	0
23	10	9	7	12	0	0
24	10	11	0	10	0	0
25	10	13	14	14	0	0
26	11	8	0	7	0	0
27	11	12	0	9	0	0
28	11	13	0	9	0	0
29	12	13	6	6	0	0

COSTO TOTAL= 0

PROGRAMA MAXFLUJO SOLUCION BASICA

REPRESENTACION DE LA RED

NUMERO DE NODOS: 13
NUMERO DE ARCOS: 29

PARAMETROS TRANSFORMADOS DE LOS NODOS

NODO NO.	FLUJO EXTERNO
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

PARAMETROS TRANSFORMADOS DE LOS ARCOS

ARCO NO.	ORIGEN	TERMINAL	CAPACIDAD	COSTO P/U	FLUJO INI.
1	1	2	6	0	0
2	1	3	8	0	0
3	1	4	8	0	0
4	1	5	20	0	0
5	2	13	4	0	0
6	2	7	7	0	0
7	2	9	17	0	0
8	2	6	11	0	0
9	3	2	3	0	0
10	3	7	12	0	0
11	4	3	7	0	0
12	5	4	4	0	0
13	5	7	14	0	0
14	5	8	17	0	0
15	6	9	6	0	0
16	6	7	7	0	0
17	7	8	16	0	0
18	7	9	5	0	0
19	7	10	13	0	0
20	8	10	17	0	0
21	8	12	8	0	0
22	9	13	17	0	0
23	10	9	12	0	0
24	10	11	10	0	0
25	10	13	14	0	0
26	11	8	7	0	0
27	11	12	8	0	0
28	11	13	8	0	0
29	12	13	6	0	0

SOLUCION DEL PROBLEMA DE MAXIMO FLUJO EN LA RED

FLUJO REQUERIDO: 50

DESDE EL NODO FUENTE 1

AL NODO SUMIDERO 13

TODOS LOS FLUJOS INICIALES SON CERO

NO EXISTE RUTA ADICIONAL

MAXIMO FLUJO POSIBLE = 41

NODO NO.	APUNTADOR HACIA ATRAS
1	0
2	-7
3	-10
4	3
5	4
6	8
7	13
8	14
9	23
10	20
11	24
12	21
13	28

ARCO NO.	ORIGEN	TERMINAL	FLUJO	CAPACIDAD	COSTO P/U	COSTO FLUJO
1	1	2	6	6	0	0
2	1	3	8	8	0	0
3	1	4	7	8	0	0
4	1	5	20	20	0	0
5	2	13	4	4	0	0
6	2	7	0	7	0	0
7	2	9	5	17	0	0
8	2	6	0	11	0	0
9	3	2	3	3	0	0
10	3	7	12	12	0	0
11	4	3	7	7	0	0
12	5	4	0	4	0	0
13	5	7	6	14	0	0
14	5	8	14	17	0	0
15	6	9	0	6	0	0
16	6	7	0	7	0	0
17	7	8	0	16	0	0
18	7	9	5	5	0	0
19	7	10	13	13	0	0
20	8	10	14	17	0	0
21	8	12	0	8	0	0
22	9	13	17	17	0	0
23	10	9	7	12	0	0
24	10	11	6	10	0	0
25	10	13	14	14	0	0
26	11	8	0	7	0	0
27	11	12	0	9	0	0
28	11	13	6	9	0	0
29	12	13	0	6	0	0

COSTO TOTAL= 0

A P E N D I C E A

A.1 LISTADO DEL PROGRAMA MAXFLUJO

```
      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)
      COMMON /I/LT(0:120)/J/PO(0:120)/K/C(0:120)/L/H(0:120)
      COMMON /M/B(0:120)/N/CL(0:120)/O/F(0:120)
      COMMON /R/PD(0:120)/Q/PP(0:120)
      COMMON ITIPO,LISTAS
      INTEGER PP,PD,AD
      INTEGER O,T,PB,PF,PR,PT,LT,PO,C,H,B,CL,F,PI
      INTEGER VR,SN,TN,V
      CHARACTER ARRI*6,ABAJ*6,BORR*6,ENME*6,SUPE*6,ALAR*6
      CALL ASSIGN (2,'SALIDA.LIS',10)
      CALL ASSIGN (5,'SYS$COMMAND',11)
      ARRI=CHAR(27)//'*3'
      SUPE=CHAR(27)//'[0;0H'
      ABAJ=CHAR(27)//'*4'
      ENME=CHAR(27)//'[7;0H'
      BORR=CHAR(27)//'[2J'
      ALAR=CHAR(27)//'*6'
C
C
C
C
C     ************************************************************
C     *                                                          *
C     *              PROGRAMA PRINCIPAL: MAXFLUJO                *
C     *                                                          *
C     *                 ASOCIADO CON LA TESIS:                   *
C     *                                                          *
C     * " EL PROBLEMA DE FLUJO MAXIMO EN UNA RED Y SU SOLUCION   *
C     * POR MEDIO DE ALGORITMOS IMPLEMENTADOS EN COMPUTADORA "   *
C     *                                                          *
C     *  ESTE PROGRAMA HA SIDO INTEGRADO Y MODIFICADO POR EL     *
C     *  ALUMNO RICARDO J. HERNANDEZ LOYOLA DE ACUERDO A LOS     *
C     *  ALGORITMOS UTILIZADOS EN EL LIBRO " NETWORK FLOW        *
C     *  PROGRAMMING " DE JENSEN, P. A. Y BARNES, J. W. 1980     *
C     *  (CAPITULOS 4,5 Y 6) Y SERA INCLUIDO COMO PARTE DE SU    *
C     *  TESIS PREVIA A LA OBTENCION DEL TITULO PROFESIONAL      *
C     *  EN LA CARRERA DE INGENIERO MECANICO ELECTRICISTA DE      *
C     *  LA FACULTAD DE INGENIERIA.                              *
C     *                                                          *
C     *   UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO (U.N.A.M.)     *
C     *          CIUDAD UNIVERSITARIA,FEBRERO DE 1983            *
C     *                                                          *
C     *    PROPOSITO: RESOLVER EL PROBLEMA DE FLUJO MAXIMO EN    *
C     *  REDES PUDIENDO UTILIZAR EL METODO NO BASICO O EL ME-    *
C     *  TODO BASICO DEPENDIENDO DE LAS NECESIDADES DEL USUA-    *
C     *  RIO.                                                    *
C     *  EL PROGRAMA ESTA INTEGRADO POR UNA SERIE DE SUBRUTI-    *
C     *  NAS QUE SON EJECUTADAS DEPENDIENDO DEL TIPO DE SOLU-    *
C     *  CION QUE SE REQUIERA.  LA ENTRADA DE DATOS DE LA RED    *
C     *  ES REALIZADA POR UNA SUBRUTINA QUE FUNCIONA EN FORMA    *
C     *  CONVERSACIONAL CON EL USUARIO POR MEDIO DE LA TERMI-    *
C     *  NAL UTILIZADA.  LOS RESULTADOS SE PUEDEN OBTENER POR    *
C     *  TERMINAL O IMPRESORA.                                   *
C     *                                                          *
C     ************************************************************
C
C
C
      TYPE*,BORR
      TYPE*,ENME
      TYPE*,ARRI,' ********************************'
      TYPE*,ABAJ,' ********************************'
```

```
      TYPE*,ARRI,' *                                *'
      TYPE*,ABAJ,' *                                *'
      TYPE*,ARRI,' *       PROGRAMA  MAXFLUJO       *'
      TYPE*,ABAJ,' *       PROGRAMA  MAXFLUJO       *'
      TYPE*,ARRI,' *                                *'
      TYPE*,ABAJ,' *                                *'
      TYPE*,ARRI,' **********************************'
      TYPE*,ABAJ,' **********************************'
      DO 6 I=1,6
6     TYPE*,' '
      DO 10 I=1,500
      DO 10 J=1,1000
10    NADA=J**2
      TYPE*,BORR
      TYPE*,SUPE
      TYPE*,' '
      TYPE*,ALAR,' El programa MAXFLUJO encuentra la '
      TYPE*,ALAR,' trayectoria para un flujo requeri-'
      TYPE*,ALAR,' do o el maximo flujo para una red '
      TYPE*,ALAR,' dirigida con capacidad en cada ar-'
      TYPE*,ALAR,' co.'
      TYPE*,' '
      TYPE*,ALAR,' La solucion puede ser basica o no '
      TYPE*,ALAR,' basica dependiendo de las necesi- '
      TYPE*,ALAR,' dades del usuario.'
      TYPE*,' '
      TYPE*,ALAR,' Este programa esta intesrado por  '
      TYPE*,ALAR,' subrutinas que son ejecutadas de- '
      TYPE*,ALAR,' pendiendo de la solucion deseada. '
      TYPE*,' '
      TYPE*,ALAR,' La entrada de datos de la red es  '
      TYPE*,ALAR,' por medio de una subrutina de lec-'
      TYPE*,ALAR,' tura que funciona en forma conver-'
      TYPE*,ALAR,' sacional con el usuario, a traves '
      TYPE*,ALAR,' de la terminal utilizada ; los re-'
      TYPE*,ALAR,' sultados se pueden obtener por im-'
      TYPE*,ALAR,' presora, terminal o ambos medios. '
      TYPE*,' '
      DO 210 I=1,6000
      DO 210 J=1,1000
210   NADA=J**2
      TYPE*,BORR
      TYPE*,SUPE
      TYPE*,'INDIQUE LA SOLUCION QUE DESEA'
      TYPE*,'    1 -SOLUCION NO BASICA'
      TYPE*,'    2 -SOLUCION BASICA'
21    READ (5,8,ERR=16) ITIPO
8     FORMAT(I1)
      IF (ITIPO.LT.1.OR.ITIPO.GT.2) GO TO 16
      GO TO 22
16    TYPE*,'LA CLAVE DEBE SER 1 O 2, FAVOR DE REPETIRLA'
      GO TO 21
22    CALL READ
      TYPE*,BORR
      TYPE*,SUPE
      N=N-1
      TYPE*,'INDIQUE EL NUMERO DEL NODO FUENTE O RAIZ'
901   READ(5,88,ERR=900)DA1
88    FORMAT(F3.0)
      SN=DA1
      IF (SN.GT.N) GO TO 900
      GO TO 902
900   WRITE (5,950)N
```

```
930     FORMAT (  DEBE SER MENOR O IGUAL AL NODO ',I3)
        GO TO 90.                                                       88
902     TYPE*,' '
        TYPE*,'INDIQUE EL NUMERO DEL NODO TERMINAL O SUMIDERO'
903     READ(5,88,ERR=905)DA1
        TN=DA1
        IF (TN.GT.N) GO TO 905
        GO TO 906
905     WRITE (5,951)N
951     FORMAT (' DEBE SER MENOR O IGUAL AL NODO ',I3)
        GO TO 903
908     TYPE*,' '
        TYPE*,'INDIQUE EL FLUJO DESEADO EN LA RED'
        READ (5,965,ERR=908)DA1
965     FORMAT (F3.0)
        VR=DA1
        TYPE*,' '
        TYPE*,'EXISTEN FLUJOS INICIALES EN LOS ARCOS ?'
        TYPE*,'            0 -NO    1 -SI'
912     READ (5,8,ERR=915)IFLOW
        IF (IFLOW.LT.0.OR.IFLOW.GT.1) GO TO 915
        GO TO 916
915     TYPE*,' EL NUMERO DEBE SER 0 O 1'
        GO TO 912
916     IF (IFLOW.NE.1) GO TO 942
        TYPE*,BORR
        TYPE*,SUPE
        TYPE*,'INDIQUE LOS FLUJOS INICIALES FACTIBLES EN LOS ARCOS'
        TYPE*,' '
        TYPE*,' '
        DO 984 I=1,M
        WRITE (5,986)I
986     FORMAT (' ESCRIBA EL FLUJO INICIAL PARA EL ARCO  ,I3)
        READ (5,965,ERR=916)DA1
        F(I)=DA1
984     TYPE*,' '
942     TYPE*,BORR
        TYPE*,SUPE
        TYPE*,'COMO DESEA SUS RESULTADOS ?'
        TYPE*,'      1 -POR PANTALLA
        TYPE*,'      2 -POR IMPRESORA'
        TYPE*,'      3 -POR LOS DOS'
        READ (5,8,ERR=942)LISTAS
        IF (LISTAS.LT.0.OR.LISTAS.GT.3) GO TO 984
        IF (LISTAS.EQ.2) GO TO 708
        TYPE*,BORR
        TYPE*,SUPE
        IF (ITIPO.EQ.2) GO TO 600
        TYPE*,ALAR,'PROGRAMA MAXFLUJO SOLUCION NO BASICA'
        GO TO 601
600     TYPE*,ALAR,' PROGRAMA MAXFLUJO SOLUCION BASICA'
601     TYPE*,' '
        TYPE*,'                                REPRESENTACION DE LA RED
        TYPE*,' '
        WRITE (5,610)N
610     FORMAT (' '                                NUMERO DE NODOS:  ,I3)
        WRITE (5,611)M
611     FORMAT ('                                  NUMERO DE ARCOS:  ,I3)
        TYPE*,'
        TYPE*,'
     1  '                        PARAMETROS TRANSFORMADOS DE LOS NODOS
        TYPE*,'                          NODO NO.          FLUJO EXTERNO'
        DO 613 I=1,N
```

```
      IF (I.EQ.13.OR.I.EQ.32.OR.I.EQ.51) CALL BORRA
      IF (I.NE.13.OR.I.NE.32.OR.I.NE.51) GO TO 613
      TYPE*,
     1  '                    PARAMETROS TRANSFORMADOS DE LOS NODOS'
      TYPE*,'                          NODO NO.          FLUJO EXTERNO'
613   WRITE (5,612)I,B(I)
612   FORMAT (26X,I3,17X,I6)
      TYPE*,' '
      TYPE*,' PARA CONTINUAR TECLEE RETURN'
      ACCEPT 625,RETU
625   FORMAT (A1)
      TYPE*,BORR
      TYPE*,SUPE
      TYPE*,'                              REPRESENTACION DE LA RED'
      TYPE*,' '
      TYPE*,
     1  '                    PARAMETROS TRANSFORMADOS DE LOS ARCOS'
      WRITE (5,648)
648   FORMAT (3X,'ARCO  NO.',5X,'ORIGEN',6X,'TERMINAL',5X,
     1 'CAPACIDAD',4X,'COSTO P/U',4X,'FLUJO INI.')
      DO 627 I=1,M
      IF (I.EQ.18.OR.I.EQ.37.OR.I.EQ.56) CALL BORRA
      IF (I.NE.18.OR.I.NE.37.OR.I.NE.56) GO TO 627
      TYPE*,
     1  '                    PARAMETROS TRANSFORMADOS DE LOS ARCOS'
      WRITE (5,648)
627   WRITE (5,628)I,O(I),T(I),C(I),H(I),F(I)
628   FORMAT (5X,I3,2(10X,I3),7X,I8,2(5X,I8))
      TYPE*,' '
      TYPE*,' PARA CONTINUAR TECLEE RETURN'
      ACCEPT 625,RETU
      TYPE*,BORR
      TYPE*,SUPE
      WRITE (5,182)VR,SN,TN
182   FORMAT (///,16X,
     1  ' SOLUCION DEL PROBLEMA DE MAXIMO FLUJO EN LA RED',
     2 //,'    FLUJO REQUERIDO: ',I4,//,'    DESDE EL NODO FUENTE ',
     3 I3,//,'    AL NODO SUMIDERO ',I3,//)
      IF (IFLOW.NE.1) TYPE*,
     1  '    TODOS LOS FLUJOS INICIALES SON CERO'
      IF (IFLOW.EQ.1) TYPE*,
     1  '    LOS FLUJOS INICIALES LOS DIO EL USUARIO'
      TYPE*,'
      TYPE*,' '
      TYPE*,'                              PARA CONTINUAR TECLEE RETURN'
      ACCEPT 625,RETU
709   IF (LISTAS.EQ.1) GO TO 709
      IF (ITIPO.EQ.2) GO TO 710
      WRITE (2,250)
250   FORMAT (//,22X,'PROGRAMA MAXFLUJO SOLUCION NO BASICA',/)
      GO TO 711
710   WRITE (2,251)
251   FORMAT (23X,'PROGRAMA MAXFLUJO SOLUCION BASICA',/)
711   WRITE (2,252)
252   FORMAT (28X,'REPRESENTACION DE LA RED',/)
      WRITE (2,610)N
      WRITE (2,611)M
      WRITE (2,253)
253   FORMAT (/,21X,'PARAMETROS TRANSFORMADOS DE LOS NODOS',/,
     1 24X,'NODO NO.          FLUJO EXTERNO')
      DO 618 I=1,N
618   WRITE (2,254)I,B(I)
254   FORMAT (27X,I3,16X,I6)
```

```
      WRITE (2,255)
255   FORMAT (/,21X,' PARAMETROS TRANSFORMADOS DE LOS ARCOS',/)        90
      WRITE (2,648)
      DO 619 I=1,M
619   WRITE (2,628)I,O(I),T(I),C(I),H(I),F(I)
      WRITE (2,182)VR,SN,TN
      IF (IFLOW.NE.1) WRITE (2,271)
271   FORMAT (4X,'TODOS LOS FLUJOS INICIALES SON CERO',/)
      IF (IFLOW.EQ.1) WRITE (2,272)
272   FORMAT (4X,'LOS FLUJOS INICIALES LOS DIO EL USUARIO',/)
709   CALL MAXFLO(SN,TN,VR,V,INF)
      IF (LISTAS.EQ.2) GO TO 768
      TYPE*,BORR
      TYPE*,SUPE
      IF (INF.NE.1) GO TO 85
      TYPE*,'    NO EXISTE RUTA ADICIONAL'
85    CONTINUE
      IF (V.LT.VR) WRITE(5,83)V
83    FORMAT ('    MAXIMO FLUJO POSIBLE = ',I5)
      IF (V.GE.VR) WRITE (5,91)V
91    FORMAT ('    EL FLUJO DESEADO DE ',I5,' FUE PROCESADO')
      WRITE (5,438)
438   FORMAT (/,17X,'NODO NO.',17X,'APUNTADOR HACIA ATRAS')
      DO 192 I=1,N
      IF (I.EQ.18.OR.I.EQ.38.OR.I.EQ.58) CALL BORRA
      IF (I.EQ.18.OR.I.EQ.38.OR.I.EQ.58) WRITE (5,438)
192   WRITE (5,193) I,PB(I)
193   FORMAT (19X,I3,29X,I5)
      TYPE*,  '
      TYPE*,' PARA CONTINUAR TECLEE RETURN'
      ACCEPT 625,RETU
      TYPE*,BORR
      TYPE*,SUPE
      WRITE (5,448)
448   FORMAT (2X,'ARCO NO.   ORIGEN   TERMINAL    FLUJO    CAPACIDAD',
     1  3X,'COSTO P/U   COSTO FLUJO')
      ICOST=0
      DO 146 K=1,M
      KCOST=F(K)*H(K)
      IF (K.EQ.21.OR.K.EQ.41.OR.K.EQ.61) CALL BORRA
      IF (K.EQ.21.OR.K.EQ.41.OR.K.EQ.61) WRITE (5,448)
      WRITE (5,147) K,O(K),T(K),F(K),C(K),H(K),KCOST
147   FORMAT (4X,I3,7X,I3,7X,I3,2(1X,I10),2(3X,I10))
146   ICOST=ICOST+KCOST
      WRITE (5,148) ICOST
148   FORMAT (/,49X,'   COSTO TOTAL= ',I10)
768   IF (LISTAS.EQ.1) GO TO 769
      IF (INF.EQ.1) WRITE (2,465)
465   FORMAT (4X,'NO EXISTE RUTA ADICIONAL',/)
      IF (V.LT.VR) WRITE (2,83)V
      IF (V.GE.VR) WRITE (2,91,V
      WRITE (2,438)
      DO 492 I=1,N
492   WRITE (2,193)I,PB(I)
      WRITE (2,448)
      ICOST=0
      DO 846 K=1,M
      KCOST=F(K)*H(K)
      WRITE (2,147) K,O(K),T(K),F(K),C(K),H(K),KCOST
846   ICOST=ICOST+KCOST
      WRITE (2,148)ICOST
      TYPE*,' '
      TYPE*,' FAVOR DE SOLICITAR SU LISTADO DE RESULTADOS
```

```
      TYPE*,'  "SALIDA.LIS" AL OPERADOR DE LA IMPRESORA.'
      CALL LIB$DO_COMMAND ('PRINT/DELETE SALIDA.LIS')                    91
769   CALL EXIT
      END
      SUBROUTINE BORRA
      CHARACTER BORR*6,SUPE*6
      SUPE=CHAR(27)//'[0;0H'
      BORR=CHAR(27)//'[2J'
      TYPE*,' '
      TYPE*,'                              PARA CONTINUAR TECLEE RETURN'
      ACCEPT 625,RETU
625   FORMAT (A1)
      TYPE*,BORR
      TYPE*,SUPE
      RETURN
      END
      SUBROUTINE READ
      COMMON/M/B(0:120)/G/M,N/B/T(0:120)/H/PT(0:120)/A/O(0:120)
      COMMON/I/LT(0:120)/J/PO(0:120)/K/C(0:120)/L/H(0:120)
      COMMON ITIPO,LISTAS
      INTEGER SALACK,T,B,O,C,H,PO,PT
      REAL LOWER
      CHARACTER BORR*6,POSE*6,SUPE*6
      SUPE=CHAR(27)//'[0;0H'
      BORR=CHAR(27)//'[2J'
      POSE=CHAR(27)//'[4;0H'
C
C     ************************************************************
C     *                                                          *
C     * PROPOSITO: LEER Y GUARDAR LOS DATOS CORRESPONDIENTES     *
C     * A NODOS Y ARCOS, PARA EL PROBLEMA DE REDES DE FLUJO.     *
C     * LAS SUBRUTINAS REQUERIDAS SON : ORIGS Y TERMS.           *
C     *                                                          *
C     ************************************************************
C
      TYPE*,' '
      TYPE*,'INDIQUE EL NUMERO DE NODOS DE LA RED ORIGINAL:'
      TYPE*,'(DEBE SER UN NUMERO ENTERO,MENOR O IGUAL A 120)'
30    READ (5,88,ERR=17)DA1
88    FORMAT (F3.0)
      N=DA1
      IF (N.LT.1.OR.N.GT.120) GO TO 17
      GO TO 18
17    TYPE*,'DEBE SER ENTERO Y MENOR O IGUAL A 120,FAVOR DE REPETIRLO'
      GO TO 30
18    M=0
      SLACK=N+1
      N=N+1
      DO 51 I=1,N
51    B(I)=0
C
C     NODO
C
      TYPE*,' '
      TYPE*,'EXISTEN DATOS DE LOS NODOS ?'
      TYPE*,'     1 -SI     2 -NO'
115   READ (5,8,ERR=116)IOPC
8     FORMAT (I1)
      IF (IOPC.LT.1.OR.IOPC.GT.2) GO TO 116
      GO TO 117
116   TYPE*,'LA CLAVE DEBE SER 1 O 2,FAVOR DE REPETIRLA'
      GO TO 115
117   IF (IOPC.EQ.2) GO TO 123
```

```
      TYPE*,BORR
      TYPE*,SUPE
      TYPE*,'INDIQUE EL NUMERO DE NODOS CON DATOS'
      TYPE*,'  (DEBE SER UN NUMERO ENTERO)'
122   READ (5,88,ERR=120)DA1
      NUME=DA1
      IF (NUME.LT.1.OR.NUME.GE.N) GO TO 120
      GO TO 121
120   TYPE*,'DEBE SER MAYOR DE 1 Y MENOR O IGUAL'
      TYPE*,'AL NUMERO DE NODOS DE LA RED'
      GO TO 122
121   DO 123 LM=1,NUME
130   TYPE*,' '
      TYPE*,BORR
      TYPE*,POSE
      TYPE*,'INDIQUE EL NUMERO DE NODO:'
      READ (5,88,ERR=130)DA1
      I=DA1
      IF (I.GE.N) GO TO 130
      TYPE*,' '
131   WRITE (5,500)I
500   FORMAT (' INDIQUE EL FLUJO EXTERNO PARA EL NODO ',I3)
      READ (5,98,ERR=131)BF
98    FORMAT (F10.0)
      TYPE*,' '
132   WRITE (5,501)I
501   FORMAT(' INDIQUE LA COTA SUPERIOR DEL FLUJO DE HOLGURA EN EL NODO
     1',I3)
      READ (5,98,ERR=132)BS
      TYPE*,' '
133   WRITE (5,502)I
502   FORMAT(' INDIQUE EL COSTO UNITARIO DEL FLUJO DE HOLGURA EN EL NODO
     1 ',I3)
      READ (5,98,ERR=133)CS
      B(I)=BF
      IF (BS.GT.0) GO TO 71
      J=SLACK
      LOWER=0
      UPPER=-BS
      COST=CS
      GO TO 81
71    J=1
      I=SLACK
      LOWER=0
      UPPER=BS
      COST=CS
81    CONTINUE
      CALL ORIGS(I,J,LOWER,UPPER,COST)
123   CONTINUE
C
C     ARCO
C
      TYPE*,BORR
      TYPE*,SUPE
      TYPE*,' '
      TYPE*,'INDIQUE EL NUMERO DE ARCOS DE LA RED ORIGINAL:'
      TYPE*,'(DEBE SER UN NUMERO ENTERO,MENOR O IGUAL A 120)'
600   READ (5,88,ERR=601)DA1
      KA2=DA1
      IF (KA2.LT.1.OR.KA2.GT.120) GO TO 601
      TYPE*,' '
      GO TO 602
601   TYPE*,'DEBE SER ENTERO Y MENOR O IGUAL A 120,FAVOR DE REPETIRLO'
```

```
      GO TO 600
602   DO 620 KH=1,KA2
888   TYPE*,BORR
      TYPE*,POSE
      WRITE (5,280)KH
280   FORMAT(' ENTRADA DE INFORMACION REFERENTE AL ARCO NUMERO ',I3)
      TYPE*,' '
      WRITE (5,281)KH
281   FORMAT(' INDIQUE EL NUMERO DEL NODO ORIGEN DEL ARCO ',I3)
      READ (5,88,ERR=888)DA1
      I=DA1
      IF (I.LT.1.OR.I.GE.N) GO TO 888
      TYPE*,' '
      WRITE (5,282)KH
282   FORMAT(' INDIQUE EL NUMERO DEL NODO TERMINAL DEL ARCO ',I3)
      READ (5,88,ERR=888)DA1
      J=DA1
      IF (J.LT.1.OR.J.GE.N) GO TO 888
      TYPE*,' '
      WRITE (5,283)KH
283   FORMAT (' INDIQUE LA COTA INFERIOR DEL ARCO ',I3)
      READ (5,98,ERR=888)LOWER
      TYPE*,' '
      WRITE (5,284)KH
284   FORMAT (' INDIQUE LA COTA SUPERIOR DEL ARCO ',I3)
      READ (5,98,ERR=888)UPPER
      TYPE*,' '
      WRITE (5,285)KH
285   FORMAT(' INDIQUE EL COSTO UNITARIO DE ARCO ',I3)
      READ (5,98,ERR=888)COST
      CALL ORIGS(I,J,LOWER,UPPER,COST)
620   CONTINUE
      RETURN
      END
      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)
     1/L/H(0:120)/M/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 CORRESPON-*
C     *DIENTES A UN ARCO, EN UNA LISTA ORDENADA INCREMENTAN-*
C     *DO EL INDICE DEL NODO ORIGEN.                            *
C     *                                                          *
C     ************************************************************
C
      NPLUS1=N+1
C
C  INICIA
C
      IF (M.NE.0) GO TO 10
1     DO 5 II=1,NPLUS1
5     PO(II)=1
C
C     MUEVE
C
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                                    94
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)
C
C       ARCO
C
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
C
C       ************************************************************
C       *                                                          *
C       * PROPOSITO: ELABORA LA LISTA DE APUNTADORES(LT) DEL       *
C       *INDICE DE LOS ARCOS INCREMENTANDO EL NODO TERMINAL.       *
C       *TAMBIEN OBTIENE LOS APUNTADORES(PT) A LOS NODOS TER-      *
C       *MINALES DE LOS ARCOS EN LA LISTA(LT),TAL QUE(LT) PUE-     *
C       *DE SER RAPIDAMENTE REFERENCIADA. ESTE ALGORITMO ES        *
C       *LLAMADO UNA VEZ POR CADA ARCO EN LA RED.                  *
C       *                                                          *
C       ************************************************************
C
        IF (M.NE.1) GO TO 25
        IJ=N+1
        DO 20 I=1,IJ
        PT(I)=1
20      CONTINUE
25      CONTINUE
C
C       MUEVE
C
        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
        KK=M-L
        LT(KK+1)=LT(KK)
40      CONTINUE
50      CONTINUE
        PT(N+1)=PT(N+1)+1
        KK=PT(J+1)-1
```

```
      LT(KK)=K
      RETURN
      END
      SUBROUTINE ORIG (I,LISA,LISN,L)
      COMMON /J/PO(0:120)/B/T(0:120)
      DIMENSION LISA(0:120),LISN(0:120)
      INTEGER PO,T
C
C     ************************************************************
C     *                                                          *
C     * PROPOSITO:DETERMINAR LA LISTA DE ARCOS LISA ORIGINA-*
C     *DOS EN EL NODO I, ENCUENTRA EL NODO TERMINAL DE CADA *
C     *ARCO Y LO PONE EN LA LISTA DE NODOS LISN.            *
C     *                                                          *
C     ************************************************************
C
      ISTA=PO(I)
      ISTO=PO(I+1)-1
      L=0
      IF (ISTO.LT.ISTA) GO TO 40
10    DO 35 K=ISTA,ISTO
      L=L+1
      LISA(L)=K
35    LISN(L)=T(K)
40    CONTINUE
      RETURN
      END
      SUBROUTINE TERM(I,LISA,LISN,L)
      COMMON /A/O(0:120)/H/PT(0:120)/I/LT(0:120)
      DIMENSION LISA(0:120),LISN(0:120)
      INTEGER O,T,PB,PF,PR,PT,LT,PO,C,H,B,CL,F
C
C     ************************************************************
C     *                                                          *
C     * PROPOSITO:DETERMINAR LA LISTA DE ARCOS LISA QUE TER-*
C     *MINAN EN EL NODO (I) A TRAVES DE LA LISTA AUXILIAR (-*
C     *LT) DE TERMS. ADEMAS SE ENCUENTRA EL NODO ORIGINAL DE*
C     *CADA ARCO Y LO PONE EN LA LISTA DE NODOS LISN.       *
C     *                                                          *
C     ************************************************************
C
      ISTA=PT(I)
      ISTO=PT(I+1)-1
      L=0
      IF (ISTO-ISTA)1,2,2
2     DO 3 KK=ISTA,ISTO
      K=LT(KK)
      L=L+1
      LISA(L)=K
3     LISN(L)=O(K)
1     CONTINUE
      RETURN
      END
      SUBROUTINE ROOT(IROOT,LISA,LISN,IC,CYC)
      COMMON /A/O(0:120)/B/T(0:120)/C/PB(0:120)/D/PF(0:120)/F/PR(0:120)
      DIMENSION LISA(0:120),LISN(0:120)
      INTEGER O,T,CYC,PB,PF,PR
C
C     ************************************************************
C     *                                                          *
C     * PROPOSITO: ENCONTRAR LA LISTA DE ARCOS(LISA) Y LA   *
C     *LISTA DE NODOS (LISN) QUE ESTAN EN EL ARBOL DIRIGIDO *
C     *ENRAIZADO EN EL NODO (IROOT), SI LOS APUNTADORES DE- *
```

```
C       *TECTAN UN CICLO SE HACE CYC=1                                  *
C       *                                                               *      96
C       *****************************************************************
C
C       INICIO
C
        II=IROOT
        IC=0
        LISN(1)=IROOT
        CYC=0
C
C       ADELANTE
C
90      JJ=PF(II)
        IF (JJ.EQ.0) GO TO 10
C
C       LISTA ADICIONAL
C
20      IC=IC+1
        LISA(IC)=PB(JJ)
        LISN(IC+1)=JJ
        II=JJ
        IF (II.EQ.IROOT) GO TO 80
        GO TO 90
10      IF (II.EQ.IROOT) GO TO 40
C
C       DERECHO
C
170     JJ=PR(II)
        IF (JJ.EQ.0) GO TO 110
        GO TO 20
C
C       ATRAS
C
110     K=PB(II)
        IF (K.GT.0) GO TO 150
        II=T(-K)
        GO TO 160
150     II=D(K)
160     CONTINUE
        IF (II.EQ.IROOT) GO TO 40
        GO TO 170
80      CYC=1
40      CONTINUE
        RETURN
        END
        SUBROUTINE DELTRE (KL)
        COMMON /A/D(0:120)/B/T(0:120)/C/PB(0:120)/D/PF(0:120)/F/PR(0:120)
     1/G/M,N
        INTEGER D,T,PB,PF,PR
C
C       ***************************************************
C       *                                                 *
C       * PROPOSITO: QUITAR UN ARCO DEL ARBOL Y ACTUALIZAR SU *
C       *REFRESENTACION EN TERMINOS DE LOS APUNTADORES DE LAS *
C       *TRES ETIQUETAS.                                  *
C       *                                                 *
C       ***************************************************
C
        IF (KL.LE.0) GO TO 20
C
C       ADELANTE
C
```

```
10      IL=O(KL)
        JL=T(KL)
        GO TO 30
20      IL=T(-KL)
        JL=O(-KL)
30      CONTINUE
        IF (PF(IL).NE.JL) GO TO 40
        PF(IL)=PR(JL)
        GO TO 80
40      L=PF(IL)
C
C       DERECHO
C
50      CONTINUE
        IF (PR(L).NE.JL) GO TO 90
        PR(L)=PR(JL)
C
C       SALIDA
C
80      PB(JL)=0
        PR(JL)=0
        GO TO 60
90      L=PR(L)
        GO TO 50
60      RETURN
        END
        SUBROUTINE ADDTRE(KE)
        COMMON /A/O(0:120)/B/T(0:120)/C/PB(0:120)/D/PF(0:120)/F/PR(0:120)
        COMMON /G/PD(0:120)/G/M,N
        DIMENSION LISA(0:120),LISN(0:120)
        INTEGER O,T,PB,PF,PR,PD,PDADJ
C
C       ************************************************************
C       *                                                          *
C       * PROPOSITO: INSERTAR EL ARCO  KE(IE,JE) A UN BOSQUE.     *
C       * LOS NODOS  (IE) Y (JE)  DEBERAN ESTAR EN DIFERENTES      *
C       * ARBOLES Y EL NODO  (JE)  DEBERA SER RAIZ DE UN ARBOL     *
C       * LA SUBRUTINA REQUERIDA ES EL ALGORITMO : ROOT.           *
C       *                                                          *
C       ************************************************************
C
C       ADELANTE
C
        IF (KE) 15,15,20
15      IE=T(-KE)
        JE=O(-KE)
        GO TO 30
20      IE=O(KE)
        JE=T(KE)
30      CONTINUE
        IF (PF(IE).EQ.0) GO TO 40
        PR(JE)=PF(IE)
C
C       ATRAS
C
40      PF(IE)=JE
        PB(JE)=KE
C
C       PROFUNDIDAD
C
        PDADJ=PD(IE)-PD(JE)+1
        CALL ROOT (JE,LISA,LISN,IC,CYC)
        DO 50 I=1,IC+1
```

```
      PD(LISN(I))=PD(LISN(I))+PDADJ
50    CONTINUE                                                        98
      RETURN
      END

      SUBROUTINE TRECHG(KL,KE)
      COMMON/A/O(0:120)/B/T(0:120)/C/PB(0:120)/Q/PD(0:120)
      COMMON/G/M,N
      DIMENSION LISA(0:120),LISN(0:120),LISND(0:120)
      INTEGER O,T,PD,PB
C
C     ************************************************************
C     *                                                          *
C     * PROPOSITO: QUITAR UN ARCO  (KL)  DEL ARBOL BASE E        *
C     * INSERTAR OTRO ARCO (KE), ADEMAS DIRIGIR CIERTOS AR-      *
C     * COS EN EL ARBOL PARA MANTENERLO DIRIGIDO.                *
C     * LAS SUBRUTINAS REQUERIDAS SON LOS ALGORITMOS:            *
C     * DELTRE Y ADDTRE.                                         *
C     *                                                          *
C     ************************************************************
C
      CALL DELTRE(KL)
C
C     ENCUENTRA
C
      IF (KL.GT.0) GO TO 10
      JL=O(-KL)
      GO TO 20
10    JL=T(KL)
20    IF (KE.GT.0) GO TO 30
      JE=O(-KE)
      GO TO 40
30    JE=T(KE)
C
C     OBTENCION DE LA RUTA INVERSA
C
40    IC=1
      LISA(1)=-KE
      LISN(1)=JE
      IF (JE.EQ.JL) GO TO 55
60    IC=IC+1
      LISA(IC)=PB(LISN(IC-1))
      IF (LISA(IC).GT.0) GO TO 70
      LISN(IC)=T(-LISA(IC))
      GO TO 80
70    LISN(IC)=O(LISA(IC))
80    IF (LISN(IC).NE.JL) GO TO 60
      DO 1000 I=2,IC
      CALL DELTRE(LISA(I))
      CALL ADDTRE(-LISA(I-1))
1000  CONTINUE
55    CALL ADDTRE (-LISA(IC))
      RETURN
      END
      SUBROUTINE TREINT(N)
      COMMON/A/O(0:120)/B/T(0:120)/C/PB(0:120)/D/PF(0:120)/F/PR(0:120)
      COMMON/Q/PD(0:120)/R/PP(0:120)
      DIMENSION LISA(0:120),LISN(0:120)
      INTEGER PB,PD,O,T,PP
      INTEGER PF,PR
C
C     ************************************************************
```

```
C       *                                                       *
C       * PROPOSITO: ESPECIFICAR LOS APUNTADORES DE UN ARBOL    *          99
C       * TENIENDO CONOCIMIENTO DEL APUNTADOR HACIA ATRAS.      *
C       * LA SUBRUTINA REQUERIDA ES:  ADDTRE.                   *
C       *                                                       *
C       *********************************************************
C
C       OBTENCION DE PP (O PF Y PR) Y TAMBIEN IRROT
C
        DO 80 I=1,N
        PP(I)=I
        PF(I)=0
        PR(I)=0
        PB(I)=0
80      CONTINUE
        DO 10 I=1,N
        IF (PB(I).EQ.0) GO TO 10
        CALL ADDTRE(PB(I))
10      CONTINUE
        RETURN
        END
        SUBROUTINE FLOCHG(LISA,IC,MF)
        COMMON /O/F(0:120)/G/M,N
        DIMENSION LISA(0:120)
        INTEGER F
C
C       *********************************************************
C       *                                                       *
C       * PROPOSITO: CAMBIAR LOS FLUJOS EN UNA TRAYECTORIA      *
C       * (LISA), PARA CADA ARCO EN LA TRAYECTORIA, SI K>0      *
C       * INCREMENTA EL FLUJO EN EL ARCO K POR MF, SI K<0       *
C       * DISMINUYE EL FLUJO EN EL ARCO -K EN MF.               *
C       *                                                       *
C       *********************************************************
C
        DO 100 L=1,IC
        K=LISA(L)
        IF (K.GT.0) GO TO 98
        KW=-K
        F(-K)=F(-K)-MF
        GO TO 99
98      CONTINUE
        F(K)=F(K)+MF
99      CONTINUE
100     CONTINUE
        RETURN
        END
        SUBROUTINE MFLO(LISA,IC,MF,KL,ILC)
        COMMON /O/F(0:120)/K/C(0:120)
        DIMENSION LISA(0:120)
        INTEGER F,C
C
C       *********************************************************
C       *                                                       *
C       * PROPOSITO: DETERMINAR EL MAXIMO CAMBIO DE FLUJO (MF)*
C       * EN LA TRAYECTORIA (LISA).   EL ALGORITMO TAMBIEN DE-*
C       * TERMINA EL ARCO (KL) PARA EL CUAL SE SATURA SU CAPA-*
C       * CIDAD, GUARDANDO ESTA INFORMACION EN (ILC).           *
C       *                                                       *
C       *********************************************************
C
        MF=9999
        KL=0
```

```
      ILC=0
      DO 100 L=1,IC                                                    100
      K=LISA(L)
      IF (K.GT.0) GO TO 97
      IF (MF.GT.F(-K)) GO TO 96
      GO TO 99
96    MF=F(-K)
      KL=K
      ILC=L
      GO TO 99
97    CONTINUE
      KCMF=C(K)-F(K)
      IF (MF.GT.(C(K)-F(K))) GO TO 98
      GO TO 99
98    MF=C(K)-F(K)
      KL=K
      ILC=L
99    CONTINUE
100   CONTINUE
      RETURN
      END
      INTEGER FUNCTION AD(K)
      COMMON /K/C(0:120)/N/CL(0:120)/O/F(0:120)
      INTEGER C,CL,F
C
      AD=0
      IF (K) 20,30,10
10    IF (F(K).LT.C(K)) AD=1
      RETURN
20    IF (F(-K).GT.CL(-K)) AD=1
      RETURN
30    STOP
      END
      SUBROUTINE MAXFLO(SN,TN,VR,V,INF)
      COMMON/A/O(0:120)/B/T(0:120)/C/PB(0:120)/F/PR(0:120)/G/M,N/
     1H/PT(0:120)/I/LT(0:120)/J/PO(0:120)/K/C(0:120)/L/H(0:120)
     2/M/B(0:120)/N/CL(0:120)/O/F(0:120)/D/PF(0:120)/P/PI(0:120)
      DIMENSION LISA(0:120),LISN(0:120)
      INTEGER O,T,PB,PF,PR,PT,PO,C,H,B,CL,F,PI,SN,TN,DEL,V,VR,DWES
C
C     ************************************************************
C     *                                                          *
C     * PROPOSITO: ENCONTRAR EL FLUJO QUE PUEDE PASAR EN LA      *
C     * RED, EL CUAL SE OBTIENE DANDO INICIALMENTE UN FLUJO      *
C     * (VR) DESDE EL NODO FUENTE (SN) HASTA EL NODO SUMIDE-     *
C     * RO (TN).  SI EL FLUJO DADO NO PUEDE SER OBTENIDO EL      *
C     * MAXIMO FLUJO DEL NODO FUENTE AL NODO SUMIDERO ES OB-     *
C     * TENIDO.  SE DEBE DAR UN FLUJO INICIAL FACTIBLE.          *
C     *                                                          *
C     ************************************************************
C
C     INICIA
C
      V=0
      IFIN=0
      INF=0
      IF=1
C
C     RUTA
C
7     CALL FPATH(IF,SN,TN,NF)
      IF (NF.EQ.0) GO TO 1
      INF=1
```

```
      GO TO 3
1     IC=0                                                          101
      IJ=TN
C
C     LISTA DE ARCO
C
9     K=PB(IJ)
      IC=IC+1
      LISA(IC)=K
      IF (K.LT.0) GO TO 6
      IJ=O(K)
10    IF (IJ.EQ.SN) GO TO 8
      GO TO 9
6     IJ=T(-K)
      GO TO 10
8     CONTINUE
C
C     FLUJO
C
      CALL MFLO(LISA,IC,DEL,KL,ILC)
      DWES=DEL+V-VR
      IF (DEL+V.LT.VR) GO TO 4
      DEL=VR-V
      IFIN=1
4     CALL FLOCHG(LISA,IC,DEL)
      V=V+DEL
      IF (IFIN.EQ.0) GO TO 7
3     CONTINUE
      RETURN
      END
      SUBROUTINE LABEL1(SN,TN,NP)
      DIMENSION LISA(0:120),LISN(0:120),S(0:120)
      COMMON /C/PB(0:120)/G/M,N
      INTEGER SN,TN,S,PB,AD
C
C     ************************************************************
C     *                                                          *
C     * PROPOSITO: ENCONTRAR UNA RUTA DESDE EL NODO (SN) AL      *
C     * NODO (TN), USANDO SOLAMENTE ARCOS ADMISIBLES HACIA       *
C     * ADELANTA O HACIA ATRAS.                                  *
C     *                                                          *
C     ************************************************************
C
C     INICIA
C
      NP=0
      DO 10 I=1,N
      S(I)=0
10    PB(I)=0
      S(SN)=1
      I=SN
      IT=1
      IC=1
C
C     ADELANTE
C
20    CALL ORIG (I,LISA,LISN,L)
      IF (L.LE.0) GO TO 40
      DO 30 LK=1,L
      K=LISA(LK)
      J=LISN(LK)
      IF (AD(K).NE.1.OR.S(J).NE.0) GO TO 30
      IT=IT+1
```

```
      S(J)=IT
      PB(J)=K
      IF (TN.LE.0) GO TO 30
      IF (S(TN).GT.0) GO TO 80
30    CONTINUE
C
C     REFLEJADO
C
40    CALL TERM(I,LISA,LISN,L)
      IF (L.LE.0) GO TO 60
      DO 50 LK=1,L
      K=LISA(LK)
      J=LISN(LK)
      IF (AD(-K).NE.1.OR.S(J).NE.0) GO TO 50
      IT=IT+1
      S(J)=IT
      PB(J)=-K
      IF (TN.LE.0) GO TO 50
      IF (S(TN).GT.0) GO TO 80
50    CONTINUE
C
C     SELECCIONA
C
60    IC=IC+1
      IF (IC.GT.N) GO TO 80
      DO 70 I=1,N
      IF (S(I).EQ.IC) GO TO 20
70    CONTINUE
      NP=1
80    CONTINUE
      RETURN
      END
      SUBROUTINE FPATH(IF,SN,TN,NP)
      COMMON /C/PB(0:120)/G/M,N
      COMMON ITIPO
      INTEGER PB,SN,TN
C
C     ************************************************************
C     *                                                          *
C     * PROPOSITO: PROCEDE A ENCONTRAR LA RUTA PARA EL ALGO-*
C     * RITMO DE RUTA MAS CORTA CUANDO SE REQUIERE UNA SOLU-*
C     * CION BASICA.                                             *
C     *                                                          *
C     ************************************************************
C
      IF (ITIPO.EQ.1) GO TO 88
      IF (IF.NE.1) GO TO 10
      CALL LABEL1(SN,0,NP)
      IF (NP.EQ.1) GO TO 30
      CALL TREINT(N)
30    IF=0
      RETURN
10    CALL LABEL2(SN,INF)
      IF (INF.EQ.1) NP=1
      RETURN
88    CALL LABEL1(SN,TN,NP)
      RETURN
      END
      SUBROUTINE LABEL2(SN,INF)
      COMMON /A/D(0:120)/B/T(0:120),G/M,N
      COMMON /C/PB(0:120)
      DIMENSION LISN(0:120),LISA(0:120),S(200)
      INTEGER AD,D,S,SN,T,CYC
```

```
C
C       ********************************************************      103
C       *                                                      *
C       * PROPOSITO:   PROPORCIONAR UN NUEVO ARBOL GENERADOR   *
C       * DESPUES DE QUE UNO O MAS ARCOS EN EL ARBOL ORIGINAL  *
C       * SE HACEN INADMISIBLES.                               *
C       *                                                      *
C       ********************************************************
C
C       INICIA
C
        INF=0
        JJ=SN
15      CONTINUE
C
C       SALE
C
        DO 10 I=1,N
        S(I)=0
10      CONTINUE
        CALL ROOT(JJ,LISA,LISN,IC,CYC)
        IF (IC.LT.1) GO TO 1
        DO 20 L=1,IC
        K=LISA(L)
        IF (AD(K).NE.0) GO TO 20
        KL=K
        L1=L+1
        J=LISN(L1)
        GO TO 40
20      CONTINUE
        GO TO 1
C
C       ARBOL
C
40      CALL ROOT(J,LISA,LISN,IC,CYC)
        DO 42 L=1,IC+1
42      S(LISN(L))=1
C
C       ENTRA
C
50      KE=0
        DO 54 K=1,M
        I=O(K)
        J=T(K)
        IF (AD(K).EQ.1.AND.S(I).EQ.0.AND.S(J).EQ.1) KE=K
        IF (AD(-K).EQ.1.AND.S(I).EQ.1.AND.S(J).EQ.0) KE=-K
        IF (KE.NE.0) GO TO 60
54      CONTINUE
        INF=1
1       CONTINUE
        RETURN
60      CONTINUE
C
C       PIVOTE
C
        CALL TRECHG(KL,KE)
        IF (KE) 61,95,62
61      JJ=O(-KE)
        GO TO 15
62      JJ=T(KE)
        GO TO 15
95      CONTINUE
        STOP
```

```
END
```

B I B L I O G R A F I A

- Jensen P.A., Barnes J.W. Network Flow Programming.
  John Wiley, 1980.

- Bazaraa Mokhtar, Jarvis John, Programación Lineal y Flujo en Redes.
  Editorial Limusa, México 1981.

- Gerez Victor, Grijalva Manuel. El Enfoque de Sistemas.
  Editorial Limusa. México, 1978.

- Barsov A.S. Qué es Programación Lineal
  Editorial Limusa, México 1976.

- Kenninton J.L., Helgason R.V. Algorithms for Network Programming.
  John Wiley, 1980.

- Dantzig G.B., D.R. Fulkerson. On the MIN-CUT MAX-FLOW Theorem of
  Networks. Princenton University, 1956

- Fora L.R., D.R. Fulkerson. A Simple Algorithm for Finding Maximal
  Network Flows. Canadian Journal of Mathematics, 1957.

- Johnson E.L. Programming in Networks and Graphs.
  University of California, 1965.