$500.-
BC

27 ENE.

V.B°
Pablo Verg
28/XI/85

UNIVERSIDAD DE MONTERREY

DIVISION DE CIENCIAS NATURALES
Y EXACTAS

Título

ELABORACION DE UN SISTEMA COMPUTACIONAL PARA LA SOLUCION DE PROBLEMAS DE OPTIMIZACION DE FLUJO EN REDES

REPORTE DEL PROGRAMA DE EVALUACION FINAL
PRESENTADO POR

Autor
CARLOS JORGE SADA SERNA

EN OPCION AL TITULO DE
INGENIERO INDUSTRIAL Y DE SISTEMAS

MONTERREY, N. L. DICIEMBRE DE 1985

I N D I C E

LA INTRODUCCION

Optimización, es un concepto que atañe a infinidad de circunstancias en la vida diaria, a las que se le aplica éste, inconscientemente, o bien usando el sentido común, pero en cuanto a ingeniería o cualquier otra área productiva se refiere, se enfoca a problemas a los que si bien en muchas ocasiones se les puede atacar con el sentido común o con la experiencia, existen casos en que el grado de dificultad aumenta conforme aumenta su importancia y complejidad, por lo que no es posible tratarlos empíricamente, ejemplo de esto podría ser algún proceso mecánico iterativo, o bien de producción, o de economía, así como una representación gráfica del clásico problema de transporte, de redes eléctricas, o bien modelos abstractos que representan la toma de decisiones, flujo de información en un sistema, etc.

El estudio de las redes como una rama independiente dentro de la optimización matemática determinística, puede aportar una ayuda muy eficaz en el tratamiento de algunos problemas que aparecen en muy diversas áreas, tanto económicas como tecnológicas o sociológicas; prometiendo dar buenos resultados tanto al matemático puro y teórico como al ingeniero práctico, el biólogo, el sociólogo, psicólogo, etc. Esta forma de estudiar las redes, es algo relativamente nuevo, ya que anteriormente se trataban éstos tipos de problemas, mediante modelos de programación lineal, lo que los hacía un tanto complicados y laboriosos de resolver, así como limitados en su complejidad.

Para el estudio de las redes, existen muy diversas maneras de resolverlas, unas son simplemente por medio de la visualización gráfica, lo cual las limita en cuanto a grado de dificultad se refiere, otras son muy laboriosas con muchas operaciones, alternaciones a la red original gene-

rando arcos y nodos nuevos, cambios en la capacidad de - flujo de los arcos, repitiéndose sucesivamente hasta llegar al óptimo, lo cual los hace un tanto imprácticos, ya que al igual que losmétodos del tipo anterior, conforme - crece el tamaño de la red en estudio, crece la complejidad del problema, así mismo, si lo que se pretende es implantar ese método en un programa computacional, aparte de la dificultad del algoritmo para programarse, la memoria de la computadora tiende a saturarse rápidamente debido al incremento en el número de arcos y nodos con respecto a la red original, limitando el tamaño de las redes que se pueden resolver.

Lo que se necesita para éste fin, es un algoritmo no tan complejo, pero que sea eficiente, que no altere el número de nodos y arcos de la red original, y que sea lo suficientemente iterativo como para que el proceso sea más generalizado y se resuelvan diversos tipos de problemas

Aquí se estudiarán tres casos distintos de problemas, cuyos algoritmos para resolverlos son similares aunque no en su totalidad, de manera que al hacer un solo programa computacional para resolverlos, gran parte de éste sea - utilizado por más de un método o tipo de problemas.

Dichos métodos, encuentran un camino, con el flujo a través de éste si fuese necesario, mediante un proceso iterativo denominado "de cortes" el cual consiste en separar en dos el conjunto de nodos de la red, elegir un arco que conecte uno de los conjuntos con el otro, y su correspondiente nodo, volver a realizar un corte, y así sucesivamente hasta llegar a una conclusión, ya sea la solución óptima, o bien decidir que no existe una solución factible.

La diferencia entre estos métodos radica en el tipo de -

restricciones o condiciones del problema. El primero y más sencillo, solamente busca un camino entre dos nodos determinados con las restricciones de que la cantidad que sale del nodo I a través del arco J es la misma que llega al nodo I', o sea que el flujo se conserva en cada uno de los arcos, y que la capacidad de ellos es siempre ilimitada, de manera que nunca un arco llegará a saturarse.

En el segundo, existe otra restricción, que al menos uno de los arcos está limitado en su capacidad máxima de flujo y el algoritmo busca un camino tal que el flujo que pase a través de él sea máximo, el procedimiento es el mismo excepto de que el algoritmo se repite, buscando nuevos caminos para que exista un mayor flujo a través de la red cada vez, hasta que ya no exista un camino por el cual - ninguno de sus arcos esté saturado, al llegar a ésto, se ha encontrado el máximo flujo.

El tercer método, aparte de que existen restricciones en la capacidad de los arcos, se requiere que la diferencia entre la cantidad del flujo que sale de un nodo y la cantidad que entra a él, sea igual a una cantidad previamente determinada, así como la restricción que la suma de - éstas diferencias (llamadas divergencia) sea igual a cero y el algoritmo encontrará un camino tal, que el flujo por esos arcos sea factible en cuanto a su capacidad y - satisfaga las restricciones de divergencia.

LA TEORIA

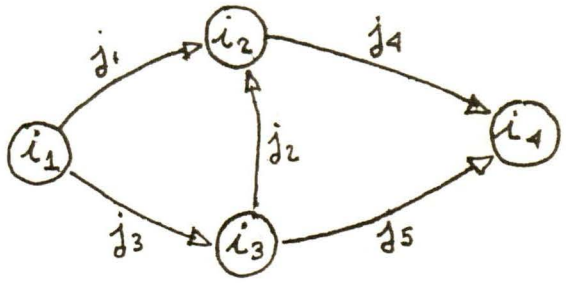
TERMINOLOGIA:

Una red es un conjunto abstracto, que se puede representar gráficamente y consta de 3 elementos: nodos, arcos y una función que le dá dirección a los arcos y conecta los nodos.

Notación: $I \longrightarrow$ Nodo

$J \longrightarrow$ Arco

$\sim \longrightarrow$ Función



$$N = \{I_1, I_2, I_3, I_4\}$$

$$A = \{J_1, J_2, J_3, J_4, J_5\}$$

Función que relaciona los nodos y los arcos:

$$J_3 \sim (I_1, I_3)$$

$$\text{Arco } J \sim (\text{Nodo donde se origina}, \text{Nodo donde se termina})$$

REPRESENTACION NUMERICA DE LAS REDES

Para estudiar las redes, mediante la optimización matemática determinística es de gran utilidad, más no indispensable, la representación gráfica de las redes, es útil para entender y visualizar el proceso, hacer ésto es sencillo, pero efectuar las operaciones con los flujos, determinar el máximo puede resultar algo laborioso, para ello se pueden representar las redes numéricamente y resolverlos utilizando una computadora que haga las operaciones - por uno.

Función de Incidencias:

La función de incidencias "E" está dado por

$$E(I, J) = \begin{cases} +1 & \text{si I es el nodo inicial de J} \\ -1 & \text{si I es el nodo terminal de J} \\ 0 & \text{de cualquier otra manera} \end{cases}$$

MATRIZ DE INCIDENCIAS:

Es una matriz bidimensional de "N" renglones por "A" columnas, donde N es el número de nodos y A el número de arcos y relaciona arcos y nodos mediante la función de incidencias

$$\begin{array}{c} \\ I_1 \\ I_2 \\ I_3 \\ I_4 \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccc} J_1 & J_2 & J_3 & J_4 & J_5 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ -1 & -1 & 0 & 1 & 0 \\ 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 \end{bmatrix} \end{array} = E$$

FUNCION ADYACENTE:

La función adyacente " $\hat{e}$ ", relaciona al nodo emisor con el nodo receptor:

$$\hat{e}(I, I') = \begin{cases} 1 & \text{si existe un arco } J \ni J \sim (I, I') \\ 0 & \text{de otra manera} \end{cases}$$

MATRIZ ADYACENTE:

Es una matriz siempre cuadrada de N renglones por N columnas, cuya diagonal principal es siempre cero, donde N es el número de nodos

$$
\begin{array}{c}
\\
I_1 \\
I_2 \\
I_3 \\
I_4 \\
\\
\end{array}
\begin{array}{c}
\begin{array}{cccc} I_1 & I_2 & I_3 & I_4 \end{array} \\
\left[\begin{array}{cccc}
0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0
\end{array}\right]
\end{array}
= \hat{E}
$$

FLUJO:

Identificado por la variable X_J, es una cantidad no negativa que tiene la dirección del arco J que conecta a los nodos I con I' $\quad J \sim (I, I')$, donde la cantidad X_J que sale del nodo I, es la misma que llega al nodo I', en otras palabras, el flujo se conserva en todos los arcos.

Si consideramos un conjunto de arcos $A = \{J_1, J_2 \ldots J_n\}$ cada uno de los cuales tienen asociado un flujo X, entonces, podemos pensar en el flujo X como un vector $\tilde{X}$

$$\tilde{X} = (XJ_1, XJ_2 \ldots . X_{Jn})$$

DIVERGENCIA:

Cuando se desea saber que pasa con el flujo "$\tilde{X}$" de una red "G" para un nodo en particular, se tiene la divergencia, que es la diferencia que hay entre lo que sale del nodo y lo que llega a él, y como la matriz de incidencias nos dice que arcos llegan y cuáles salen de cada nodo, sólo nos faltaría multiplicar dicho renglón (nodo) por el vector de flujo $\tilde{X}$.

Así, la divergencia Y del nodo I_K es

$$Y(I_K) = \sum_{j \in A} E(I_K, J) \cdot X_J$$

el vector de divergencia $\tilde{Y} = E \cdot \tilde{X}$

De aquí tenemos que:

si $Y(I) > 0$ => El nodo I es un nodo fuente
si $Y(I) < 0$ => El nodo I es un nodo receptor
si $Y(I) = 0$ => Se dice que el flujo se conserva en el nodo I

CIRCULACION:

Circulación se le denomina a cualquier flujo $\tilde{X}$ tal que div $(\tilde{X}) = \tilde{0}$ (lo que significa que el flujo se conserva en todos los nodos de la red). (Esto incluye el flujo $\tilde{X} = \tilde{0}$).

Las circulaciones se consideran importantes debido a que discusiones teóricas pueden ser simplificadas en términos de ellas.

LA RED AUMENTADA:

Se puede identificar cualquier flujo arbitrario $\tilde{X}$ de una red determinada "G" como una circulación de una red aumentada "$\bar{G}$".

La forma de construir una red aumentada "$\bar{G}$" es añadiendo un nodo $\bar{I}$ y conectarlo con un arco para cada nodo de la red original, el flujo en los arcos originales no se altera en nada, pero para los nuevos arcos, su flujo depende de la divergencia del nodo I, de manera que:

si $Y(I) > 0$ => $J_k \sim (\bar{I}, I)$ y $X_{J_K} = Y\,I$

si $Y(I) < 0$ => $J_K \sim (I, \bar{I})$ y $X_{J_K} = |Y\,I|$

si $Y(I) = 0$ => No necesitamos conectar I con $\bar{I}$

CAMINOS EN UNA RED:

Un camino P de una red G es una secuencia finita de la -

forma:

I_0, J_1, I_1, J_2, I_2.... J_r, I_r donde I_K es un nodo y J_K es un arco $\ni$ $J_K \sim (I_{K-1}, I_K)$ donde I_{K-1} es el nodo inicial de J_K e I_K es el nodo terminal.

FUNCION DE INCIDENCIAS PARA CAMINOS:

La función de incidencias para el camino P, nos indica - por cuales nodos de la red son utilizados en el camino, y está dada por:

$$\epsilon_p(J) = \epsilon(J, P) = \begin{cases} 1 \text{ si } J \in P \\ 0 \text{ si } J \notin P \end{cases} \quad \forall_j \in A$$

La Red Pintada: Pintando los arcos de una red se distinguen dos categorías:

Arcos Blancos: Los arcos que pueden utilizarse en la dirección apropiada.

Arcos Rojos: Los arcos que no se pueden utilizar.

De esta manera, si se tiene una red, y se identifican dos nodos diferentes, "I_0" nodo inicial y "I_r" nodo terminal, así como el color apropiado para cada arco, sólo nos restaría el problema de encontrar un camino P.: S $\longrightarrow$ S' tal que todos los arcos de "P" sean blancos.

CORTES DE UNA RED.

Un corte de una red es un conjunto de la forma $[S, N/S]$ donde "S" es un subconjunto de nodos de la red "G" y "N/S" es el complemento de "S" en "N".

El conjunto $[S, N/S]$ se describe como:

$Q^{+} = \left[S, N/S\right]^{+}$ = Conjunto de arcos J, elementos de A tal que $J \sim (I, I')$ para I elemento de "S" e I' elemento de N/S

$Q^{-} = \left[S, N/S\right]^{-}$ = $J \in A/J \sim (I', I)$ para $I \in$ "S", $I' \in N/S$

Un corte de una red, es un conjunto de arcos, de manera que si no existiera, se desconectaría completamente la red. Un corte separa a la fuente de la red, del destino de la misma ("I_0" de "I_r")

FUNCION DE INCIDENCIAS PARA UN CORTE:

Esta función, determina qué arcos salen del conjunto "S" al conjunto "N/S" y viceversa, y está dada por:

$$e_Q(J) = e(Q, J) = \begin{cases} 1 \text{ si } J \in Q^{+} \\ -1 \text{ si } J \in Q^{-} \\ 0 \text{ si } J \notin Q \end{cases}$$

EL CAMINO

ALGORITMO PARA ENCONTRAR UN CAMINO EN UNA RED PINTADA;

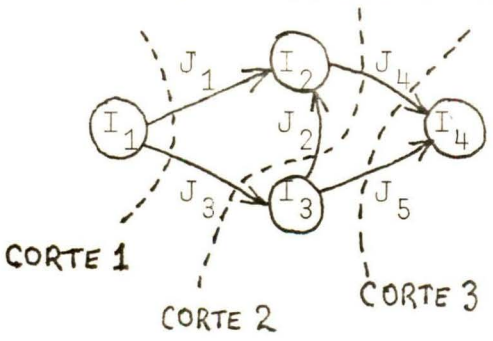
TODOS LOS ARCOS TIENEN CAPACIDAD ILIMITADA.

Definida ya la red y los colores de sus arcos, se identifica inicialmente el conjunto $S = \{I_0\}$.

Inspeccionamos el corte $Q = [S, N/S]$ y se determina - si existe un arco J elemento de Q^+ que sea blanco. De no haberlo, se detiene el procedimiento y se concluye - que no existe una solución factible, por el contrario, - si hubiere un arco J elemento de Q^+ de color blanco, entonces se determina $\Theta(I) = J$, donde I es el nodo terminal de J.

Ahora redefinimos $S = S \cup \{I\}$, pero si $I = a\ I_r$, se detiene el procedimiento y se ha obtenido un camino P: $I_0 \rightarrow I_r$, por medio de la ruta Θ, si no, se repite el procedimiento hasta que se llegue al nodo I_r o se determine la inexistencia del camino.

En el caso de que en un corte existieran más de un arco elemento de Q^+ y blanco, se elige el arco con el subíndice menor o bien se le puede dar prioridad a los arcos provenientes de los nodos que entraron antes al conjunto "S"



$$\begin{array}{c} \\ I_1 \\ I_2 \\ I_3 \\ I_4 \end{array} \begin{array}{c} \begin{array}{ccccc} J_1 & J_2 & J_3 & J_4 & J_5 \end{array} \\ \begin{bmatrix} 1 & & 1 & & \\ -1 & -1 & & 1 & \\ & 1 & -1 & & 1 \\ & & & -1 & -1 \end{bmatrix} \end{array}$$

Corte 1:

$$S = \{I_1\} \quad Q^+ = \{J_1, J_3\} \quad \Theta(I_2) = J_1$$

$$S = S + \{I_2\}$$

Corte 2:

$$S = \{I_1, I_2\} \quad Q^+ = \{J_3, J_4\} \quad \Theta(I_3) \quad J_3$$

$$S = S + \{I_3\}$$

Corte 3 $S = \{I_1, I_2, I_3\}$ $Q^+ = \{J_4, J_5\}$ $\Theta(I_4) = J_4$

$$I_4 = \text{SPRI}$$

$\Rightarrow$ Camino P: $I_1 \longrightarrow I_4$

$=$ P: I_1, J_1, I_2, J_4, I_4

EL FLUJO

ALGORITMO PARA MAXIMIZAR EL FLUJO A TRAVES DE UNA RED.

Cuando no se tienen restricciones de capacidad de flujo en los arcos, o sea, cuando todos ellos tienen capacidad ilimitada, maximizar el flujo es cuestión de aumentarlo y ya, pues no es necesario usar otros arcos si los que se usan tienen la capacidad de aumentar indefinidamente su flujo. Pero si los problemas involucran flujos, en la mayoría de los casos, existen restricciones en por lo menos algunos de los arcos con respecto al mínimo y al máximo que pudiera enviarse de un nodo a otro a través de un arco.

El siguiente algoritmo además de determinar un flujo tal, que llena los requisitos de capacidad, encuentra el que maximice la cantidad a mover a través de la red.

DEFINICIONES:

Intervalo de capacidades: $C^+(J)$ = Capacidad máxima del arco J

para

$C^-(J)$ = Capacidad mínima del arco J

$$C(J) = \left[C^-(J), C^+(J) \right]$$

Entonces decimos que un flujo "X" a través de una red - "G" es factible con respecto a las capacidades de los - arcos si

$$X(J) = X_J \in C(J) \quad \text{para toda J elemento de A}$$

EL FLUJO A TRAVES DE UN CORTE DE RED:

El flujo de X a través de un corte Q, es igual a la suma de los flujos de los arcos que salen de "S" a "N/S" menos la suma de los flujos de los arcos que van de "N/S"

a "S", dicho de otra manera,

$$e_Q \cdot X = \sum_{j \in Q^+} x_j - \sum_{j \in Q^-} x_j$$

donde e = función de incidencias para Q

LA DIVERGENCIA DEL CONJUNTO S:

para un corte Q = S, N/S , la divergencia de X para - el conjunto S está dada por:

$$Y(S) = \sum_{I \in S} Y(I) \qquad \text{donde } I = \text{div } (X)$$

Conocido esto, el problema es maximizar el flujo total de I_0 a I_r conservando el flujo en todos los demás nodos de la red y considerando flujos que sean factibles con respecto a sus intervalos de capacidades. O sea, se toma en cuenta cualquier flujo $\tilde{X}$ tal que $Y(I) = 0 \ \forall I \neq I_0, I_r, \Rightarrow Y(I_0) + Y(I_r) = 0$

TEOREMA DE FORD-FULKERSON (MAXIMO FLUJO - MINIMO CORTE)

Asumir que existe por lo menos un flujo X que satisface las condiciones de factibilidad con respecto a los intervalos de capacidades y de conservar el flujo en todos - los nodos diferentes de I_0 y de I_r . Entonces el máximo flujo = mínimo corte.

Este flujo puede tener un valor $+\infty$ si existe un camino P: $I_0 \rightarrow I_r$ de capacidad ilimitada; si no lo hay, el valor es finito y el problema de maximizar el flujo de - I_0 a I_r tiene solución.

Si se tiene un flujo $\tilde{X}$ que satisface las condiciones y - restricciones del problema de maximizar el flujo de I_0 a I_r y además se tiene un corte Q: $I_0 \leftarrow I_r$ tal que el flujo a través de dicho corte sea igual a la capacidad - máxima del corte, entonces se puede afirmar que $\tilde{X}$

es la solución del problema de maximizar el flujo.

Este algoritmo, "mejora" el flujo X a través de la red en cada iteración, superimponiendo otro flujo a través de un camino P: $I_0 \rightarrow I_r$.

Un camino que va de I_0 a I_r , puede aumentar el flujo que pasa a través de él, si para cada uno de los arcos pertenecientes a este camino tiene un flujo menor a su capacidad máxima. Si este es el caso, entonces existe un número α mayor que cero, de tal manera que el nuevo flujo $X'(J) = X(J) + \alpha \cdot e_p(J)$.

Este nuevo flujo deberá satisfacer las condiciones y restricciones del problema de maximización del flujo si el número α es un número positivo menor o igual a $C^+(J) - X(J)$ para toda J elemento del camino "P".

Se debe asumir que:

* No existe un camino P: $I_0 \rightarrow I_r$ de capacidad ilimitada.
* Exista un flujo X tal que sea factible con respecto a los intervalos de capacidad de los arcos y que la divergencia en todos los nodos diferentes a I_0 y a I_r sea igual a cero.
* Exista por lo menos un camino P: $I_0 \rightarrow I_r$.

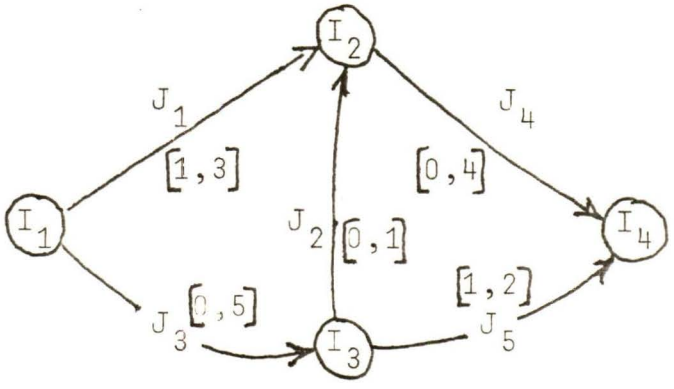
Después de ésto se debe pintar los arcos de:

blanco si $X(J) < C^+(J)$

rojo si $X(J) = C^+(J)$

La solución se obtiene cuando se llega a un corte Q $I_0 \leftarrow I_r$ para el cual $X(J) = C^+(J) \quad \forall \quad (J) \in Q^+$

Si se termina el proceso con un camino que aún tiene posibilidades de aumento de su flujo, entonces se define - el número como el mínimo de $(C^{+}(J) - X(J))$ Para toda J elemento del camino "P", como α será positivo, el flujo $X' = X + \alpha e_p$ el flujo total a través de la red será mayor y éste también satisface las restriccioes de capacidades y que las divergencias sean iguales a cero para todo nodo diferente de I_0 y de I_r.



Arco	=	J_1	J_2	J_3	J_4	J_5
Capacidad máxima	=	3	1	5	4	2
Capacidad mínima o flujo inicial	=	1	0	0	0	1
Color Inicial	=	b	b	b	b	b

Se encuentra un camino mediante el proceso iterativo del método anterior.

P: I_1, J_1, I_2, J_4, I_4

$J \in P$: J_1, J_4

α = mín ((3 -1) , (4 -0)) = 2

e_p = (1 , 0 , 0 , 1 , 0)

$X' = X + \alpha \cdot e_p = (1 , 0 , 0 , 0 , 1) + (1,0,0,1,0) = 2$

X' = (3, 0, 0, 2, 1)

y se pinta de rojo el arco # 1

Se obtiene el camino de la red pintada mediante el algoritmo del primer método, y se obtiene:

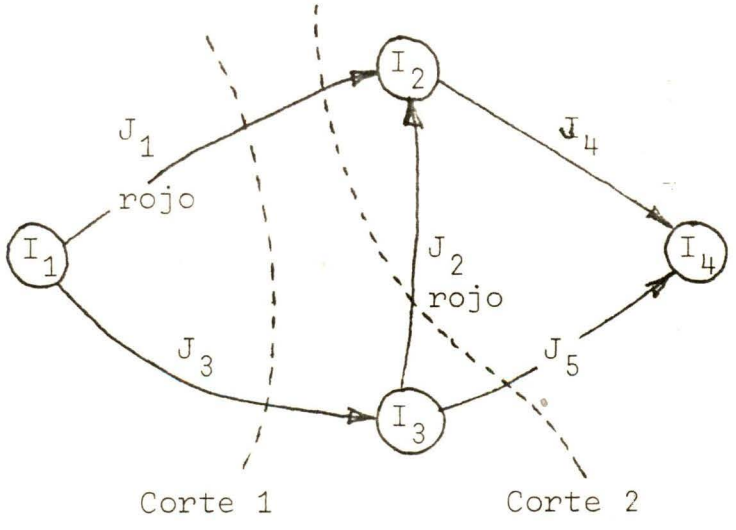
P: I_1 ,J_3 ,I_3 ,J_2 ,I_2 ,J_4 ,I_4

e_p = (0 ,1 ,1 ,1 ,0)

α = mín ((5 -0) , (1 -0) , (4 -2)) =1

Así se obtiene el nuevo flujo X' = (3 ,1 1, 3, 1) para los arvos, que pintados ahora con el nuevo flujo, se tiene:

Color = (r ,r ,b ,b ,b)



Analizando la red, y sus respectivos cortes se encuentra el camino factible P:

$$P = I_1 , J_3 , I_3 , J_5 , I_4 \quad y$$

$$e_p = (0 , 0 , 1 , 0 , 1) \quad y$$

$$X = ((5 -1) , (2 -1)) = 1$$

$\Rightarrow$ X' = (3 1, ,2 ,3 2) con sus respectivos -

colores = (r ,r ,b ,b ,r) ,

por lo tanto, ya no se puede encontrar un camino factible por el que se pueda aumentar el flujo, así, habiendo llegado a este punto del proceso, se tiene ya un flujo - máximo que puede pasar por la red.

$$X (J_1) = 3$$

$$X (J_2) = 1$$

$$X (J_3) = 2$$

$$X (J_4) = 3$$

$$X (J_5) = 2$$

LA DIVERGENCIA

ALGORITMO PARA DETERMINAR UN FLUJO QUE CUMPLA CON LAS - RESTRICCIONES DE DIVERGENCIA.

El resolver un problema que aparte de requerir que los - flujos estén dentro de un intervalo requiere que las divergencias en cada nodo sea un número previamente determinado no es algo inmediato. En el algoritmo anterior, el no usar un arco, no afectaba las restricciones de capacidad, si tenía como límite inferior de capacidad - igual a cero, pero ahora existe una restricción más, el que un arco emplee su capacidad mínima puede no cumplir con las restricciones de divergencia en los nodos, de - manera que es necesario emplear otro algoritmo, uno que determine el flujo que llene los requisitos, o sea que sea factible o bien que indique que el problema no tiene solución.

El problema, es encontrar un flujo X tal que:

$$X(J) \in C(J) \quad \forall J \in A$$

$$Y(I) \in C(I) \quad \forall I \in N$$

donde

$$C(J) = \left[C^-(J),\ C^+\right]$$

$$C(I) = \left[C^-(I),\ C^+(I)\right]$$

$$Y: -\infty < C^-(I) \leq C^+(I) < +\infty$$

$$X(J) \in C(J); \quad Y(I) = b(I) \in C(I)$$

donde b (I) es la restricción de divergencia para el nodo I

TEOREMA DE GALE- HOFFMAN (DE LA DISTRIBUCION FACTIBLE)

El problema de la distribución factible tiene solución si y solo si $B(N) = 0$ $\left(B(S) = \sum_{i \in S} (b(I))\right)$ y

$$B(S) \leq C^+(Q) \quad \forall Q = \left[S,\ N/S\right] \text{ de la red}$$

El algoritmo para resolver este tipo de problemas, es muy similar al anterior, salvo algunas pequeñas diferencias. Se trata de buscar un flujo X que satisfaga las condiciones de capacidad y aparte se requiere que la divergencia $Y(I) = b(I)$ para todo nodo I elemento de la red.

Se comienza con un flujo X que sea factible a los intervalos de capacidad, independientemente de si cumple o no los requisitos de divergencias, claro que si los cumple, el problema ya está resuelto pero generalmente se empieza haciendo $X(J) = C^-(J)$ para todo arco J elemento de la red y el proceso comienza de la siguiente manera:

Se definen dos grupos, N^+ y N^-

$$N^+ = \{ I \in N / (b(I) - Y(I)) > 0 \}$$
$$N^- = \{ I \in N / (b(I) - Y(I)) < 0 \}$$

Si $N^+ = a$ $N^- = \phi$, entonces el flujo disponible es la solución de la distribución factible, si no, ambos conjuntos serán diferentes al conjunto vacío y $N^+ \cap N^- = \phi$ y se les asignan colores a los arcos:

J = blanco si $X(J) < C^+(J)$

J = rojo si $X(J) = C^+(J)$

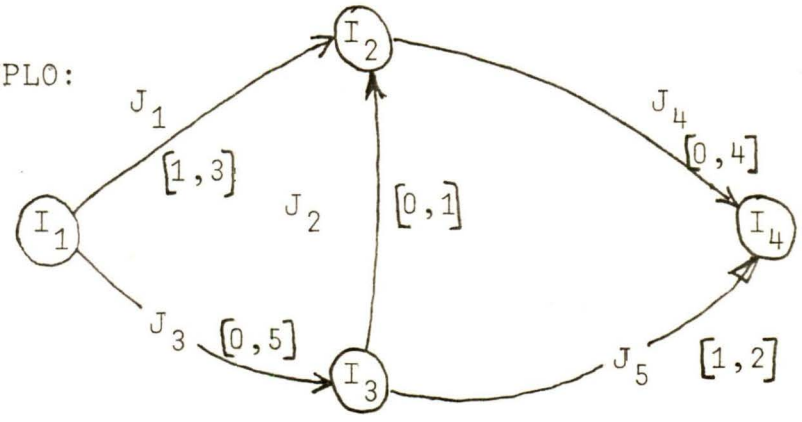
Se selecciona un nodo I_0 cualquiera de N^+ y un nodo I_r cualquiera de N^- y se aplica el algoritmo para encontrar un camino $P: I_0 \rightarrow I_r$ de una red pintada. si la solución fuese un corte Q que separa I_0 de I_r ($Q: I_0 \leftarrow I_r$), entonces es necesario parar la ejecución, ya que $Q \left[S, N/S \right]$, tiene la característica de que $b(S) > C^+(Q)$; por lo tanto, no existe un flujo factible que satisfaga las restricciones de divergencia. Por otro lado, si encontramos un camino $P: I_0 \rightarrow I_r$, entonces

definimos como un número tal que

$$\alpha = \min \begin{cases} C^{+} (J) - X (J) & J \in P \\ b (I) - Y (I) & I = I_0 \\ Y (I) - b (I) & I = I_r \end{cases}$$

y redefinimos $X' = X + \alpha e_p$ y repetimos el procedimiento.

EJEMPLO:



Se pide un flujo factible tal, que, las divergencias en los nodos sean:

$$X \max = (3, 1, 5, 4, 2)$$

$$\begin{aligned} b (I_1) &= 8 \\ b (I_2) &= -1 \\ b (I_3) &= -3 \\ b (I_4) &= -4 \\ \hline b (N) &= 0 \end{aligned}$$

El flujo inicial para la red es X = (1, 0, 0, 0, 1)

y la divergencia "Y" de cada nodo con el flujo actual, es:

$$Y\ (I_1) = 1\ ,\ b - Y = 7 > 0 \Rightarrow I_1 \in N^+$$
$$Y\ (I_2) = -1\ ,\ b - Y = 0$$
$$Y\ (I_3) = 1\ ,\ b - Y = -4 < 0 \Rightarrow I_3 \in N^-$$
$$Y\ (I_4) = -1\ ,\ b - Y = -3 < 0 \Rightarrow I_4 \in N^-$$

$$N^+ = \{I_1\} \qquad N^- = \{I_3\ ,I_4\}$$

Por definición, se obtiene un camino que vaya de cualquier elemento del conjunto N^+ al conjunto N^- y se obtiene el vector de incidencias del camino.

$$P = I_1\ ,J_3\ ,I_2$$

$$e_p = (0\ ,0\ ,1\ ,0\ ,0)$$

y $\alpha = \text{mín}\ (7\ ,(5-0),\ 4) = 4$

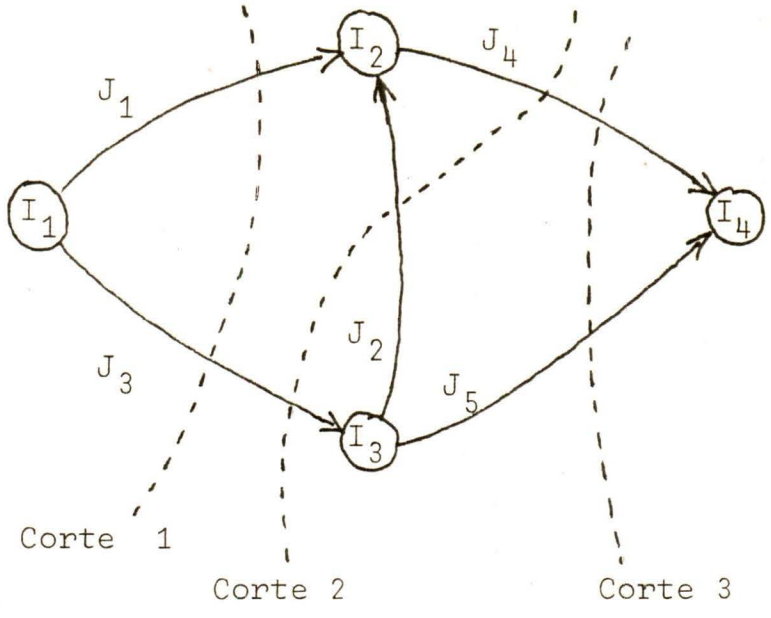
$$X' = X + e_p\alpha = (1\ ,0\ ,0\ ,0\ ,1) + (0\ ,0\ ,1\ ,0\ ,0)\cdot(4)$$

$$X = (1\ ,0\ ,4\ ,0\ ,1)$$

y la nueva divergencia es:

$$Y\ (I_1) = 5 \qquad b - Y = 3 > 0$$
$$Y\ (I_2) = -1 \qquad b - Y = 0$$
$$Y\ (I_3) = -3 \qquad b - Y = 0$$
$$Y\ (I_4) = -1 \qquad b - Y = -3 < 0$$

$$N^+ = \{I_1\} \qquad N^- = \{I_4\}$$



Corte 1 : $S = \{I_1\}$ $\quad Q^+ = \{J_1, J_3\}$ $\quad \Theta(I_2) = J_1$

Corte 2 : $S = \{I_1, I_2\}$ $\quad Q^- = \{J_3, J_4\}$ $\quad \Theta(I_3) = J_3$

Corte 3 : $S = \{I_1, I_2, I_3\}$ $\quad Q^+ = \{J_4, J_5\}$ $\quad \Theta(I_4) = J_4$

$P: = I_1, J_1, I_2, J_4, I_4$

$e_p = (1, 0, 0, 1, 0)$

$\alpha = \text{Mín}(3, 3, 2, 4) = 2$

$X = (3, 0, 4, 2, 1)$

color = (r̂ ,b ,b ,b ,b)

Se calcula la nueva divergencia

$Y(I_1) = 7 \quad b - Y = 1 > 0 \Rightarrow I_1 \in N^+$

$Y(I_2) = -1 \quad b - Y = 0$

$Y(I_3) = -3 \quad b - Y = 0$

$Y(I_4) = -3 \quad b - Y = -1 < 0 \Rightarrow I_4 \in N^-$

Corte 1 $S = \{I_1\} \quad Q^+ = \{r, J_2\} \quad \theta(I_3) = J_3$

Corte 2 $S = \{I^1, I_3\} \quad Q^+ = \{J_1, J_2, J_5\} \quad \theta(I_2) = J_2$

Corte 3 $S = \{I_1, I_2, I_3\} \quad Q^+ = \{J_4, J_5\} \quad \theta(I_4) = J_4$

$\Rightarrow$ P: $I_1, J_3, I_3, J_2, I_2, J_4, I_4$

$e_p = (0, 1, 1, 1\ 0)$

$\alpha = \text{Mín}\ (1, 1\ (5-4), (1-0), (4-2)) = 1$

$X = (3, 1, 5, 3, 1)$

$\text{div}(X) = Y(I_1) = 8$

$Y(I_2) = -1$

$Y(I_3) = -3$

$Y(I_4) = -4$

LAS VARIABLES

TIPOS DE VARIABLES

SUBIN: Conjunto de enteros comprendidos entre el 0 y el 100 inclusive.

ARR: Arreglo vectorial de 50 elementos del conjunto de los enteros. Utilizado para las variables que representan a los nodos, nombres de nodos y elementos del camino.

ARRE: Arreglo vectorial de 100 elementos del conjunto de enteros. Utilizado para las variables que representan a los arcos, nombres de arcos, etc.

ARREAL: Arreglo vectorial de 100 elementos del conjunto de los reales. Utilizado para las variables que representan flujos, diferencias de flujos, etc.

TIMAT: Arreglo bidimensional de 50 por 100 elementos del conjunto de los enteros, empleados para definir la matriz de incidencias y su respectiva variable para grabarse en un archivo.

VARIABLES:

MATIN: Variable del tipo TIMAT, empleada para representar la red númericamente mediante la matriz de incidencias.

VN: Variable del tipo ARR, que representa el vector de nombres de nodos que intervienen en la red, para accesar por medio de este MATIN para que los nombres de los nodos no tengan que ir en orden secuencial sino en el orden que el usuario desee.

VA: Variable del tipo ARRE que representa al vector de nombres de arcos que intervienen en la red, para accesar MATIN por medio de este vector. Tiene la misma finalidad que VN.

BAR: Variable del tipo ARR, empleado en la elección del siguiente nodo para el cual se pedirá información, ésto es para facilitar el armado de la red y evitar que sea necesario nombrar los nodos en orden ascendente y de uno en uno.

CLAVE: Vector de 50 elementos del tipo lógico para evitar que se lea la información concerniente a un nodo más de una vez, se inicializa con valor TRUE y se cambia a FALSE al momento en que el programa lo elige para pedir información, sólo estando TRUE puede ser elegido.

COLOR: Vector de 100 elementos del tipo "CHAR" que se le asignan a cada arco dependiendo de si tiene o no capacidad de aumentar su flujo actual. Tiene valor "B" si es blanco, o sea que puede aumentar su flujo y si es al contrario es "R" de rojo.

ITER: Variable del tipo ARRE, que determina el nombre del nodo encontrado en cada corte o iteración.

THETA: Variable del tipo ARRE que determina el nombre del arco por el cual se llega al nodo ITER.

E: Vector del tipo ARRE que representa a los arcos involucrados en un corte, equivalente a la suma vertical en la matriz de incidencias para todo nodo elemento del conjunto "S" de un corte.

EP: Vector de incidencias para un camino "P"

R: Vector utilizado como contador individual para cada nodo y así acomodar las posiciones del cursor en la lectura de información.

CAM: Vector de nombres de arcos empleados en un camino.

VCS: Vector de nombres de nodos que representa al conjunto "S" de nodos al efectuar un corte.

VF: Arreglo de reales que indica los flujos involucrados en un camino; empleado para encontrar el mínimo de éstos (ALFA).

ALFA: Número máximo en que se puede aumentar el flujo de los arcos involucrados en el camino.

CAP: Vector de reales que contiene las capacidades máximas de cada flujo para cada arco de la red.

CAPMIN: Vector de reales que contiene la restricción de mínimo flujo requerido para cada arco de la red.

X: Vector de reales que contiene los flujos actuales para cada arco de la red.

BOO, BU: Variables del tipo lógico usadas como identificadores para validar la información de entrada al cambiar la posición de un arco en la red. Si la información cambiada es válida se hacen TRUE y se acepta, de lo contrario, la información se pide de nuevo.

MAXIFLUX: Condición del tipo lógico que vale FALSE y cambia a TRUE cuando se llega al máximo flujo.

CN: Conjunto de nombres de nodos, incluye sólo los nombres de los nodos empleados en la red.

CA: Conjunto de nombres de arcos, incluye sólo los nombres de los arcos empleados en la red.

CS: Conjunto de nombres de nodos, que incluye sólo los nombres de los nodos que son elementos del conjunto "S" del corte.

CSP: Conjunto de nombres de nodos que incluye sólo los nombres de los nodos que son elementos del conjunto "N/S"

I,J,K,L,M: Variables enteras utilizadas como contadores generalmente para los estatutos FOR, de uso variado, excepto I y J, los que generalmente se emplearon para variar nodos y arcos respectivamente, para darle un mayor orden a la programación.

A: Variable entera empleada para determinar el número de arcos de la red.

NUNO: Variable entera empleada para determinar el número de nodos de la red.

IP: Variable entera usada temporalmente como el nodo al que llega cada arco.

INU: Variable entera, representa el siguiente nodo a leer.

NNU: Variable entera que representa el nodo nuevo a leer.

SPRI: Variable entera que denota el último nodo leído y es equivalente a I_r, en el algoritmo, el programa buscará un camino entre I_0 y SPRI.

EL PROGRAMA

El programa está dividido en dos partes, al terminarse - cada una de éstas, se ejecuta la otra.
La primera de estas partes, forma la red; lee la información necesaria respectiva de cada nodo; arcos que salen y nodo al que estos llegan. Acepta los cambios voluntarios respecto a la posición de los arcos en la red y - graba ésta en el disco, posteriormente se ejecuta la segunda parte del programa, la cual lee del disco la red y la resuelve por cualquiera de los tres métodos respectivos dependiendo del tipo de problema.

La formación de una nueva red toma lugar en el procedimiento "NUEVA RED" después de haber inicializado todas las variables en el procedimiento "INICIALIZA".

El procedimiento "NUEVA RED" comienza asignando el valor inicial de 1 a la I, el cual siempre será el nodo inicial de la red, se introduce el valor de I al vector de nombres de nodos y al conjunto de nodos, para después llamar al procedimiento "LENOD", enviando como parámetro de valor la posición vertical en la pantalla del cursor.

En este procedimiento, "LENOD", se escribe el letrero - "NODO →" seguido del número de nodo en cuestión para luego posisionarse el cursor debajo del letrero "ARCO QUE SALE", donde se lee un entero entre 0 y 99, el cual es - el nombre del arco que sale del nodo "I", la lectura y la validación de dicho entero, se efectúa en el procedimiento "LEEEN" que se llama desde "LENOD" con la posición horizontal y vertical del cursor como parámetros de valor y "J" como parámetro variable, el cual es el nombre del arco que sale del nodo "I" y no puede ser igual a ningún otro nombre de arco.

Posteriormente, el cursor se posisiona debajo del letrero "NODO RECEPTOR" y por medio del procedimiento "LEEEN"

acepta de pantalla un entero, ahora entre 1 y 50 y que - no sea igual al nodo emisor.

Obtenidos estos números, se llama al procedimiento "FORMATIN" con los nombres de: El nodo emisor (I), nodo receptor (IP) y el arco entre ellos (M), como parámetros de valor, colocando un "1" en el renglón "I" columna "M" de la matriz de incidencias, así como un "-1" en el renglón "IP" de la misma columna de la misma matriz de incidencias "MATIN".

Este proceso, desde la llamada a "LENOD, se repite hasta - que no se ve el siguiente nombre de arco que sale, o sea, cuando ya no hay más arcos que salen, se oprime la tecla "RETURN", con lo que se sale del procedimiento "LENOD" para entrar en el procedimiento "ELIGE", el cual, de la lista de nombres de nodos afectados en la columna "NODO RECEPTOR" escoge el menor de los que no han pasado aún por la columna "NODO EMISOR".

Todo ésto, se repite hasta que no exista un nodo de la - lista de nodos receptores que no haya pasado por la columna de nodos emisores.

Habiendo llegado hasta aquí, ya se tiene una red de "N" nodos, y "A" arcos, representado numericamente mediante la matriz de incidencias. Si existiese un error en la posición de un arco, es decir, si uno o más de los arcos ya existentes está mal en cuanto al nodo del que sale y el nodo al que llega, se puede corregir, al responder oprimiendo la letra "S" a la pregunta "¿ DESEAS CAMBIAR LA INFORMACION DE UN ARCO ?" con lo que se ejecutaría el procedimiento "CAMBIMAT" el cual llamaría al procedimiento "CORRIMAT".

Efectuados los cambios necesarios, se graba en disco la representación número ica de la red y se llama la segunda parte del programa.

LOS METODOS

La segunda parte del programa, la cual resuelve los tres tipos de problemas con su respectivo algoritmo, fué subdividido en dos partes, debido a la limitada memoria de la computadora disponible, la primera parte de ésta sección resuelve los dos primeros tipos de problemas dejando el problema de las divergencias para la segunda parte. Ambas partes son muy similares excepto por algún procedimiento que difiere un poco de la otra parte.

En éstos, se lee la información necesaria para resolver el método, incluyendo la matriz formada previamente, se inicializan las variables necesarias en el procedimiento "INICIALIZA" y se llama, desde el programa principal, al procedimiento "MENU", el cual pregunta al usuario - cuál de los tres métodos se desea usar, y de ser el tercero, el de las divergencias, se llama a la otra parte de este subprograma.

Cuando se elige el primer método, el cual sólo busca un camino entre el nodo inicial y el terminal, se llama al procedimiento "FORMACAM", el cual realiza los cortes, sumando para cada columna de la matriz de incidencias "MATIN", todos los renglones elementos del vector "VCS"; "renglones" representan a los nodos y "columnas" a los arcos. Cuando para una columna la suma es igual a 1 y el arco correspondiente es de color blanco se le da al identificador "ID" el nombre de la columna.

Hecho ésto, se procede a buscar el nodo receptor del arco "ID" y se agrega éste al vector "VCS" ésto se realiza en el procedimiento "BUSCA" al llamarlo con un "-1" e "ID" como parámetros de valor y "VCS" como parámetro variable; ya se tiene ahora el nodo de la actual iteración y su correspondiente arco de la función "THETA"

Se continúa efectuando cortes hasta llegar a SPRI o bien que no se encuentre ningún identificador y determine que no hay camino.

Al llegar a SPRI, se llama al procedimiento "SACAM", el cual, a partir de los valores obtenidos en las iteraciones, encuentra el camino.

Después de aquí sólo resta imprimir el camino llamando - al procedimiento "IMPRICAM".

En el segundo de los métodos el procedimiento es muy similar, pero antes de empezar llama sucesivamente a los procedimientos "LEEMAXCAP" y "LEEMINFLUJO", los cuales "Leen" la capacidad máxima y mínima de flujo de cada arco respectivamente, y al leer la capacidad mínima, inmediatamente se le asigna dicha cantidad al flujo actual y "pinta" la red llamando al procedimiento "PINTA RED", en el cual se le asigna al vector "COLOR" la letra "R" a la posición del número de arco cuyo flujo ya no es posible incrementar.

Formada y "pintada" la red se procede a sacar el camino por medio de "FORMACAM" con la diferencia del anterior método que después de llamar a "SACAM", en vez de llamar a "IMPRICAM" se llama a "SACAFLUJO" seguido de - "PINTARED"; en el procedimiento "SACAFLUJO" se calcula el máximo número en que se debe incrementar el flujo de todos los arcos, del camino recién obtenido, y en "PINTARED" se le asigna la letra "R" a los nuevos arcos cuyo flujo ya llegó al límite.

Se repite el proceso, desde que se llamó al procedimiento "FORMACAM" en delante, hasta que se llegue al óptimo, el cual se sabe que llegó cuando ya no exista un camino posible cuyos arcos sean todos blancos.

Habiendo llegado a ésto, se llama al procedimiento - - "IMPRIFLUJO" el cual dá a conocer el flujo que debe tener cada arco para que el flujo total de la red sea el máximo.

El tercer método, que encuentra un flujo tal que cumpla con las restricciones de divergencia comienza de manera muy similar al segundo método, lee la capacidad máxima - así como el mínimo flujo que debe pasar por cada uno de los arcos y "pinta" la red (LEEMAXCAP; LEECAPMIN; PINTARED).

Después de obtenido ésto, lee las restricciones de divergencia para cada uno de los nodos al llamar al procedimiento "LEEDIVER", el cual se repite hasta que la suma de estas divergencias dé un valor de cero.

Obtenido ya todos los datos para la red, se empieza el algoritmo para resolver el tercer tipo de problema. Comienza inicializando la condición lógica "BU" como "FALSE" para después calcular la divergencia en los nodos con el - flujo actual mediante la resta de la suma de los flujos que entran a cada nodo, de los flujos que salen de éste - mismo, esto se realiza al llamar "SACAYE".

Después, repite un proceso similar al "FORMACAM", el - "HAZCAM", con la diferencia que en vez de buscar un posible camino entre I_0 e I_r, busca un posible camino entre - cualquier nodo elemento del conjunto N^+ (representado en el programa como "CS") y cualquier nodo elemento del conjunto N^- (representado por la variable "CSP"); se repite "HAZCAM" hasta que el procedimiento llegue a un nodo elemento del conjunto "CSP". Durante dicho proceso, en cada iteración se obtuvieron los valores de ITER (nodo) y THETA (ITER) de ITER, nodo para el camino y THETA (ITER) respectivo arco por el cual se llega al nodo ITER, elementos necesarios para obtener el camino en el procedimiento - - "SACAM", el cual es llamado inmediatamente después de haber llegado el camino, al conjunto "CSP".

Habiendo obtenido ya el camino, se encuentra el nuevo - flujo para la red, que resulta de multiplicar el número "ALFA" por el vector CP obtenido en SACAM y sumarlo al - antiguo vector de flujo X.

Después de ésto, se llama a "PINTARED" y se repite el algoritmo desde la inicialización de la condición lógica "BU".

Este proceso se repite hasta que se llga a un flujo tal, que la divergencia de todos los nodos para ese flujo, - sea igual a la divergencia requerida en las restricciones para luego dar como resultado el flujo final para cada - arco, llamando al procedimiento "IMPRIFLUJO".

LAS CONCLUSIONES

Después de haber consultado y estudiado diversos métodos de optimización de flujo en redes, puedo concluir que la elección del método de cortes, del Algoritmo de Ford-Fulkerson, así como el de Gale-Hoffman fué una decisión correcta, ya que de los modelos que pude encontrar (los cuales no fueron muchos, debido a la escasa bibliografía que explica éstos conceptos ampliamente) éstos, que usan el mismo principio, son los de más sencilla comprensión, así como los que usan el algoritmo de más amplia aplicación, ya que para diferentes tipos de modelos, las variaciones en el procedimiento son casi nulas, por lo que al hacer un programa computacional, éste tiene una aplicación más generalizada.

No por ésto aseguro que sea el mejor método, tal vez lo sea, o tal vez exista uno mejor pero dados los métodos que tuve oportunidad de analizar y considerando el poco tiempo que tiene el estudio de las redes de ser parte de la optimización matemática determinística, dichos algoritmos deben ser, si no los más optimizados, de los mejores y más completos.

En cuanto al programa, se refiere, considero que el uso del lenguaje "PASCAL" para programar, es una gran ventaja dada su estructura y facilidad de entender la lógica, ya que de ser necesario, es posible adoptar el programa en otra máquina de mayor capacidad, pudiendo así aumentar los alcances de este programa, ya que está muy limitado por la capacidad de memoria que tiene la APPLE II y la lentitud de ejecución para problemas de tamaño considerable.

La variedad de posibles problemas por resolver por medio de éstos métodos matemáticos, es tan amplia como la imaginación de uno la quiera hacer, ya que como existen in-

finidad de situaciones que se pueden expresar como modelos abstractos que representan redes, tantas así pueden ser las aplicaciones del método.

Como posible ejemplo podría nombrarse alguna discusión - teórica, con los posibles caminos para resolver algún - problema o teoría.

Más no sólo como modelos abstractos se presentan las opciones de uso de éstos métodos, pueden ser físicamente - existentes, como por ejemplo, el mejorar la circulación vehicular de un sector de una ciudad, usando flujos y capacidades de las vías a diferentes horas, y en diferentes condiciones.

Otro ejemplo podría ser el problema de la elección de rutas para mantener un inventario determinado en una cadena de tiendas o distribuidoras, utilizando el tercer método para ésto y tomando como divergencias la producción de las plantas y el inventario de las distribuidoras.

Las limitaciones para el uso de este programa son mucho mayores que las que tienen por sí mismos los métodos empleados, debido a la reducida capacidad de la máquina, - así como el sacrificio de variedad en opciones con el - fin de facilitar la ejecución del programa y para evitar hacer ésta más lenta aún. Dejándose al usuario la necesaria comprensión del concepto de la red para poder aplicar el programa a algún ejemplo.

EL APENDICE 'A'

```
          (*$S+*)
PROGRAM ELPEF;

USES APPLESTUFF,CHAINSTUFF;

TYPE          SUBIN=0..100;
                ARR=ARRAY[0..50] OF INTEGER;
               ARRE=ARRAY[0..100] OF INTEGER;
              TIMAT=ARRAY[0..50,0..100] OF INTEGER;


VAR     IP,M,A,I,K,CON,
        SPRI,NUNO,
        INU,J                 :INTEGER;
        CONT,IDE,
        BAR,VN                :ARR;
        VA,R                :ARRE;
        MATIN                     :TIMAT;
        CLAVE                     :ARRAY[0..50] OF BOOLEAN;
        BU,BOO                    :BOOLEAN;
        MATARCH                   :FILE OF TIMAT;
        VEC1ARCH                  :FILE OF ARRE;
        VEC2ARCH                  :FILE OF ARR;
        CN,CA,CSP               :SET OF SUBIN;
        TK                        :CHAR;


PROCEDURE ELIGE(BAR:ARR;A:INTEGER;VAR  NNU:INTEGER);

                VAR  L,IC:INTEGER;


BEGIN
IC:=0;
SPRI:=0;

   NNU:=100;
      FOR L:=1 TO A DO
      BEGIN
         IF (BAR[L]<NNU) AND (CLAVE[BAR[L]]) THEN
         BEGIN
            NNU:=BAR[L];
            IC:=L;
         END;
         IF BAR[L] > SPRI
            THEN SPRI:=BAR[L]
      END;
      CLAVE[BAR[IC]]:=FALSE;
      NNU:=BAR[IC];
      CSP:=[SPRI];

END;



PROCEDURE GETT(VAR WK:CHAR);

BEGIN

   WHILE NOT KEYPRESS DO;
      READ(KEYBOARD,WK)

END;
```

```
PROCEDURE LEEEN(XE,Y:INTEGER; VAR  LEIDA:INTEGER);

                  VAR       I,X,LEID,N:INTEGER;
                                ESP,CH:CHAR;


  BEGIN
   GOTOXY(XE,Y);
   WRITE(' ');
   LEIDA:=0;
   I:=0;
   ESP:=' ';
   REPEAT
       X:=XE+I;
       REPEAT
          GOTOXY(X,Y);
          WRITE(ESP,CHR(8));
          GETT(CH);
          N:=ORD(CH);
          IF (N IN [48..57]) AND (I<2)
             THEN
             BEGIN
                I:=I+1;
                WRITE(N-48);
             END
             ELSE
             IF NOT (N IN [8,32])
                THEN N:=0
       UNTIL (N IN [8,32,48..57]);
       IF N IN [48..57]
          THEN
             LEIDA:=LEIDA*10+N-48
          ELSE IF N = 32
             THEN I:=3
             ELSE
             BEGIN
                IF I>0
                   THEN I:=I-1;
                LEIDA:=LEIDA DIV 10
             END
    UNTIL (I=3) ;
  END;


PROCEDURE FORMATIN(I,M,IP:INTEGER;VAR CO:INTEGER);


    BEGIN

       IF (MATIN[I,M]=0) AND (MATIN[IP,M]=0) THEN
          BEGIN
             MATIN[I,M]:=1;
             MATIN[IP,M]:=-1;
             CO:=CO+1;
             A:=A+1;
             VA[A]:=M;
             CA:=CA+[M];
             BAR[A]:=IP
          END

    END;
```

```
        PAGE(OUTPUT);
        WRITE('  NODO');
        GOTOXY(17,0);WRITE('ARCO');
        GOTOXY(30,0);WRITE('NODO');
        GOTOXY(1,1); WRITE('EMISOR');
        GOTOXY(15,1); WRITE('QUE SALE');
        GOTOXY(28,1);WRITE('RECEPTOR')

    END;



PROCEDURE LENOD(REN:INTEGER);

                VAR  RENG :INTEGER;

    BEGIN

        RENG:=REN;
            WHILE RENG > 21 DO
                RENG:=RENG-19;
        GOTOXY(1,RENG);
        WRITE('NODO -> ',I);
        REPEAT
            RENG:=REN+CONT[I];
            WHILE RENG > 21 DO
                RENG:=RENG-19;
            REPEAT
                LEEEN(17,RENG,J)
            UNTIL NOT (J IN CA);
            IF J <> 0 THEN BEGIN
                M:=J;
                REPEAT
                    LEEEN(31,RENG,IP);
                UNTIL IP IN ([1..50]-[I]);
                FORMATIN(I,M,IP,CONT[I]);
            END;
            IF (RENG = 3) AND (REN+CONT[I] > 4)
                THEN TITULA;

        UNTIL J=0 ;

    END;



PROCEDURE NUEVARED;

    BEGIN

        TITULA;
        REPEAT
            I:=INU;
            CONT[I]:=0;
            R[I]:=A+3;
            NUNO:=NUNO+1;
            VN[NUNO]:=I;
            CN:=CN+[I];
            LENOD(R[I]);
            ELIGE(BAR,A,INU)

        UNTIL INU=0;

    END;
```

```
PROCEDURE INICIALIZA;
          VAR K,M:INTEGER;

  BEGIN
     PAGE(OUTPUT);
     FILLCHAR(MATIN,10302,0);
     FOR K:=0 TO 100 DO
     BEGIN
          VA[K]:=0;
     END;

     FOR K:=0 TO 50 DO
     BEGIN
        CLAVE[K]:=TRUE;
          BAR[K]:=0;
           VN[K]:=0;
     END;

     A:=0;
     NUNO:=0;
     INU:=1;
     CN:=[];
     CA:=[];
     CSP:=[];
     CLAVE[0]:=FALSE;
     CLAVE[1]:=FALSE;
     PAGE(OUTPUT);

  END;



PROCEDURE IMPRIMAT(QW:CHAR);
          VAR  WK:CHAR;

  BEGIN
     IF QW='S' THEN
     BEGIN
        MATIN[0,0]:=SPRI;
        REWRITE(MATARCH,'#5:MATRIZ.DATOS');
        MATARCH^:=MATIN;
        PUT(MATARCH);
        CLOSE(MATARCH,LOCK);
           VN[0]:=NUNO;
           VA[0]:=A;
        REWRITE(VEC2ARCH,'#5:VEC2.DATOS');
        VEC2ARCH^:=VN;
        PUT(VEC2ARCH);
        CLOSE(VEC2ARCH,LOCK);
        REWRITE(VEC1ARCH,'#5:VEC1.DATOS');
        VEC1ARCH^:=VA;
        PUT(VEC1ARCH);
        CLOSE(VEC1ARCH,LOCK);
     END;
     WRITELN;WRITELN;WRITELN;WRITELN;
     WRITE('DESEAS VER LA MATRIZ DE INC.? <S/*>');
     GETT(WK);
     IF WK = 'S'
     THEN BEGIN
        PAGE(OUTPUT);
        WRITE('       ');
        FOR K:=1 TO A DO
           WRITE(' J',VA[K]:2);
        WRITELN;WRITELN;
        FOR M:=1 TO NUNO
```

```
        DO BEGIN
           WRITE(' I',VN[M]:2,': ');
           FOR K:=1 TO A DO
              WRITE('  ',MATIN[VN[M],VA[K]]:2);
           WRITELN;WRITELN
        END
     END;
  END;



PROCEDURE CORRIMAT(CON:INTEGER;IDE:ARR);
                VAR  NNU  :INTEGER;


  BEGIN

     PAGE(OUTPUT);
     GOTOXY(22,0);WRITE('NODO');
     GOTOXY(31,0);WRITE('NODO');
     GOTOXY(21,1);WRITE('EMISOR');
     GOTOXY(29,1);WRITE('RECEPTOR');

     FOR J:=1 TO CON DO
        REPEAT
           BU:=FALSE;
           BOO:=FALSE;
           GOTOXY(1,J+3);
           FOR I:=1 TO NUNO DO
              MATIN[VN[I],IDE[J]]:=0;
           WRITE('PARA EL ARCO-> ',IDE[J]);
           LEEEN(24,J+3,NNU);
           IF NNU IN CN
              THEN BU:=TRUE;
           K:=NNU;
           MATIN[NNU,IDE[J]]:=1;
           LEEEN(33,J+3,NNU);
           IF NNU IN (CN-[K])
              THEN BOO:=TRUE;
           MATIN[NNU,IDE[J]]:=-1
        UNTIL BU AND BOO

  END;



PROCEDURE CAMBIMAT;
                VAR   WK :CHAR;


  BEGIN

     REPEAT
        WRITELN;WRITELN;
        WRITELN('DESEAS CAMBIAR LA INFORMACION');
        WRITE('                    DE ALGUN ARCO? <S/*>');
        GETT(WK);
        IF WK='S' THEN
        BEGIN
           FOR J:=0 TO 50 DO
              IDE[J]:=0;
           WRITELN;WRITELN;
           WRITE('CUANTOS ARCOS QUIERES CAMBIAR-> ');
           READLN(CON);
```

```
            FOR J:=1 TO CON DO
            BEGIN
               WRITE(J,') N DE ARCO : ');
               READLN(IDE[J]);
               WRITELN
            END;
            CORRIMAT(CON,IDE)
         END
      UNTIL WK <> 'S'

   END;



PROCEDURE NUEVAOVIEJA;

   BEGIN

      PAGE(OUTPUT);
      WRITELN;WRITELN;WRITELN;
      WRITELN('DESEAS EJECUTAR EL PROGRAMA CON :');
      WRITELN;WRITELN;
      WRITELN('  1)  UNA RED NUEVA ');
      WRITELN;
      WRITELN('  2)  LA RED ANTERIOR ');
      GOTOXY(38,3);
      REPEAT
         GETT(TK);
      UNTIL (TK='1') OR (TK='2');
      PAGE(OUTPUT);

   END;


 BEGIN

      PAGE(OUTPUT);

      INICIALIZA;

      NUEVAOVIEJA;

      IF TK='1'
      THEN
      BEGIN
         NUEVARED;

         IMPRIMAT('N');

         CAMBIMAT;
            K:=0;
            FOR J:=1 TO 99 DO
               IF J IN CA THEN
               BEGIN
                  K:=K+1;
                  VA[K]:=J;
               END;
            J:=0;
            FOR I:=1 TO 50 DO
               IF I IN CN THEN
               BEGIN
                  J:=J+1;
                  VN[J]:=I;
               END;

         IMPRIMAT('S');

         PAGE(OUTPUT);
      END;
```

```
        GOTOXY(16,10);
        WRITE('MOMENTO...');
        GOTOXY(56,10);
        SETCHAIN('N5:CONTI');

END.
```

```
(*$S+*)
      PROGRAM ELPEF;

      USES APPLESTUFF,CHAINSTUFF;

      TYPE          SUBIN=0..100;
                      ARR=ARRAY[0..50] OF INTEGER;
                     ARRE=ARRAY[0..100] OF INTEGER;
                    TIMAT=PACKED ARRAY[0..50,0..100] OF INTEGER;
                  ARREAL=ARRAY[0..100] OF REAL;


      VAR       IP,M,A,K,I,KO,
                SPRI,ID,NUNO,
                J,NAC,NNU              :INTEGER;
                CAM,VCS,
                VN,EP                :ARR;
                E,VA,
                THETA,ITER             :ARRE;
                MATIN                  :TIMAT;
                COLOR                  :ARRAY[0..100] OF CHAR;
                MAXIFLUX,DIVERYA,BU    :BOOLEAN;
                MATARCH                :FILE OF TIMAT;
                VEC1ARCH               :FILE OF ARRE;
                VEC2ARCH               :FILE OF ARR;
                CN,CA,CSP,CS           :SET OF SUBIN;
                CAP,X,CAPMIN,B,Y       :ARREAL;
                VF                     :ARRAY[0..50] OF REAL;
                ALFA,ALF1,DIF,KR       :REAL;
                TK,QQ                  :CHAR;
                OP                     :TEXT;
                PID                    :STRING;


   PROCEDURE GETT(VAR WK:CHAR);

   BEGIN

      WHILE NOT KEYPRESS DO;
         READ(KEYBOARD,WK)

   END;



PROCEDURE PINTARED;

      BEGIN

         FOR J:=1 TO A DO
            IF (CAP[VA[J]]-X[VA[J]]) <= (0.00001)
               THEN COLOR[VA[J]]:='R';

      END;
```

```
PROCEDURE BUSCA(CONSTA,ARCO:INTEGER;VAR NODO:INTEGER);

        BEGIN

           FOR M:=1 TO NUNO DO
              IF MATIN[VN[M],ARCO]=CONSTA
                 THEN NODO:=M

        END;


PROCEDURE SACAM(K:INTEGER;VAR CAM:ARR);
                        VAR    WK:INTEGER;

        BEGIN

           I:=-1;
           WK:=K;
           REPEAT
              I:=I+1;
              CAM[I]:=ITER[WK];
              I:=I+1;
              CAM[I]:=THETA[ITER[WK]];
              EP[CAM[I]]:=EP[CAM[I]]+1;
                 BUSCA(1,CAM[I],ITER[WK-1]);
              WK:=WK-1;
           UNTIL  (ITER[WK] IN CS) OR (WK=0);
           CAM[I+1]:=ITER[WK];
        END;


PROCEDURE FORMACAM;

     BEGIN

        K:=0;
        REPEAT
           K:=K+1;
           ID:=0;
           FOR J:=1 TO A DO BEGIN
              E[VA[J]]:=E[VA[J]]+MATIN[VCS[K],VA[J]];
              IF (E[VA[J]]=1) AND (ID=0) AND (COLOR[VA[J]]='B')
                 THEN ID:=VA[J];
           END;
           IF ID <> 0 THEN
              BEGIN
                 BUSCA(-1,ID,VCS[K+1]);
                 ITER[K]:=VCS[K+1];
                 THETA[ITER[K]]:=ID
              END;
        UNTIL (ID=0) OR (VCS[K]=SPRI) OR (K=98);
        IF VCS[K]=SPRI
           THEN SACAM(K,CAM)
           ELSE BEGIN
              MAXIFLUX:=TRUE;
           END

     END;
```

```
PROCEDURE INICIALIZA;

          VAR K,M:INTEGER;

  BEGIN

     FOR K:=0 TO 100 DO
     BEGIN
        COLOR[K]:='B';
        THETA[K]:=0;
         ITER[K]:=0;
          CAP[K]:=0;
           VA[K]:=0;
            E[K]:=0;

          CAP[K]:=0.0;
      CAPMIN[K]:=0.0;
             X[K]:=0.0;
     END;

     FOR K:=0 TO 50 DO
     BEGIN
           CAM[K]:=0;
           VCS[K]:=0;
            VN[K]:=0;
            EP[K]:=0;
            VF[K]:=0.0;
             Y[K]:=0.0;
             B[K]:=0.0;
     END;

     A:=0;
     K:=0;
     NUNO:=0;
     NNU:=0;
     ALF1:=100000.0;
     VCS[1]:=1;
     CN:=[];
     CA:=[];
     CSP:=[];
     CS:=[1];
     DIVERYA:=FALSE;
     MAXIFLUX:=FALSE;

  END;



PROCEDURE DISIMPRIMAT;

  BEGIN

     REWRITE(OP,PID);
     WRITELN(OP);WRITELN(OP);WRITELN(OP);WRITELN(OP);
     PAGE(OUTPUT);
     WRITE(OP,'        ');
     FOR K:=1 TO A DO
        WRITE(OP,' J',VA[K]:2);
     WRITELN(OP);WRITELN(OP);
     FOR M:=1 TO NUNO
     DO BEGIN
        WRITE(OP,' I',VN[M]:2,': ');
        FOR K:=1 TO A DO
           WRITE(OP,'  ',MATIN[VN[M],VA[K]]:2);
        WRITELN(OP);WRITELN(OP);
```

```
        END;
        GETT(TK);
    END;



PROCEDURE IMPRIMAT;
            VAR  WK:CHAR;

    BEGIN
        WRITELN;WRITELN;WRITELN;WRITELN;
        WRITE('DESEAS VER LA MATRIZ DE INC.? <S/*>');
        GETT(WK);
        IF WK = 'S'
        THEN BEGIN
            PAGE(OUTPUT);
            WRITE('          ');
            FOR K:=1 TO A DO
                WRITE(' J',VA[K]:2);
            WRITELN;WRITELN;
            FOR M:=1 TO NUNO
            DO BEGIN
                WRITE(' I',VN[M]:2,': ');
                FOR K:=1 TO A DO
                    WRITE('  ',MATIN[VN[M],VA[K]]:2);
                WRITELN;WRITELN;
            END;
            GETT(TK);
        END;

    END;



PROCEDURE VIEJARED;

    BEGIN
        RESET(MATARCH,'N5:MATRIZ.DATOS');
        MATIN:=MATARCH^;
        GET(MATARCH);
        CLOSE(MATARCH);
        RESET(VEC1ARCH,'N5:VEC1.DATOS');
        RESET(VEC2ARCH,'N5:VEC2.DATOS');
        VA:=VEC1ARCH^;
        GET(VEC1ARCH);
        VN:=VEC2ARCH^;
        GET(VEC2ARCH);
        CLOSE(VEC1ARCH);
        CLOSE(VEC2ARCH);
        A:=VA[0];
        NUNO:=VN[0];
        SPRI:=MATIN[0,0];

    END;
```

```
PROCEDURE LEEMAXCAP;

   BEGIN

      PAGE(OUTPUT);
         PAGE(OUTPUT);
         WRITELN;    WRITELN;    WRITELN;
         WRITELN('    DAME LA CAPACIDAD MAXIMA DE :');
         WRITELN;WRITELN;
         FOR J:=1 TO A DO
         BEGIN
            WRITE(' ARCO J',VA[J],'  : ');
            READLN(CAP[VA[J]]);
            WRITELN;
         END;
   END;


PROCEDURE LEEMINFLUJO;

   BEGIN

      PAGE(OUTPUT);
      WRITELN;WRITELN;
      WRITELN(' EXISTEN RESTRICCIONES DE');
      WRITELN;
      WRITE('    CAPACIDAD MINIMA ?  <S/*> ');
      GETT(TK);
      IF TK = 'S' THEN
      BEGIN
         PAGE(OUTPUT);
         WRITELN;    WRITELN;    WRITELN;
         WRITELN('    DAME LA CAPACIDAD MINIMA DE :');
         WRITELN;WRITELN;
         FOR J:=1 TO A DO
         BEGIN
            REPEAT
            WRITE(' ARCO J',VA[J],'  : ');
            READLN(X[VA[J]]);
            IF ((CAP[VA[J]]-X[VA[J]]) < -0.0001)
               THEN WRITE(CHR(7));
            UNTIL ((CAP[VA[J]]-X[VA[J]]) > -0.0001);
            WRITELN;
         END;
      END;

   END;



PROCEDURE SACAFLUJO ;

   BEGIN

      NAC:=0;
      FOR J:=1 TO A DO
      BEGIN
         ALFA:=EP[VA[J]]*(CAP[VA[J]]-X[VA[J]]);
         IF ALFA > 0
         THEN BEGIN
            NAC:=NAC+1;
            VF[NAC]:=ALFA
         END
```

```
        END;
        ALFA:=10000;
        FOR J:=1 TO NAC DO
        BEGIN
           IF VF[J] < ALFA
              THEN ALFA:=VF[J];
        END;
        IF ALF1 < ALFA
           THEN ALFA:=ALF1;
        FOR J:=1 TO A DO
           X[VA[J]]:=X[VA[J]]+EP[VA[J]]*ALFA;

    END;


PROCEDURE IMPRIFLUJO;

     BEGIN

        DISIMPRIMAT;
        WRITELN(OP);WRITELN(OP);WRITELN(OP);
        WRITELN(OP,'   >>>  EL FLUJO "X" EN LOS ARCOS ES :');
        WRITELN(OP);WRITELN(OP);WRITELN(OP);
        FOR J:=1 TO A DO
           WRITELN(OP,'  X(',VA[J],') = ',X[VA[J]]:10:4);
        CLOSE(OP,LOCK);

     END;



PROCEDURE IMPRICAM;

    BEGIN

       DISIMPRIMAT;
       PAGE(OUTPUT);
       M:=I+1;
       WRITELN(OP);WRITELN(OP);
       WRITE(OP,'  CAMINO "P" S->S'' :[');
       REPEAT
          WRITE(OP,' I',CAM[M],', J',CAM[M-1],',');
          M:=M-2
       UNTIL M <= 3;
       WRITE(OP,' I',CAM[M],' ]');
       WRITELN(OP);WRITELN(OP);WRITELN(OP);WRITELN(OP);

       IMPRIMAT;
       WRITELN(OP);WRITELN(OP);
       CLOSE(OP,LOCK)

    END;
```

```
PROCEDURE MENU;
              VAR  WK :CHAR;

    BEGIN

       PAGE(OUTPUT);
       WRITELN;WRITELN;
       WRITELN('      MENU DE OPCIONES ');
       WRITELN;WRITELN;WRITELN;
       WRITELN(' 1)  ARCOS DE CAPACIDAD ILIMITADA');
       WRITELN;
       WRITELN(' 2)  RESTRICCION EN LA CAP.DE LOS ARCOS');
       WRITELN;
       WRITELN(' 3)  RESTRICCION EN LA DIVERGENCIA');
       WRITELN;WRITELN;WRITELN;WRITELN;
       WRITE('      ELIGE UNA OPCION -> ');
       REPEAT
          GETT(TK);
       UNTIL (TK IN ['1','2','3']);

    END;


PROCEDURE OPCION;

    BEGIN

       PAGE(OUTPUT);
       WRITELN;WRITELN;WRITELN;
       WRITELN('     DONDE DESEAS LAS RESPUESTAS ?');
       WRITELN;WRITELN;
       WRITELN('  P) EN LA PANTALLA ');
       WRITELN;
       WRITELN('  I) EN LA IMPRESORA');
       WRITELN;
       WRITELN('  D) EN EL DISCO ');
       WRITELN;WRITELN;WRITELN;
       WRITE('          ELIGE UNA OPCION => ');
       REPEAT
          GETT(QQ);
       UNTIL QQ IN ['P','I','D'];
       CASE QQ OF
       'P': PID:='#1:';
       'I': PID:='#6:';
       'D': BEGIN
               WRITELN;WRITELN;WRITELN;
               WRITELN('ESCRIBE EL NOMBRE DEL ARCHIVO');
               WRITELN;
               WRITE('          => ');
               READLN(PID);
               PID:=CONCAT('#5:',CONCAT(PID,'.TEXT'));
            END;
       END;

    END;
```

```
BEGIN

      PAGE(OUTPUT);

      INICIALIZA;

       VIEJARED;

      IMPRIMAT;


      MENU;
       CASE (TK) OF
         '1':BEGIN
                OPCION;
                FORMACAM;
                IMPRICAM;
                GETT(TK);
                PAGE(OUTPUT);
             END;
         '2':BEGIN
                OPCION;
                LEEMAXCAP;
                LEEMINFLUJO;
                PINTARED;
                REPEAT
                   FORMACAM;
                   SACAFLUJO;
                   PINTARED;
                   FOR J:=1 TO A DO
                      EP[VA[J]]:=0;
                UNTIL MAXIFLUX;
                IMPRIFLUJO;
                GETT(TK);
                PAGE(OUTPUT);
             END;
         '3':SETCHAIN('Ñ5:DIVE');
      END;
      SETCHAIN('Ñ5:EMP');

END.
```

```
(*$S+*)
      PROGRAM ELPEF;

      USES APPLESTUFF,CHAINSTUFF;

      TYPE          SUBIN=0..100;
                      ARR=ARRAY[0..50] OF INTEGER;
                     ARRE=ARRAY[0..100] OF INTEGER;
                    TIMAT=PACKED ARRAY[0..50,0..100] OF INTEGER;
                   ARREAL=ARRAY[0..100] OF REAL;



      VAR      M,A,K,I,KO,
               SPRI,ID,NUNO,
               J,NAC,NNU              :INTEGER;
               CAM,VCS,
               VN,EP                :ARR;
               E,VA,
               THETA,ITER             :ARRE;
               MATIN                  :TIMAT;
               COLOR                  :ARRAY[0..100] OF CHAR;
               DIVERYA,BU   :BOOLEAN;
               MATARCH                :FILE OF TIMAT;
               VEC1ARCH               :FILE OF ARRE;
               VEC2ARCH               :FILE OF ARR;
               CN,CA,CSP,CS           :SET OF SUBIN;
               CAP,X,CAPMIN,B,Y       :ARREAL;
               VF                     :ARRAY[0..50] OF REAL;
               ALFA,ALF1,DIF,KR       :REAL;
               TK,QQ                  :CHAR;
               OP                     :TEXT;
               PID                    :STRING;


   PROCEDURE GETT(VAR WK:CHAR);

   BEGIN

      WHILE NOT KEYPRESS DO;
         READ(KEYBOARD,WK)

   END;



PROCEDURE PINTARED;

      BEGIN

         FOR J:=1 TO A DO
            IF (CAP[VA[J]]-X[VA[J]]) <= (0.00001)
               THEN COLOR[VA[J]]:='R';

      END;



 PROCEDURE BUSCA(CONSTA,ARCO:INTEGER;VAR NODO:INTEGER);

         BEGIN

            FOR M:=1 TO NUNO DO
               IF MATIN[VN[M],ARCO]=CONSTA
                  THEN NODO:=M

         END;
```

```
PROCEDURE SACAM(K:INTEGER;VAR CAM:ARR);
                    VAR   WK:INTEGER;
        BEGIN
            I:=-1;
            WK:=K;
            REPEAT
                I:=I+1;
                CAM[I]:=ITER[WK];
                I:=I+1;
                CAM[I]:=THETA[ITER[WK]];
                EP[CAM[I]]:=EP[CAM[I]]+1;
                   BUSCA(1,CAM[I],ITER[WK-1]);
                WK:=WK-1;
            UNTIL  (ITER[WK] IN CS) OR (WK=0);
            CAM[I+1]:=ITER[WK];
        END;



PROCEDURE INICIALIZA;
          VAR K,M:INTEGER;

  BEGIN
     FOR K:=0 TO 100 DO
     BEGIN
        COLOR[K]:='B';
        THETA[K]:=0;
         ITER[K]:=0;
          CAP[K]:=0;
           VA[K]:=0;
            E[K]:=0;

          CAP[K]:=0.0;
       CAPMIN[K]:=0.0;
            X[K]:=0.0;
     END;

     FOR K:=0 TO 50 DO
     BEGIN
          CAM[K]:=0;
          VCS[K]:=0;
           VN[K]:=0;
           EP[K]:=0;
           VF[K]:=0.0;
            Y[K]:=0.0;
            B[K]:=0.0;
     END;

     A:=0;
     K:=0;
     NUNO:=0;
     NNU:=0;
     ALF1:=100000.0;
     VCS[1]:=1;
     CN:=[];
     CA:=[];
     CSP:=[];
     CS:=[1];
     DIVERYA:=FALSE;

  END;
```

```
PROCEDURE DISIMPRIMAT;

    BEGIN

        REWRITE(OP,PID);
        WRITELN(OP);WRITELN(OP);WRITELN(OP);WRITELN(OP);
        PAGE(OUTPUT);
        WRITE(OP,'        ');
        FOR K:=1 TO A DO
            WRITE(OP,' J',VA[K]:2);
        WRITELN(OP);WRITELN(OP);
        FOR M:=1 TO NUNO
        DO BEGIN
            WRITE(OP,' I',VN[M]:2,': ');
            FOR K:=1 TO A DO
                WRITE(OP,'  ',MATIN[VN[M],VA[K]]:2);
            WRITELN(OP);WRITELN(OP);
        END;

    END;



PROCEDURE IMPRIMAT;

            VAR  WK:CHAR;

    BEGIN

        WRITELN;WRITELN;WRITELN;WRITELN;
        WRITE('DESEAS VER LA MATRIZ DE INC.? <S/*>');
        GETT(WK);
        IF WK = 'S'
        THEN BEGIN
            PAGE(OUTPUT);
            WRITE('        ');
            FOR K:=1 TO A DO
                WRITE(' J',VA[K]:2);
            WRITELN;WRITELN;
            FOR M:=1 TO NUNO
            DO BEGIN
                WRITE(' I',VN[M]:2,': ');
                FOR K:=1 TO A DO
                    WRITE('  ',MATIN[VN[M],VA[K]]:2);
                WRITELN;WRITELN;
            END;
            GETT(TK);
        END;

    END;



PROCEDURE VIEJARED;

    BEGIN

        RESET(MATARCH,'N5:MATRIZ.DATOS');
        MATIN:=MATARCH^;
        GET(MATARCH);
        CLOSE(MATARCH);
        RESET(VEC1ARCH,'N5:VEC1.DATOS');
        RESET(VEC2ARCH,'N5:VEC2.DATOS');
```

```
        VA:=VEC1ARCH^;
        GET(VEC1ARCH);
        VN:=VEC2ARCH^;
        GET(VEC2ARCH);
        CLOSE(VEC1ARCH);
        CLOSE(VEC2ARCH);
        A:=VA[0];
        NUNO:=VN[0];
        SPRI:=MATIN[0,0];

    END;



PROCEDURE LEEMAXCAP;

    BEGIN

        PAGE(OUTPUT);
            PAGE(OUTPUT);
            WRITELN;   WRITELN;   WRITELN;
            WRITELN('   DAME LA CAPACIDAD MAXIMA DE :');
            WRITELN;WRITELN;
            FOR J:=1 TO A DO
            BEGIN
               WRITE(' ARCO J',VA[J],'  : ');
               READLN(CAP[VA[J]]);
               WRITELN;
            END;
    END;


PROCEDURE LEEMINFLUJO;

    BEGIN

        PAGE(OUTPUT);
        WRITELN;WRITELN;
        WRITELN(' EXISTEN RESTRICCIONES DE');
        WRITELN;
        WRITE('   CAPACIDAD MINIMA ?  <S/*> ');
        GETT(TK);
        IF TK = 'S' THEN
        BEGIN
            PAGE(OUTPUT);
            WRITELN;   WRITELN;   WRITELN;
            WRITELN('   DAME LA CAPACIDAD MINIMA DE :');
            WRITELN;WRITELN;
            FOR J:=1 TO A DO
            BEGIN
               REPEAT
               WRITE(' ARCO J',VA[J],'  : ');
               READLN(X[VA[J]]);
               IF ((CAP[VA[J]]-X[VA[J]]) < -0.0001)
                  THEN WRITE(CHR(7));
               UNTIL ((CAP[VA[J]]-X[VA[J]]) > -0.0001);
               WRITELN;
            END;
        END;

    END;
```

```
PROCEDURE LEEDIVER;

    BEGIN

       REPEAT
          KR:=0;
          PAGE(OUTPUT);
          WRITELN;   WRITELN;   WRITELN;
          WRITELN('   DAME LOS LIMITES DE DIVERGENCIA :');
          WRITELN;WRITELN;
          FOR I:=1 TO NUNO DO
          BEGIN
             WRITE(' NODO I',VN[I],'  : ');
             READLN(B[VN[I]]);
             WRITELN;
             KR:=KR+B[VN[I]];
          END;
          IF KR <> 0 THEN
          BEGIN
             WRITELN;WRITELN;
             WRITELN(' LA SUMA DE LAS DIVERGENCIAS ');
             WRITELN;
             WRITELN('  DEBE SER IGUAL A CERO ');
             WRITELN;WRITELN;WRITELN;
             WRITE('     INTENTE DE NUEVO ');
             GETT(TK);
          END;
       UNTIL KR = 0;

    END;



PROCEDURE SACAFLUJO ;

   BEGIN

      NAC:=0;
      FOR J:=1 TO A DO
      BEGIN
         ALFA:=EP[VA[J]]*(CAP[VA[J]]-X[VA[J]]);
         IF ALFA > 0
         THEN BEGIN
            NAC:=NAC+1;
            VF[NAC]:=ALFA
         END
      END;
      ALFA:=10000;
      FOR J:=1 TO NAC DO
      BEGIN
         IF VF[J] < ALFA
            THEN ALFA:=VF[J];
      END;
      IF ALF1 < ALFA
         THEN ALFA:=ALF1;
      FOR J:=1 TO A DO
         X[VA[J]]:=X[VA[J]]+EP[VA[J]]*ALFA;

   END;
```

```
PROCEDURE IMPRIFLUJO;

    BEGIN

       DISIMPRIMAT;
       WRITELN(OP);WRITELN(OP);WRITELN(OP);
       WRITELN(OP,'   >>>  EL FLUJO "X" EN LOS ARCOS ES :');
       WRITELN(OP);WRITELN(OP);WRITELN(OP);
       FOR J:=1 TO A DO
          WRITELN(OP,'   X(',VA[J],') = ',X[VA[J]]:10:4);
       CLOSE(OP,LOCK);

    END;

PROCEDURE SACAYE;

   BEGIN

      ALF1:=10000.0;
      CS:=[];
      CSP:=[];
      KO:=0;
      M:=0;
      FOR J:=1 TO NUNO DO
      BEGIN
         VCS[J]:=0;
         Y[J]:=0;
      END;
      FOR I:=1 TO NUNO DO
      BEGIN
         FOR J:=1 TO A DO
            Y[VN[I]]:=Y[VN[I]]+MATIN[VN[I],VA[J]]*X[VA[J]];
         DIF:=B[VN[I]]-Y[VN[I]];
         IF DIF > 0 THEN
         BEGIN
            CS:=CS+[VN[I]];
            KO:=KO+1;
            VCS[KO]:=VN[I];
            IF ALF1 > DIF
               THEN ALF1:=DIF;
         END;
         IF DIF < 0 THEN
         BEGIN
            CSP:=CSP+[VN[I]];
            IF ALF1 > -DIF
               THEN ALF1:=-DIF;
         END;
         IF DIF = 0
            THEN M:=M+1;
      END;
      IF M = NUNO THEN
      BEGIN
         DIVERYA:=TRUE;
         BU:=TRUE;
      END;

   END;
```

```
PROCEDURE HAZCAM;

    BEGIN

        J:=0;
        ID:=0;
        FOR J:=1 TO A DO
            E[VA[J]]:=0;
        FOR J:=1 TO A DO
        BEGIN
            FOR I:=1 TO KO DO
                E[VA[J]]:=E[VA[J]]+MATIN[VCS[I],VA[J]];
            IF (E[VA[J]] = 1) AND (ID = 0) AND (COLOR[VA[J]] = 'B')
                THEN ID:=VA[J];
        END;
        IF ID <> 0
        THEN
            BEGIN
                BUSCA(-1,ID,NNU);
                KO:=KO+1;
                VCS[KO]:=NNU;
                K:=K+1;
                ITER[K]:=NNU;
                THETA[ITER[K]]:=ID;
                IF NNU IN CSP THEN
                    BU:=TRUE;
            END
            ELSE
                IF NOT BU THEN
                    BEGIN
                        WRITELN;WRITELN;
                        WRITE('     NO HAY SOLUCION FACTIBLE ');
                        READLN;
                    END;

    END;



PROCEDURE OPCION;

    BEGIN

        PAGE(OUTPUT);
        WRITELN;WRITELN;WRITELN;
        WRITELN('     DONDE DESEAS LAS RESPUESTAS ?');
        WRITELN;WRITELN;
        WRITELN('  P) EN LA PANTALLA ');
        WRITELN;
        WRITELN('  I) EN LA IMPRESORA');
        WRITELN;
        WRITELN('  D) EN EL DISCO ');
        WRITELN;WRITELN;WRITELN;
        WRITE('          ELIGE UNA OPCION => ');
        REPEAT
            GETT(TK);
        UNTIL TK IN ['P','I','D'];
        CASE TK OF
        'P': PID:='#1:';
        'I': PID:='#6:';
        'D': BEGIN
                WRITELN;WRITELN;WRITELN;
                WRITELN('ESCRIBE EL NOMBRE DEL ARCHIVO');
                WRITELN;
                WRITE('             => ');
                READLN(PID);
                PID:=CONCAT('#5:',CONCAT(PID,'.TEXT'));
             END;
        END;

    END;
```

```
BEGIN

     PAGE(OUTPUT);
     INICIALIZA;
      VIEJARED;
     OPCION;
     PAGE(OUTPUT);
           BEGIN
              LEEMAXCAP;
              LEEMINFLUJO;
              PINTARED;
              LEEDIVER;
              REPEAT
              BEGIN
                 BU:=FALSE;
                 SACAYE;
                 K:=0;
                 NNU:=0;
                 REPEAT
                    HAZCAM;
                 UNTIL (K > A) OR BU;
                 IF NOT DIVERYA THEN
                 BEGIN
                    SACAM(K,CAM);
                    SACAFLUJO;
                    PINTARED;
                 END;
                 FOR J:=1 TO A DO
                    EP[VA[J]]:=0;
              END;
              UNTIL (DIVERYA) OR (K > A);
              IF DIVERYA
                 THEN IMPRIFLUJO
                 ELSE WRITE('     NO HAY SOLUCION FACTIBLE');
              READLN
           END;
      SETCHAIN('N5:EMP');
END.
```

MATRIZ DE INCIDENCIAS

	J 1	J 2	J 3	J 4	J 5	J 6	J 7	J 8	J 9	J10
I 1:	1	1	0	0	0	0	0	0	0	0
I 2:	-1	0	1	0	1	0	0	0	0	0
I 3:	0	-1	0	1	0	1	0	0	0	0
I 4:	0	0	-1	-1	0	0	1	1	0	0
I 5:	0	0	0	0	-1	-1	-1	0	1	0
I 6:	0	0	0	0	0	0	0	-1	0	1
I 7:	0	0	0	0	0	0	0	0	-1	-1

CAMINO "P" S->S' :[I1, J1, I2, J5, I5, J9, I7]

MATRIZ DE INCIDENCIAS

	J 1	J 2	J 3	J 4	J 5	J 6	J 7	J 8	J 9	J10
I 1:	1	1	0	0	0	0	0	0	0	0
I 2:	-1	0	1	0	1	0	0	0	0	0
I 3:	0	-1	0	1	0	1	0	0	0	0
I 4:	0	0	-1	-1	0	0	1	1	0	0
I 5:	0	0	0	0	-1	-1	-1	0	1	0
I 6:	0	0	0	0	0	0	0	-1	0	1
I 7:	0	0	0	0	0	0	0	0	-1	-1

>>> EL FLUJO "X" EN LOS ARCOS ES :

X(1) = 5.0000
X(2) = 1.0000
X(3) = 1.0000
X(4) = 1.0000
X(5) = 4.0000
X(6) = 0.0000
X(7) = 0.0000
X(8) = 2.0000
X(9) = 4.0000
X(10) = 2.0000

MATRIZ DE INCIDENCIAS

	J 1	J 2	J 3	J 4	J 5	J 6	J 7	J 8
I 1:	1	1	0	0	0	0	0	0
I 2:	-1	0	1	1	0	1	0	0
I 3:	0	-1	-1	0	1	0	1	0
I 4:	0	0	0	-1	-1	0	0	1
I 5:	0	0	0	0	0	-1	-1	-1

>>> EL FLUJO "X" EN LOS ARCOS ES :

X(1) =	2.0000	Y(1) =	5.0000
X(2) =	3.0000	Y(2) =	5.0000
X(3) =	2.0000	Y(3) =	1.0000
X(4) =	2.0000	Y(4) =	-4.0000
X(5) =	3.0000	Y(5) =	-7.0000
X(6) =	3.0000		
X(7) =	3.0000		
X(8) =	1.0000		

EL APENDICE `B'

Al ejecutarse el programa, aparece en la pantalla la opción de si se desea leer de Disco la última red que se usó en el programa o bien hacer una red nueva; de elegir ésta última opción, aparecen en la pantalla las letras "NODO EMISOR", "ARCO QUE SALE" y "NODO RECEPTOR", así como la palabra "NODO" seguida de un número, el cual indica el nodo para el que se introducirá información, primero el número del arco que sale de dicho nodo y posteriormente el nodo al cual llega el arco, el programa no permite la introducción de arcos repetidos, ni de arcos que salen de un nodo y llegan a él mismo, cuando se acaban de introducir todos los arcos que salen de el nodo I se oprime sólo la tecla "RETURN" y en la pantalla aparecerá el siguiente nodo al cual es necesario introducir información. Si un nodo no tiene ningún arco que sale, con sólo oprimir RETURN basta y aparecerá en su lugar otro nodo si es que existe. Así sucesivamente hasta que ya no exista ningún otro nodo en el conjunto de nodos mencionados en la columna "NODOS RECEPTORES".

De aquí se pasa a la pregunta "¿ DESEAS VER LA MATRIZ DE INCIDENCIAS ?" que al responder con "S" mostrará la matriz de incidencias sólo en la pantalla, cualquier otro caracter oprimido evitará dicha pantalla.

En seguida aparece la pregunta "¿ DESEAS CAMBIAR LA INFORMACION DE ALGUN ARCO ?" , si se responde con "S" preguntará "¿CUANTOS ARCOS QUIERES CAMBIAR ?" , a lo que se deberá responder con un número entero, menor que el número total de arcos.

Después, pedirá de uno en uno los nombres de los arcos que se desean cambiar, para después mencionar de uno en uno, dichos nombres, y colocando el cursor debajo de los

letreros "NODO EMISOR" y "NODO RECEPTOR" donde se deberá poner el nombre adecuado de los nodos para cada arco, - dando los nombres de los nodos deberá ser de los ya mencionados al formar la red.

Formada aquí la red, la acomoda en orden ascendente, tanto nodos como arcos con respecto a sus nombres y graba en disco la matriz, para pasar a la segunda parte del programa.

En la segunda parte del programa, se lee la matriz de - disco y aparece en la pantalla la opción para ver o no - dicha matriz de incidencias, para pasar al menú de opciones donde es necesario elegir un número de los ahí existentes:

"1" Para cuando se requiere encontrar un camino de capacidad ilimitada de la red actual,

"2" Para cuando se trata de un problema con restricciones de capacidad, y

"3" Cuando se desea resolver un problema con restricciones en la divergencia de los nodos.

Si se oprimió el "1", aparece una nueva opción en la pantalla, la opción de impresión, con la cual se elige el - lugar donde se desean tener los resultados, en la pantalla, en la impresora o en un archivo tipo texto, el cual se puede llamar a pantalla o mandar a la impresora en - cualquier momento.

Después de ésto el programa obtiene el camino e imprime, tanto el camino como la matriz de incidencias en el lugar elegido previamente.

Si lo que se oprimió fué el "2", respectivo al problema

con restricciones en la capacidad de los arcos se pregunta igual que en el anterior método el lugar a donde deben ir las respuestas, pero enseguida de ésto aparece un letrero, el cual indica que se deben introducir las capacidades máximas de flujo para cada arco, las cuales el usuario deberá introducir oprimiendo RETURN despues de escribir una cantidad REAL en la posición indicada.

Inmediatamente pasa a pregunar si existen restricciones de uso de los arcos, es decir si existe en al menos un arco una cantidad mínima de flujo que deba pasar a través del arco diferente de cero, de responder a la pregunta " ¿EXISTEN RESTRICCIONES DE CAPACIDAD MINIMA? " - con un caracter diferente de "S" las tomará iguales a cero todas, de responder "S" preguntará de una en una - para que el usuario las introduzca de igual manera que las capacidades máximas.

Habiendo obtenido los intervalos de capacidad para todos los arcos el programa se ejecuta e imprime la matriz de incidencias de la red estudiada, así como los flujos correspondientes para cada arco, tales que maximicen el flujo total de la red.

Si en el menú de opciones se oprimió el "3", correspondiente al problema con restricciones de capacidad en los arcos además de divergencia en los nodos, se llama desde este lugar, la segunda parte de esta sección donde están incluídos los métodos.

Al llamarse el subprograma correspondiente a la solución del tercer tipo de problemas, el de la divergencia, se lee el disco la información necesaria para resolver el - problema, y empieza apareciendo en pantalla el nodo para que el usuario introduzca los valores de capacidad máxima, que es igual al del método anterior, la capacidad - mínima para cada arco, hasta aquí es igual, pero después

de ésto se pide al usuario que dé los valores de diver gencia para cada nodo, a los cuales el algoritmo tratará de llegar, la suma de las divergencias introducidas debe ser igual a cero, de otra manera no existiría un - flujo tal que cumpliera con dicha restricción (el programa no admite los valores si la suma de estos no es igual a cero).

Después de haber leído las divergencias correspondientes a cada arco el programa resuelve el problema e imprime la matriz de incidencias, el flujo en cada arco y la divergencia obtenida que debe ser igual a la requerida.

LA BIBLIOGRAFIA

* Kaufmann A, Métodos y Modelos de La Investigación de operaciones, Cía Editorial Continental, S. A., México, 1980.

* Hillier, Frederick, Gerald Lieberman, Operation Research, Holden Day, Inc., San Francisco, Cal. 1974.

* Taha Hamdy A, Operation Research, Collier - - McMillan, New York, 1976.

* Sivazlian B D, Optimization Thechniques In - Operation Research, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1975.

* Phillips, Don T, A. Ravindran, J. Solberg, Operation Research Principles on Practice, John Wiley & Sons. New York, 1976.

* Wagner H. M., Principles of Operations Research Prentice - Hall Inc., Englewood Cliffs, New - Jersey, 1969.