24
2 ejc.

VNIVERSIDAD NACIONAL
AVFNOMA DE
MEXICO

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
"A C A T L A N"

TEORIA DE REDES
Y
SUS APLICACIONES

TESINA
QUE PARA OBTENER EL TITULO DE:

LICENCIADO EN MATEMATICAS APLICADAS Y COMPUTACION

P R E S E N T A

MAYRA OLGUIN ROSAS

ACATLAN, EDO. DE MEXICO 1994

TESIS CON
FALLA DE ORIGEN

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

DERECHOS RESERVADOS © PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis está protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES "ACATLAN"

DIVISION DE MATEMATICAS E INGENIERIA
PROGRAMA DE ACTUARIA Y M.A.C.

SRITA. MAYRA OLGUIN ROSAS
Alumna de la carrera de M.A.C.
P r e s e n t e .

De acuerdo a su solicitud presentada con fecha 18 de octubre de 1993, me complace notificarle que esta Jefatura tuvo a bien asignarle el siguiente tema de tesina: - "TEORIA DE REDES Y SUS APLICACIONES", el cual se desarrollará como sigue:

INTRODUCCION
CAP. I Conceptos Básicos de Redes.
CAP. II El problema general de Redes.
CAP. III Casos Especiales.
CONCLUSIONES
ANEXOS
BIBLIOGRAFIA

Asimismo fué designado como Asesor de Tesina él - Fis.Mat. Jorge Luis Suárez Madariaga, profesor de esta - Escuela.

Ruego a usted tomar nota que en cumplimiento de - lo especificado en la Ley de Profesiones, deberá presentar servicio social durante un tiempo mínimo de seis meses como requisito básico para sustentar examen profesional, así como de la disposición de la Coordinación de la Administración Escolar en el sentido de que se imprima - en lugar visible de los ejemplares de la tesina el título del trabajo realizado. Esta comunicación deberá imprimirse en el interior de la tesina.

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPIRITU"
Acatlán, Edo. Méx. abril 15 de 1994.

ACT. LAURA MA. RIVERA CACERES
Jefe del Programa de Actuaria y M.A.C.

JEFATURA DEL PROGRAMA DE ACTUARIA Y APLICADAS

L.M.R.B.'c.g.

A Dios por permitirme llegar a este momento;

a mis padres;

muy especialmente a **Manuel**;

y, en general, a todas las personas de quienes he aprendido algo.

INDICE

Pág.

Capítulo II

Capítulo III

INTRODUCCION

Desde hace mucho tiempo, el análisis de redes ha desempeñado un papel muy importante en la ingeniería. En las últimas décadas se han realizado aplicaciones importantes del análisis de redes en la teoría de la información, la cibernética, el estudio de sistemas de transporte y la planeación y control de proyectos de investigación y desarrollo. Otras áreas de aplicación incluyen estructuras de grupos sociales, sistemas de comunicación, programas de producción, estructuras de enlaces químicos, estructuras de lenguaje, etc.

La representación en redes proporciona una ayuda conceptual poderosa para visualizar las relaciones entre las componentes de sistemas complicados que con frecuencia se tienen que analizar en la investigación de operaciones. Como resultado, ciertos aspectos del análisis de redes (conocidos comúnmente como teoría de flujo en redes) se han convertido en una herramienta de gran utilidad para el investigador de operaciones.

Existen varios tipos de problemas en los sistemas de transporte, entre ellos:

i) Escoger un conjunto de conexiones que proporcione una ruta entre dos puntos cualesquiera de una red de manera que se minimice la longitud total de estas conexiones.

ii) Asignación de flujos con el fin de maximizar el flujo a través de una red que conecta una fuente u orígen y un destino.

iii) Encontrar los caminos para trasladar mercancía desde varios orígenes a diferentes destinos, de manera que se minimice el costo de transporte.

iv) La planeación y control de proyectos se ha atacado por medio de las técnicas de redes, en especial el PERT (Program Evaluation and Review Technique o técnica de Evaluación y Revisión de Programas) y el CPM (Critical Path Method o Método de la Ruta Crítica).

Todos estos problemas son estructuras especiales de la programación lineal que, aún cuando pueden resolverse por el método simplex, sus propiedades especiales ofrecen un procedimiento de solución más sencillo.

El presente trabajo trata sobre la parte de la programación lineal denominada redes de flujo, en particular sobre el algoritmo desarrollado por L. R. Ford & D. R. Fulkerson, con el nombre del algoritmo Out-Of-Kilter (o desviaciones) para resolver el problema general de redes con flujos restringidos y costo mínimo. La importancia práctica de este problema se encuentra en el hecho de que una buena cantidad de la literatura de la programación lineal esta dedicada a su estudio y una variedad de aplicaciones estan en su dominio. Los problemas de transporte que involucran cientos de restricciones y miles de variables pueden ser resueltos, mientras que para resolver un problema de programación lineal de estas dimensiones por el método simplex, es actualmente una tarea casi imposible. Cabe mencionar también que una variedad de problemas de la programación lineal (Transporte, flujo máximo, ruta más corta, asignación, distribución etc.), pueden ser formulados en el marco de redes y resueltos con el algoritmo Out-Of-Kilter.

El objetivo fundamental de esta tesina es la creación de un material de apoyo didáctico accesible a los estudiantes en particular de las carreras de Matemáticas Aplicadas y Computación, Actuaría e Ingeniería Civil; y en general, cualquier estudio profesional que lo requiera.

Los conocimientos básicos para la comprensión de este trabajo son, elementos de algebra matricial, teoría de redes, e investigación de operaciones.

Esta tesina no pretende, en modo alguno, ser un tratado exhaustivo sobre Teoría de Redes; más bien se desea motivar al lector hacia un conocimiento más profundo de la materia y, sobre todo, hacia una implementación de este tipo de modelos de redes en su campo específico de trabajo. Por esta razón se ha buscado un enfoque ante todo práctico, utilizando como apoyo una computadora.

El trabajo se desarrolla como sigue: En el capítulo I se presentan algunas definiciones básicas de redes, con el propósito de unificar conceptos. En el capítulo II se introduce el problema de redes con flujos restringidos y costo mínimo y se muestra la obtención del problema dual. A continuación se establecen las condiciones de optimalidad y algunas propiedades importantes de la matriz de coeficientes, además se presenta la estrategia del algoritmo Out-Of-Kilter para que un arco no conformable pase a ser conformable, se muestra también la no factibilidad del problema cuando $\theta = \infty$ y se prueba la convergencia del algoritmo. El capítulo III se concentra en los problemas y aplicaciones, resolviéndose un problema de ruta crítica utilizando el programa CPM. El trabajo contiene también dos secciones de anexos, en el Anexo A se presentan los diagramas de flujos de algoritmos de solución, específicos para problemas de redes. En el Anexo B se presenta el Manual de Usuario del programa CPM que resuelve problemas de redes utilizando la ruta crítica.

CAPITULO I

A N T E C E D E N T E S

En 1975 la Academia Real Sueca de Ciencias otorgó el Premio Nobel en Ciencias Económicas, compartido, al Profesor Leonid Kantorovich de la URSS y al Profesor Rjalling C. Koopmans de los Estados Unidos, por sus contribuciones a la teoría de la asignación óptima de recursos. Aunque estos distinguidos profesores investigaron una gran variedad de problemas de optimización, es interesante notar que ambos están asociados con algunos de los primeros papeles que describen los problemas de flujo en redes, tal como los conocemos ahora. En 1939, en un papel expositor, Kantorovich discutió una clase especial de modelos de optimización, junto con ejemplos específicos. La idea subyacente de cada ejemplo fue la de alcanzar la producción más alta posible, sobre la base de la utilización óptima de los recursos existentes. Uno de estos ejemplos involucró la distribución de flujos de carga entre las diferentes rutas de una red de carreteras, con el objeto de minimizar el uso de combustible.

Discusiones posteriores del flujo de carga en la URSS fueron dados por Kantorovich y Gavurin. Estos trabajos no fueron conocidos en Occidente hasta finales de la década de los cincuentas. Mientras tanto, en los Estados Unidos, Frank L. Mitchcook presentó lo que ahora es conocido como la formulación estándar del problema de transporte. Trabajando independientemente, el profesor Koopmans formuló el mismo problema en conexión con su trabajo con la Oficina de Regulación de Embarques. Debido a estos trabajos, el problema de transporte, frecuentemente es referido como el problema de

transporte de Hitchcock o el problema de transporte de Hitchcoock-Koopmans.

Tanto Hitchcock en 1941, como Koopmans en 1946, presentaron métodos computacionales del tipo primal simplex para la solución del problema de transporte. Hitchcock mostró que una solución óptima debería ser un punto extremo de solución y mostró como se podían construir iterativamente mejores soluciones de puntos extremos, tomando en cuenta soluciones óptimas alternas y degeneración. Koopmans lo resolvió desarrollando multiplicadores simplex, o "nodos potenciales" y el criterio de optimalidad, mostrando que el punto extremo es equivalente a un árbol. Dantzig mostró en 1951 que el modelo de transporte puede ser resuelto por su algoritmo simplex, y también desarrolló una variante especial del algoritmo simplex para el problema de transporte.

En 1956 Alex Orden propuso una generalización del modelo de transporte, en la cual se permitirían puntos de transbordo. Esta formulación es conocida hoy en día como el Problema de Transbordo sin Capacidades. Aproximadamente al mismo tiempo, ambos, el problema de flujo máximo en redes y el problema de costo mínimo de flujo en redes, fueron formulados e investigados por el famoso equipo de Lester Ford y Delbert Fulkerson.

Desde 1950 hasta 1965, se dirigió mucha actividad hacia el desarrollo de algoritmos para modelos lineales de flujo en redes. Los algoritmos desarrollados pueden ser clasificados en dos tipos, como sigue: (1) especializaciones del método simplex primal y (2) métodos primales-duales.

Las implementaciones contemporáneas del algoritmo primal simplex de redes están basados en la representación compacta de la base,

la determinación del arco de partida sin prueba y error y en técnicas eficientes para actualizar los multiplicadores simplex en cada pivote. Algunos de estos desarrollos fueron sugeridos en 1960 por Glicksman. Las especializaciones del método primal simplex empezaron con el trabajo de Dantzig y culminaron con el papel de Ellis Johnson y Eselson, que describieron un problema de transporte con pocas fuentes y muchos destinos; sus estructuras de datos pueden ser vistas como el almacenamiento de la base en forma de árbol y la utilización de esta estructura para encontrar eficientemente el arco de salida. E. Johnson describió un esquema de triple etiqueta que representa la base con un árbol y permite que los multiplicadores simplex sean actualizados eficientemente. Johnson describe su trabajo como una modificación del propuesto por Scoins.

Los métodos primales-duales se originaron con el *algoritmo húngaro* de Harold Kuhn para el problema de asignación y culminaron con los del *algoritmo out-of-kilter* de Delbert Fulkerson, el primal-dual de Ford y Fulkerson, el dual de Balas y Hammer, el de ruta de Busacker y Gowen, el de ciclo negativo de Kley y el de escalamiento de Edmonds y Karp. También se han desarrollado algoritmos especiales para los problemas de asignación, de la ruta más corta y de flujo máximo.

La mayor parte de la actividad desarrollada a partir de esos papeles clásicos ha involucrado la implementación eficiente de estas técnicas básicas y la extensión de esta tecnología a los problemas lineales de redes que involucran ganancias, a los problemas lineales de asignación de bienes múltiples, a los problemas lineales de redes con restricciones generales adicionales, y a los problemas de redes que tienen funciones de costo no lineales y convexas.

Los propósitos de Glicksman, L. Johnson y Eselson, Scoins y E. Johnson para los algoritmos primales no fueron inmediatamente alcanzados, el código más ampliamente conocido de esa década es una implementación del algoritmo out-of-kilter hecha por Clasen. Más tarde, e independientemente, Mc Bride y Graves expecializaron su trabajo sobre la factorización de programas lineales de problemas de transbordo. Aunque su desarrollo es un poco diferente, la especialización en redes de sus estructuras de datos es similar en muchos aspectos a las estructuras de datos que emergieron del punto de vista de teoría de gráficas de las redes.

Mulvey ha desarrollado un código primal eficiente a gran escala, Harris desarrolló un algoritmo primal para el problema de transporte. Langley, Kennington y Shetty también desarrollaron un código primal.

Un aspecto significante del trabajo contemporáneo en investigación de redes ha sido la prueba computacional de diferentes algoritmos para problemas estándares muy grandes. Uno de los tópicos principales ha sido la confrontación de los algoritmos primales contra el algoritmo de *out-of-kilter*. Los experimentos hechos en los cincuenta y a principio de los sesenta, convencieron a los investigadores de que el algoritmo *out-of-kilter* era superior, especialmente para problemas de transbordo.

1.1 LAS REDES Y SU UTILIDAD

El hombre, en su deseo de conocer y manejar su entorno, ha buscado representarlo en forma simplificada. Para ello, se ha valido de diferentes metodologías de solución, una de ellas es el uso de redes (representaciones gráficas). Este proceso no sólo le ha ayudado a comprenderla más, sino también a resolver problemas complejos que pretenden imitar el medio en que se encuentran.

En forma general, en un proyecto se construyen las redes por tres razones:

1. Planear
2. Programar
3. Controlar

En la etapa de planeación, una red muestra las interrelaciones que se dan en un sistema, esto ayuda a determinar el conjunto de actividades que se requieren incluir al realizar la asignación de los recursos de la manera más eficiente sobre las actividades que lo requieran. El uso de redes en esta fase es bastante simple y económico.

En cuanto a la programación de actividades, lo anterior es uno de tantos aspectos que hacen atractivo el uso de redes.

Una vez que un proyecto se encuentra "caminando", es muy fácil perder el control del mismo, las redes pueden ayudar a evitarlo al proporcionar al usuario el grado de avance en un período dado.

Lo anterior de ninguna manera asegura una efectividad del 100% en esta labor, pero permitirá detectar fuentes problemáticas que ameriten de mayor atención y cuidado.

1.2 MODELOS DE REDES DE FLUJO

Como se ilustra en la Figura 1.1, una red es una colección de nodos y arcos. Esta representación es útil para modelar o formular una gran variedad de situaciones físicas y conceptuales. Por ejemplo, sistemas de carreteras para el tráfico de vehículos a menudo son formulados como modelos de redes. En éstos, los nodos representan a los centros de tráfico y los arcos son las vías que unen pares de centros. En forma similar, sistemas de ferrocarriles, sistemas de aerolíneas y, en general, todos los problemas de planeación de transporte se pueden representar apropiadamente como redes. Otras situaciones, tales como sistemas de inventarios, sistemas de producción-distribución, sistemas de comunicación, sistemas de ríos y sistemas de tubería, han sido representados como redes. A los modelos de tales situaciones se les llama **modelos de redes de flujo.**

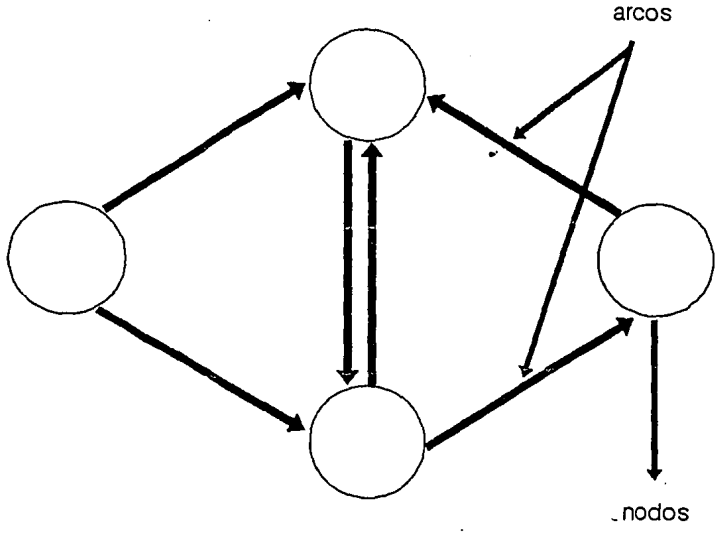


Figura 1.1 Red

Se considerarán modelos de redes de flujo en los que el flujo en cada arco se maneja entre cierto rango, y el objetivo es seleccionar flujos en los arcos que optimicen alguna medida de efectividad.

Para ilustrar esto, suponga que la red de la Figura 1.2 define un modelo de redes de flujo. Cada arco en la red lleva un flujo dirigido según la fecha y tiene tres parámetros asignados: una cota inferior, que es la mínima cantidad que puede fluir en el arco; una cota superior, que es la máxima cantidad de flujo que puede llevar el arco; y un costo para cada unidad de flujo que pase a través del arco.

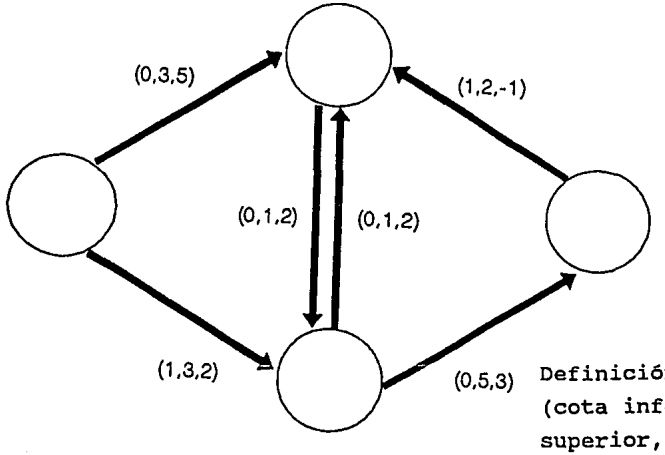


Figura 1.2 Red de Flujo con Solución

Si el parámetro de cota inferior no está presente en el arco se asume que su cota inferior es cero y si está presente, y por alguna razón se debe de reducir a cero, no hay problema, en la sección 1.6 se proporciona una transformación sencilla para remover cotas inferiores.

Claramente, los flujos en los arcos son las variables de decisión en el problema de optimización, el cual, consiste en seleccionar flujos en los arcos que satisfaciendo las condiciones restrictivas establecidas, optimicen el costo total del flujo.

1.3 RELACION ENTRE LOS PROBLEMAS DE PROGRAMACION DE REDES DE FLUJO

El problema de optimizar alguna función objetivo sujeta a restricciones es denominado **programación matemática.** Debido a que todos los problemas considerados aquí se definen como una red que lleva flujo, se usa el término **programación de redes de flujo.** En la Figura 1.2 se ilustra un ejemplo específico de un problema de flujo a costo mínimo, lineal y sin ganancias. Se dice que el flujo es sin ganancia cuando el flujo se conserva al pasar a través del arco y, con ganancia, cuando el flujo se incrementa o disminuye por algún factor al pasar por el arco. A dicho factor se le llama ganancia. Ahora bien, en la Figura 1.3 se muestra la relación entre los diferentes problemas de programación de redes de flujo. El punto central de esta figura es el problema de flujo a costo mínimo, lineal y sin ganancia. A la izquierda se listan los problemas menos generales en el sentido de que son especializaciones del problema central. Los problemas listados a la derecha son más generales o bien, los problemas básicos son un caso particular de éstos. También se muestra el problema general de programación lineal para indicar

su relación con los problemas de programación de redes. Se han identificado algoritmos que resuelven cada uno de los problemas de la Figura 1.3, encontrándose que los algoritmos para los problemas menos generales son más eficientes; en tiempo de proceso y espacio (memoria), que para los problemas más generales.

Resulta que los algoritmos definidos para los problemas más generales pueden resolver los problemas en su forma menos general, sin embargo, el inverso rara vez es cierto.

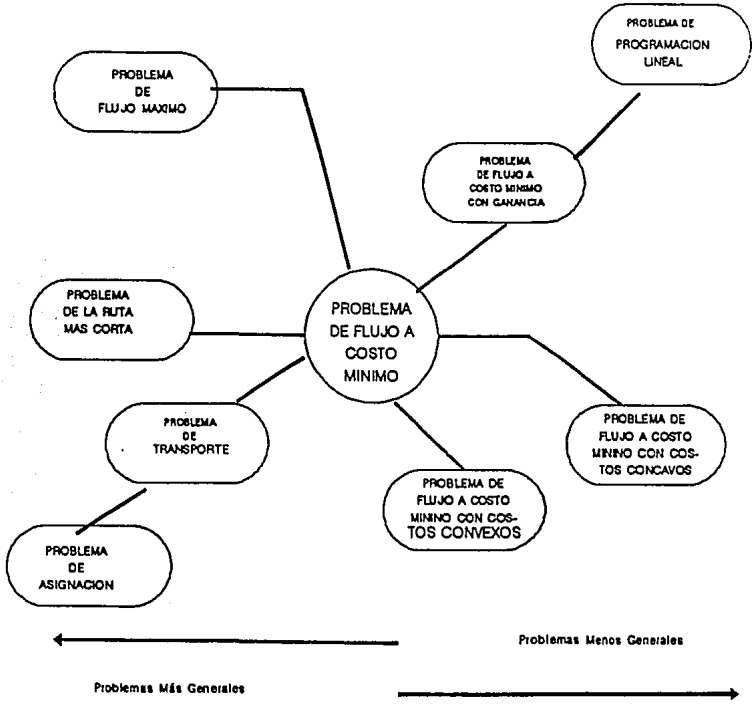


Figura 1.3
Relación entre los Problemas de Programación de Redes de Flujo

1.4 NOTACION EN REDES

1.4.1 Estructura

La estructura de un modelo de redes está definida por un conjunto finito de nodos (puntos o vértices) y arcos (líneas,aristas o ramas). Un nodo i es un elemento de la lista de nodos N={1,2,3,...,i,...,m}, y un arco queda definido por un par ordenado de nodos (i,j) o como un elemento k de la lista de arcos A={(i,j),(k,l),...,(s,t)}. El arco k o k(i,j) se origina en el nodo i, nodo origen, y termina en el nodo j, nodo terminal o destino. Para definir todas las conexiones presentes en la red, se identifican las listas origen y destino

$$O = [O_1, O_2, \ldots, O_n]$$
$$T = [t_1, t_2, \ldots, t_n]$$

donde O_k y t_k son los nodos origen y destino, respectivamente, del arco k. La colección de nodos y arcos es una red dirigida (o simplemente red) y se identifica por D = { N,A }, donde los valores de n y a y los conjuntos O y T son suficientes para definir todos los arcos de la red.

1.4.2 Flujo y Costo

Flujo

El flujo en el arco usualmente representa alguna cantidad física tal como el flujo de un fluído, flujo de personas, artículos, materiales o flujo de dinero, y se denota por X_{ij} para el flujo entre los nodo i y j. Una característica principal del flujo en la red es que se conserva en cada nodo.

Costo

Un costo se asocia con el flujo en el arco. El costo en el arco es una función del flujo, $c(\varphi)$. Nuestro objetivo será optimizar el costo de la red, el cual, es la suma de los costos de todos los arcos.

$$C(\varphi) = \sum_{k=1}^{m} \varphi\ (U_{ij})$$

1.4.3 Representación Gráfica Del Modelo De Redes

En la representación gráfica los nodos se simbolizan por círculos con el índice de nodo adentro. Los nodos se enumeran consecutivamente comenzando con el número uno. Un arco se dibuja como un segmento de línea dirigido que conecta sus nodos origen y terminal, y tanto su índice como sus parámetros se muestran adyacentes al arco, estos últimos entre paréntesis. Para identificar los parámetros, en cada representación gráfica se proporciona la definición de los mismos (Figura 1.2).

1.5 TRANSFORMACION DE UN ARCO NO DIRIGIDO

Un arco no dirigido (i,j) con capacidad mínima igual a cero y capacidad máxima C_{ij} se puede manejar reemplazándolo por un par de arcos opuestos (i,j) y (j,i), ambos con capacidad mínima igual a cero y capacidad máxima igual a C_{ij}

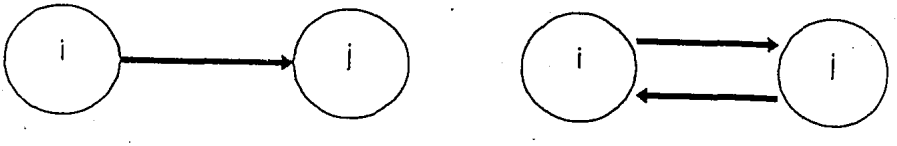


Figura 1.5
Transformación de un Arco No Dirigido

Otra manera de manejar este caso es asignando una dirección arbitraria a la línea y tratarla como un arco con C=-c y c= c. Luego, si el flujo en el arco es positivo, este va en la dirección supuesta; mientras que si el flujo es negativo, fluye en la dirección contraria a la supuesta con el signo cambiado, es decir, positivo. Por supuesto, lo anterior procede si el algoritmo de resolución usado permite flujos negativos

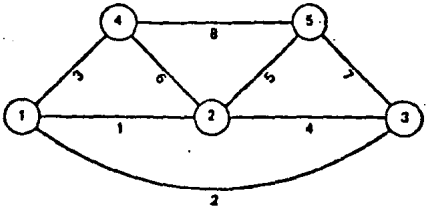
1.6 TERMINOLOGIA SOBRE REDES

Para continuar con el análisis de redes, es necesario revizar terminología básica de las mismas, para su utilización, con lo cual se pretende unificar definiciones utilizadas, ya que es común encontrar en la literatura al respecto diferentes definiciones para el mismo término.

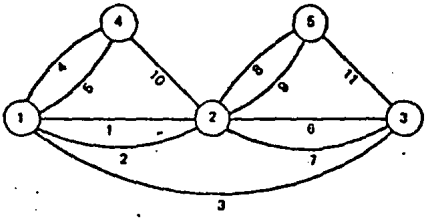
1.6.1 Gráfica, Multigráfica y Gráfica Dirigida

Una gráfica, denotada por G =[N,A], consiste de un conjunto finito de nodos N y un conjunto A formado por pares ordenados de nodos de N. Cada elemento de A es una línea o arista y se representa por (i,j) donde i y j son elementos de N. Una vuelta ("loop") se forma con una línea que une a un nodo consigo mismo. En una gráfica no se permiten vueltas ni líneas múltiples, es decir, dos o más líneas que unen al mismo par de nodos. En cambio, en una multigráfica sí se permiten líneas múltiples pero no vueltas. Finalmente, una gráfica es dirigida si todas sus líneas son arcos o pares ordenados de nodos (i,j), cada arco tiene una orientación o dirección específica. También puede haber gráficas no dirigidas en que todos los arcos no tengan dirección específica, también gráficas mixtas en las cuales algunos arcos son dirigidos otros no. Podemos dibujar este tipo de gráficas en la misma forma, solo que omitiendo las flechas en los arcos que no tienen alguna orientación. En la figura 1.6 se ilustran ejemplos de estos términos.

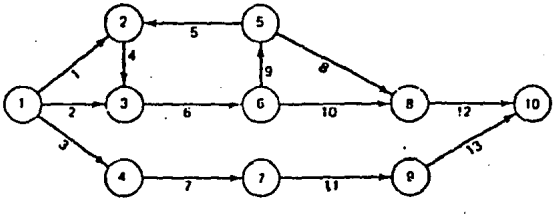
(a)



(b)



(c)

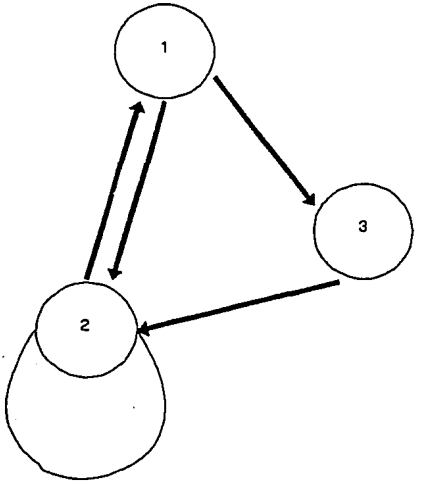


(d)

Figura 1.6
Ejemplos de (a) Vuelta ("loop") (b) Gráfica
(c) Multigráfica (d) Gráfica Dirigida

Matriz de Incidencia Nodos-Nodos:

Consiste en una matriz Q de orden mxm, donde m es el número de nodos. En la matriz $Q = (q_{ij})$ se tiene que $q_{ij} = 1$, si existe un arco que va del nodo i al nodo j; de otra manera $q_{ij} = 0$. La matriz de incidencia nodos-nodos asociada a la gráfica dirigida de la siguiente gráfica es:



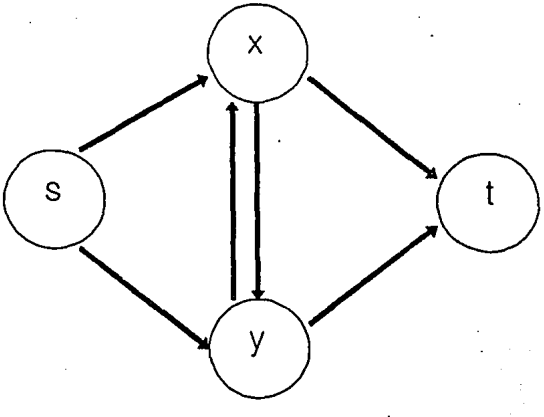
$$Q = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} \end{array}$$

Claramente, toda la información acerca de la estructura de la gráfica está caracterizada en su matriz de incidencia nodos-nodos

Matriz de Incidencia Nodos-arcos:

Consiste en una matriz P de orden mxn, donde m es el número de nodos y n el número de arcos, los cuales han sido previamente numerados. En la matriz $P = (P_{ij})$ se tiene que $P_{ij} = 1$ si el nodo i parte del arco con número j. Asimismo, $P_{ij} = -1$ si al

nodo i llega el arco con número j. En otros casos, $P_{ij} = 0$. Es conveniente hacer notar que esta característica es solo aplicable cuando no hay arcos de la forma (i,i) en la gráfica dirigida. La matriz de incidencia nodos-arcos asociada a la gráfica dirigida de la siguiente figura es P.



$$P = \begin{array}{c} \\ s \\ x \\ y \\ t \end{array} \begin{array}{c} \begin{array}{cccccc} (s,x) & (s,y) & (x,y) & (y,x) & (x,t) & (y,t) \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & -1 & 1 & 0 \\ 0 & -1 & -1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix} \end{array}$$

1.6.2 Cadena y Circuito

Una cadena del nodo i al nodo j en G, es una sucesión de nodos y arcos 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 $t=1,\ldots,r-1$. De esta manera, cada arco en la cadena tiene la misma dirección. Un circuito es una cadena en que el nodo inicial es igual al nodo final. Ver firgura 1.7.

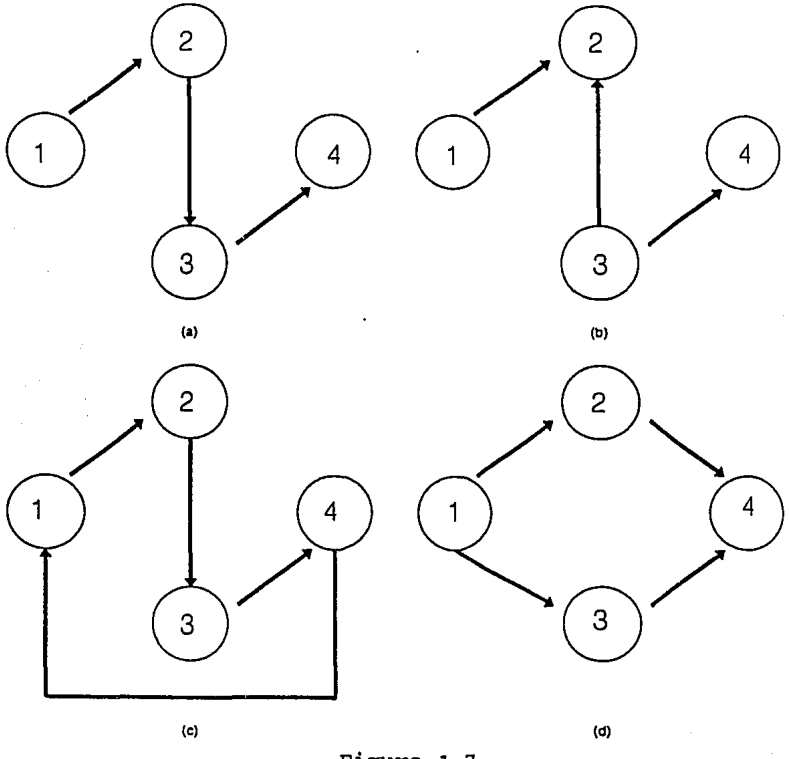


Figura 1.7
Gráficas Dirigidas (a) Cadena (b) Trayectoria
(c) Circuito (d) Ciclo

1.6.3 Trayectorias y Ciclos

Si en la definición de cadena se permite que cada arco tenga la forma $a_t=(i_t,i_{t+1})$ o bien $a_t=(i_{t+1},i_t)$ donde $t=1,\ldots,r-1$, se obtiene una trayectoria del nodo i al nodo j. Una trayectoria de i a i que contiene al menos dos arcos es un ciclo (o anillo) y se dice que contiene a i. Ver figura 1.7. De las definiciones anteriores se deduce que toda cadena es una trayectoria y que todo circuito es un ciclo, pero no recíprocamente.

Gráfica Conectada

G es una gráfica conectada (o red conexa) si existe una trayectoria entre cada par de nodos. En caso contrario G es desconectada (o red inconexa).

1.6.4 Red Simple, Reducida, Conexa, Inconexa y Subred Dirigida

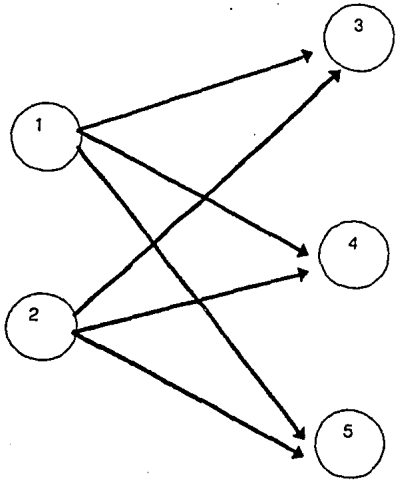
Una red dirigida es una multigráfica dirigida D=[N,M] o bien, es una gráfica en que se permiten líneas múltiples.

Red simple:

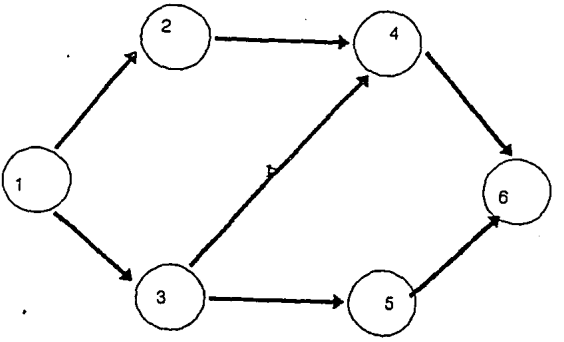
Una red G = (N, A), es simple si el conjunto de nodos N puede dividirse en dos subconjuntos, N_1 y N_2, tales que si $(i,j)\in A$ entonces, $i \in N_1$ y $j \in N_2$.

Red reducida:

Una red G = (N, A), es reducida si tiene un solo depósito s y un solo destino r, y no existen arcos de la forma (i, s) o (r, j) donde i, j ∈ N.



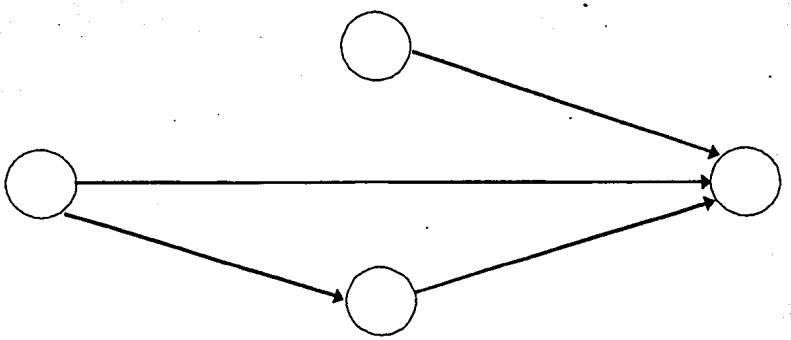
RED SIMPLE



RED REDUCIDA

Red conexa:

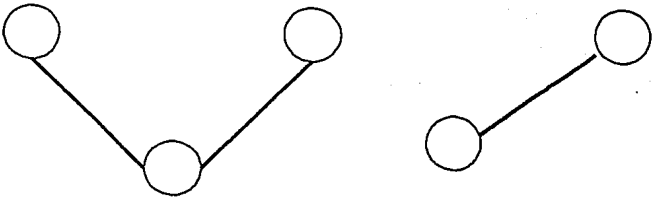
Es aquella en donde existe por lo menos una cadena que conecta a cada nodo con el resto de los nodos de la red.



RED CONEXA

Red inconexa:

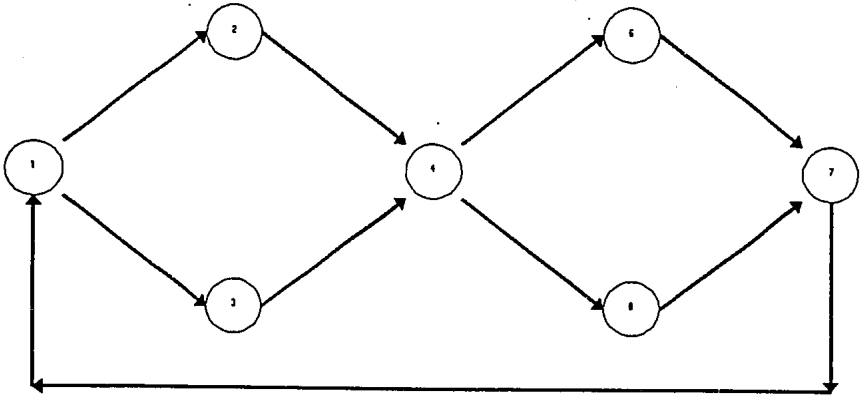
Es aquella que no está conectada.



RED INCONEXA

1.6.4.1 Red Circulatoria

Una red esta en forma circulatoria si todos sus nodos son de traspaso.



RED CIRCULATORIA

Mencionaremos que toda red simple puede fácilmente transformarse en red circulatoria.

Para convertir una red simple que tiene varios nodos fuente (depósito), por ejemplo (s_1, s_2) y varios nodos destino por ejemplo (t_1, t_2, t_3) como la que se muestra en la figura, a una red circulatoria, primero

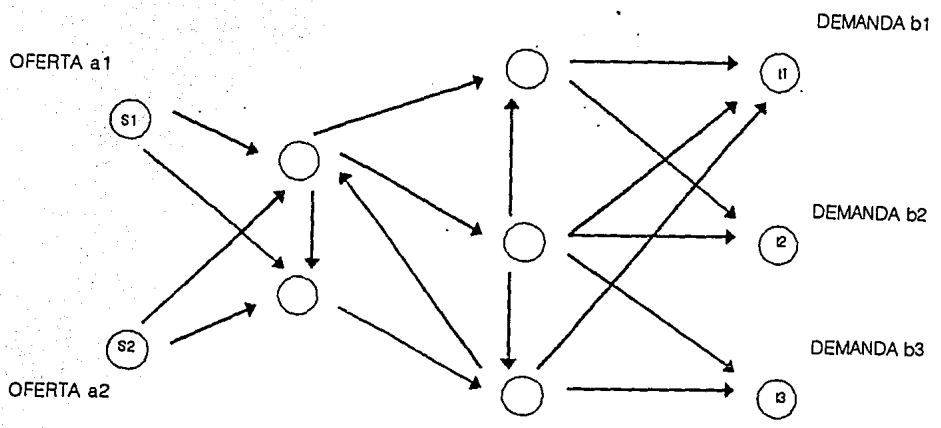
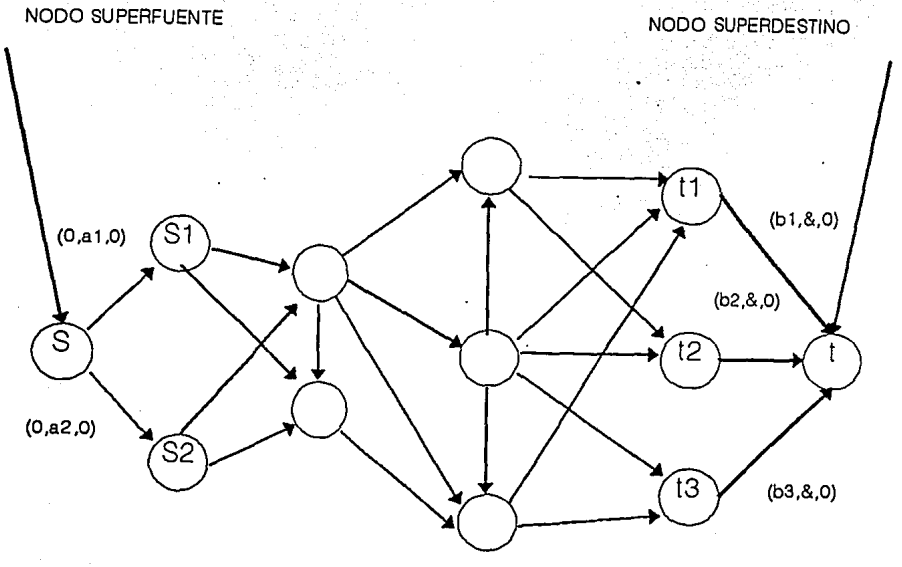


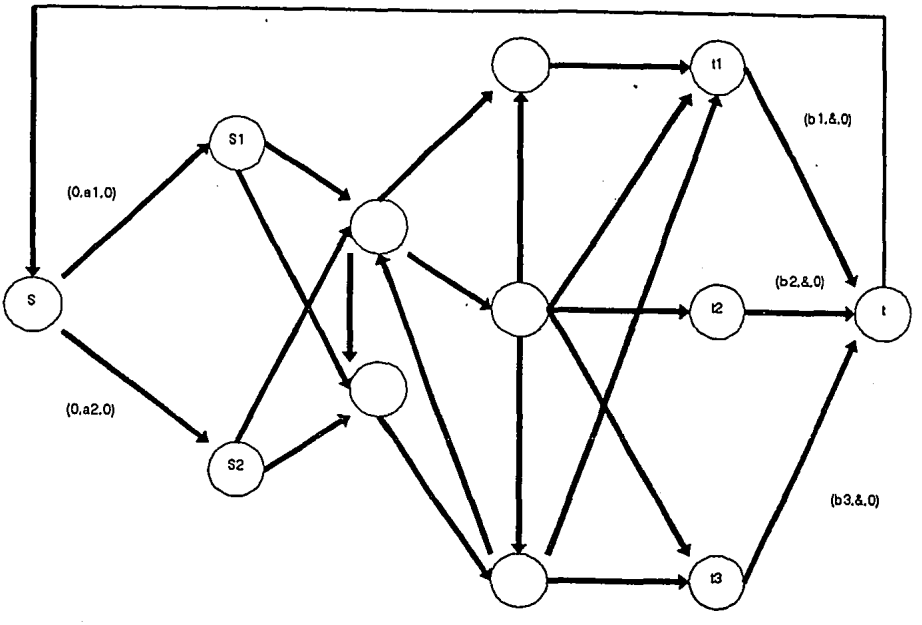
Figura 1.8 RED CON FUENTES Y DESTINOS MULTIPLES

hay que construir una red equivalente con un nodo superfuente y un nodo superdestino. Esto se logra añadiendo un nodo artificial s unido a los nodos fuente originales por arcos (s,s_1), (s,s_2), etc., con costo cero, capacidad mínima cero y capacidad máxima igual a la oferta del nodo fuente original.

Además, se construye un nodo destino artifical t, unido a los nodos destino originales por arcos (t_1, t), (t_2, t), (t_3, t), etc., con costo cero, capacidad máxima infinita y capacidad mínima igual a la demanda del nodo destino original. Gráficamente se muestra en la siguiente figura:



Para convertir la red anterior (con un solo nodo fuente y un solo destino), a una red circulatoria, añádase un arco (t, s) que una el nodo superdestino con el nodo superfuente. Este arco tendría un costo cero o negativo (para obligar a la circulación del flujo), capacidad mínima dada por la suma de todas las demandas de los destinos originales, y una capacidad máxima dada por la suma de todas las ofertas de los nodos fuentes originales. Obviamente para que el problema resulte factible, la capacidad máxima de este arco (t, s), debe ser mayor o igual a su capacidad mínima. Gráficamente se tiene:



1.6.5 Arboles y Bosques

Un árbol es una gráfica conectada y sin ciclos, mientras que un bosque es una gráfica sin ciclos. Un árbol dirigido D = (N_t,M_t), es una subred que define una trayectoria única entre algún nodo específico, llamado nodo raíz, y cualquier otro nodo en N_t. Una subred de expansión de D es una subred con N_s = N, por consiguiente, un árbol de expansión (o generador) de D tiene N_t=N. Un bosque de expansión de D es una subred de expansión

dirigida, desconectada y sin ciclos.

Características de los árboles

1. El número de arcos en un árbol es n-1 donde n es el número de nodos, ya que este es el mínimo número de arcos necesarios para tener una red conexa y el máximo número posible para que no haya ciclos.

2. En un árbol de expansión dirigido solo el nodo raíz no es terminal de algún arco.

3. Cualquier número de arcos se pueden originar en un nodo del árbol dirigido.

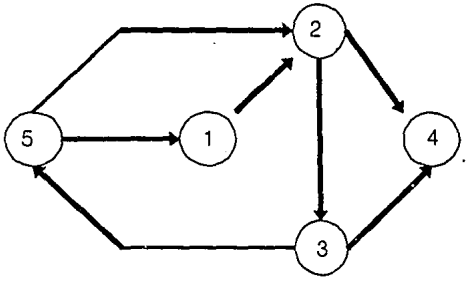
4. Entre cualesquier par de nodos hay una única trayectoria.

5. Las n-1 columnas de la matriz A, que forman la base para el problema de flujo a costo mínimo, corresponden a los arcos en Mt de un árbol de expansión. (Se explicará este punto con mayor detalle en la sección de problemas de flujo a costo mínimo)

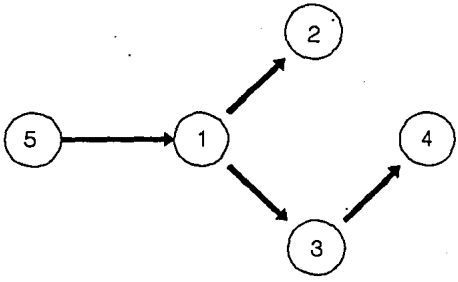
6. Los arcos que no se incluyen en Mt reciben el nombre de aristas. Si una arista es agregada a un árbol, se forma un único ciclo y la red resultante ya no es un árbol.

7. Si existe una trayectoria dirigida o cadena del nodo i al nodo j entonces, el nodo i precede al j y el nodo j sucede al i.

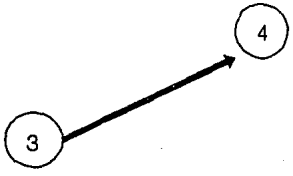
En la figura 1.9 se ilustran árboles obtenidos de la figura 1.9a. En (b) se tiene un árbol de expansión y (c) y (d) tomadas juntas constituyen un bosque de expansión.



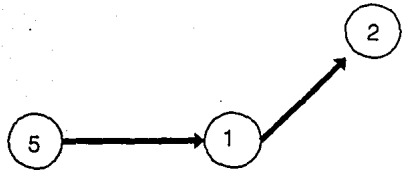
(a)



(b)
raiz=5
Nt={1,2,3,4,5}



(c)
raiz=3
Nt={3,4}



(d)
raiz=5
Nt={1,2,5}

Figura 1.9
Ejemplos de Redes de Arboles

1.6.6 Redes de Transporte

Se llama **red de transporte** a una gráfica finita sin anillos tal que :

ι) Cada arco k tiene asociado un número $C(k) \geq 0$ llamado capacidad del arco .

ιι) Existe un vértice X_0 y uno solo tal que $W^-(X_0)=\phi$ llamado **fuente de la red.**

ιιι) Existe un vértice X_n y uno solo tal que $W^+(X_n)=\phi$ llamado **sumidero de la red.**

1.6.6.1 Flujo

Sean $W^-(x_i)$ el conjunto de los arcos incidentes al vértice X_i hacia el interior y $W^+(X_i)$ el conjunto de arcos incidentes al vértice X_i hacia el exterior. Se dice que una función entera $\varphi(K)$ definida sobre el conjunto A de los arcos K_i es un flujo para una red de transporte si se tiene:

$$\varphi(k) \geq 0 \quad \forall\, k \in A$$

$$\sum_{k \in W^-(X_i)} \varphi(k) = \sum_{k \in W^+(X_i)} \varphi(k) \qquad \begin{cases} Xi \neq X_0 \\ Xi \neq X_0 \end{cases} \qquad (1.1)$$

$$\varphi(k) \leq C(k) \quad \forall\, k \in A$$

Puede pensarse que $\varphi(k)$ es un gasto que fluye por el arco k y que nunca puede exceder la capacidad de dicho arco. Aún más, si X_i no es ni la fuente ni el sumidero, entonces (1.1) significa que el gasto que llega a X_i es igual al gasto que sale de X_i, (propiedad conservativa de flujo). Es decir:

$$\sum_{u \in W^-(X_0)} \varphi(k) = \sum_{u \in W^+(X_n)} \varphi(k) = \phi(X_n) \qquad (1.1)$$

Donde a $\phi(X_n)$ se le llama el valor del flujo.

En este trabajo se tratará el problema de calcular el valor óptimo del flujo correspondiente a una red de transporte a través de técnicas de redes de optimización.

1.6.6.2 Arco Saturado, Flujo Completo.

Se dice que un arco $K_i \in A$ está saturado si se tiene: $\varphi(k_i) = C(k_i)$

Un flujo es completo si todo camino que va de la fuente al sumidero contiene al menos un arco saturado.

En la figura 1.10 se muestra una red de transporte y un flujo asociado a esta red. Obsérvese que los arcos con trazo grueso están saturados y que el flujo no es completo ya que al menos existe el camino (X_0, X_1, X_5, X_9) que no posee ningún arco saturado.

1.6.6.3 Corte, Capacidad De Un Corte.

Sea un conjunto Y c X de vértices que contiene al sumidero X_n y no contiene a la fuente X_o. El conjunto $\bar{W}(Y)$ de los arcos incidentes a Y hacia el interior es un corte de la red. Por ejemplo, si en la figura 1.10 se tienen Y = $\{X_7, X_9\}$ entonces el corte correspondiente a Y está dado por: $W(Y)=\{(X_1,X_7),(X_2,X_7),(X_3,X_7),(X_4,X_9),(X_5,X_9),(X_6,X_9),(X_8,X_9)\}$.

A la expresión $C[\bar{W}(Y)] = \sum_{k \in W(Y)} c(k)$

se le llama capacidad del corte $\bar{W}(Y)$. Así en el ejemplo se tiene:

$C[\bar{W}(Y)] = 8 + 8 + 7 + 10 + 10 + 15 + 5 = 63.$

Como siempre Y contiene al sumidero, toda unidad de gasto que va de X_o a X_n pasa al menos una vez por un arco $\bar{W}(Y)$ y, consecuentemente, cualquiera que sea un flujo $\phi(X_n)$ y un corte W(Y), se tendrá:

$$\phi(X_n) \leq C\,[\bar{W}\,(Y)]$$

Luego, existe un flujo $\phi_o(X_n)$ y un corte W (Y_o) tales que:

$$\phi_o(X_n) = C[\bar{W}(Y_o)]$$

Entonces el flujo $\phi_o(X_n)$ tiene un valor máximo y el corte $\bar{W}(Y_o)$ tiene una capacidad mínima, lo que origina el siguiente teorema:

TEOREMA DEL CORTE MINIMO - MAXIMO FLUJO
En una red de transporte dada, el valor máximo de un flujo es igual a la capacidad mínima de un corte, esto es:

$$\max \phi(X_n) = \min C[\bar{W}(Y)]$$

Es decir, si se logra encontrar un flujo igual a la capacidad de un corte mínimo, se estará seguro de que dicho flujo es máximo.

En esta idea está basado el algoritmo de **Ford y Fulkerson**, el cual permite calcular el flujo máximo asociado a una red de transporte.

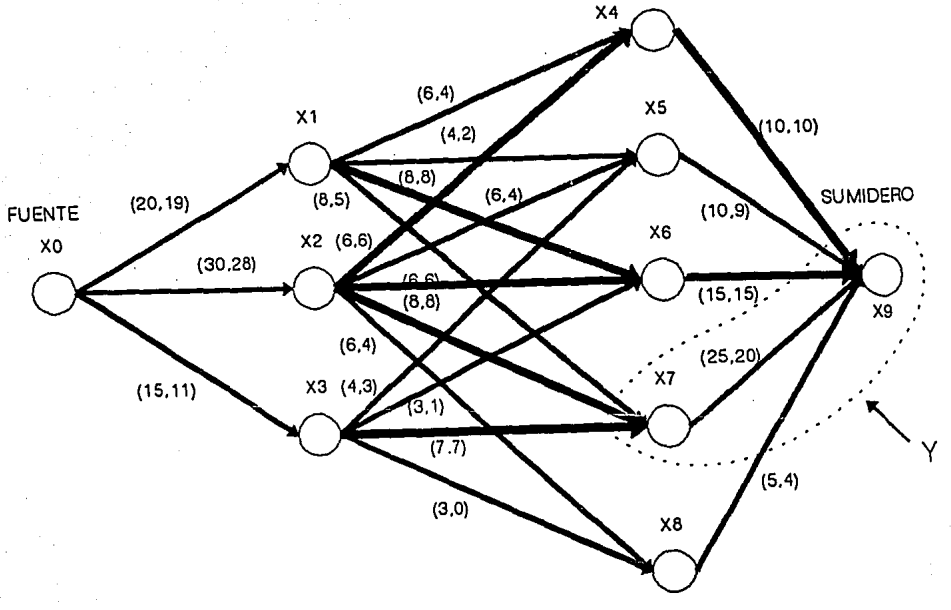


Figura 1.10 (C(ki),ϕ(ki))

1.7 REDES EXPANDIDA Y MARGINAL

1.7.1 Red Expandida

Los modelos de redes considerados aquí se basan en gráficas dirigidas. Es conveniente, desde el punto de vista de obtener soluciones óptimas a los problemas de redes de flujo, definir una red expandida, la cual se obtiene directamente de la red original.

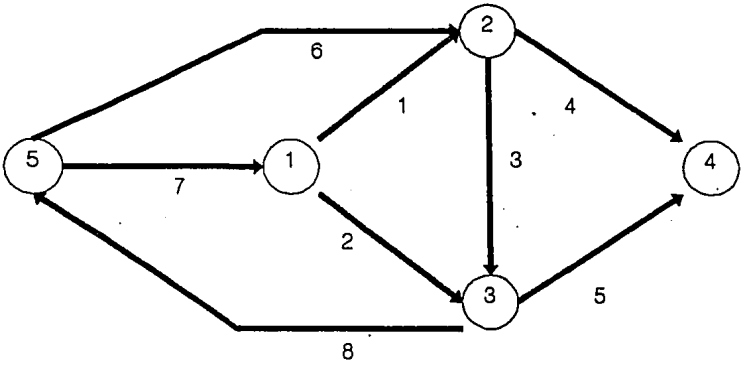
Primero es necesario definir la existencia de un arco reflejado, -k, para k $\in$ A tal que el arco -k conecta los mismos nodos que k (arco hacia adelante) pero tienen dirección opuesta.

$$O(-k) = t(k)$$
$$t(-k) = O(k)$$

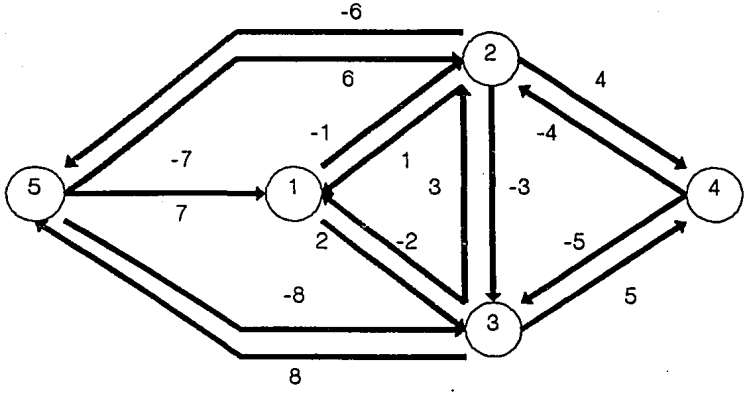
La red expandida, $D_E = [N, N_E]$, tiene el mismo conjunto de nodos que D y su conjunto de arcos contiene, además de los arcos de D, todos los arcos reflejados, esto es, si $S=\{(i,j),(k,l),\ldots,(s,t)\}$ entonces $S_E=\{(i,j),(k,l),\ldots,(s,t),(j,i),(l,k),\ldots,(t,s)\}$. La figura 1.11 muestra una red dirigida y su red expandida asociada.

Nótese que si la red original 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 su asociado arco hacia adelante.

Para una red sin ganancia, si C_k es el costo para el arco hacia adelante k, entonces $-C_k$ es el costo para el arco reflejado -k.



(a)



(b)

Figura 1.11
Ejemplo de Red Expandida (a) Red Dirigida
(b) Red Expandida Asociada

1.7.2 Red Marginal

La red marginal, $D^{\bullet}=[N,A^{\bullet}]$, tiene el mismo conjunto de nodos que D y D_E. El conjunto de arcos de $D^{\bullet}$ consiste de un subconjunto de

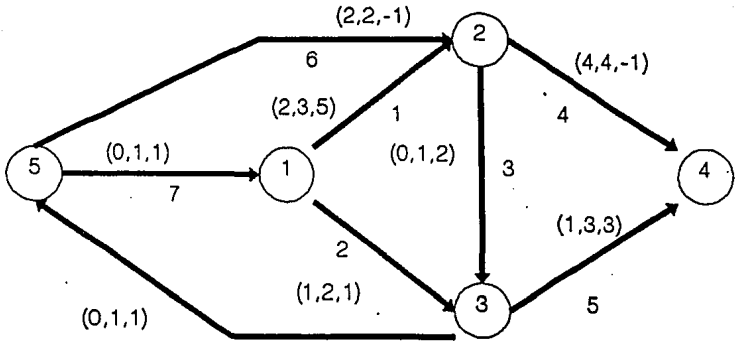
A_E que incluye solo arcos admisibles. Un arco hacia adelante es admisible si el flujo no ha alcanzado su capacidad; es decir, si es posible incrementar el flujo en la red original. Un arco reflejado es admisible si el flujo en el correspondiente arco hacia adelante no es igual a cero (cota inferior); esto es, si es posible disminuir el flujo en el arco asociado de la red original.

La definición usual de la función que indica la admisibilidad de los arcos de la red expandida es .

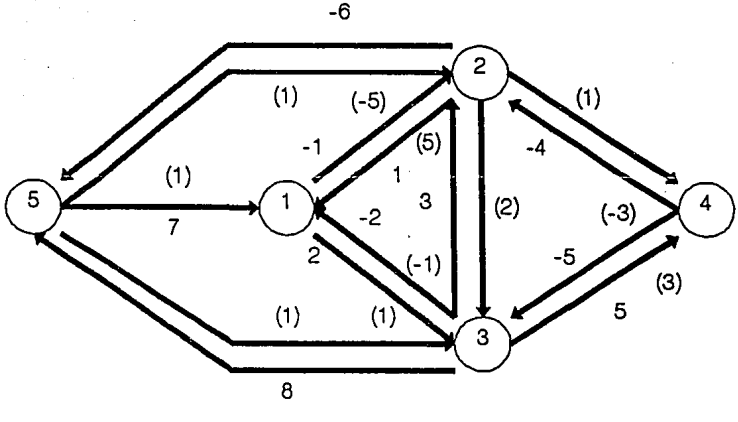
$$A_d(k) = 1 \begin{cases} \text{si } k > 0 \text{ y } \varphi_k > c_k \\ \text{si } k < 0 \text{ y } \underline{\varphi_k} > 0 \end{cases} \qquad \text{arco admisible}$$

$$A_d(k) = 0 \begin{cases} \text{si } k > 0 \text{ y } \varphi_k \geq c_k \\ \text{si } k < 0 \text{ y } \underline{\varphi_k} \leq 0 \end{cases} \qquad \text{arco inadmisible}$$

La figura 1.12b muestra la red marginal asociada con la figura 1.12a como resultado de aplicar la anterior función de admisibilidad.


(2,2,-1)
6
2
(4,4,-1)
(2,3,5)
1
4
5
(0,1,1)
1
(0,1,2)
3
4
7
(1,3,3)
2
(1,2,1)
3
5
(0,1,1)
8

1.12 (a)


-6
2
(1)
(-5)
(1)
-1
(5)
-4
(1)
1
5
(1)
3
(2)
(-3)
4
7
-2
2
(-1)
-5
(3)
(1)
(1)
5
3
8

1.12 (b)

CAPITULO II

EL PROBLEMA GENERAL DE REDES

Una clase general de problemas de redes queda resumida en el denominado problema de redes con flujos restringidos y costo mínimo. Una característica importante de este problema es que siendo uno de programación lineal tiene métodos de solución más sencillos y eficientes que el simplex revisado y sus variantes.

Una de las razones por la cual existe gran concentración en los problemas de esta área es debido a que los grandes problemas de transporte que involucran cientos de restricciones y miles de variables pueden ser fácilmente resueltos por medio de las técnicas de optimización de redes, mientras que actualmente es una tarea imposible resolver un problema general de programación lineal de estas dimensiones.

Dentro de los métodos de solución para este tipo de problemas, los dos más eficientes son: el método heurístico, diseñado por **Busacker y Gowen** y el método basado en el **teorema de holguras complementarias**, que sirvió a Ford y Fulkerson para desarrollar uno de los algoritmos más potentes de la Investigación de Operaciones. Este algoritmo se conoce con el nombre de **Out-Of-Kilter**, que significa fuera de orden.

El algoritmo de Busacker y Gowen, tiene la gran ventaja de que es muy fácil de entender y de utilizar para problemas relativamente pequeños. Su gran desventaja consiste en que en cada iteración se requiere la construción de nuevos arcos, los arcos en reversa,

y por lo tanto, el número de elementos de la red original también aumenta. En redes de tamaño medio y peor aún, en redes de gran dimensión, esto resulta un inconveniente muy serio, pues la memoria de la computadora tiende a saturarse rápidamente.

En conclusión, el algoritmo de Busacker y Gowen limita de inmediato el tamaño de las redes que se pueden resolver. Se muestra este algoritmo en la rutina A-I (Anexos).

Debido a las desventajas que presenta el algoritmo de Busacker y Gowen se desarrolló el algoritmo **Out-Of-Kilter** que resuelve el problema de flujos restringidos en una red a costo mínimo, sin modificar la estructura de la red. Este algoritmo está basado en los teoremas de holguras complementarias, de la programación lineal, los cuales serán analizados en temas posteriores.

El algoritmo Out-Of-Kilter requiere que todos los elementos U_{ij} (capacidades) sean números enteros para todos los arcos. Además requiere que la red a resolver sea circular (red donde prácticamente han dejado de existir el nodo fuente y el nodo destino, al añadir un arco artificial con costo cero).

Este algoritmo puede iniciar con cualquier clase de flujo X_{ij} en la red. Si este flujo para empezar ya es factible, entonces el algoritmo lo convierte en óptimo. Si no es factible, entonces el algoritmo lo convierte primero en factible y después en óptimo. Cabe también mencionar que este tipo de redes determinísticas (el flujo total que entra a un nodo es igual al flujo total que sale de otro, es decir, no hay pérdida de flujo), ha sido extendido a el caso estocástico por Connors y Zangwill. El algoritmo antes explicado se expresa en la rutina A-II (Anexos).

2.1. El problema de redes con flujos restringidos y costo mínimo

El problema consiste en considerar una red circulatoria en que los flujos permitidos en cada arco están sujetos a cuotas superiores e inferiores, y donde además se tiene un costo por unidad de flujo que pasa en cada arco. El objetivo que se persigue es determinar el flujo en la red que proporcione el costo mínimo.

Consideremos la red circulatoria $G = (N,A)$. Supongamos que $0 \leq a_{ij} \leq b_{ij} \leq +\infty$ son los flujos mínimos y máximo permitidos en cada arco $(i,j) \in A$, y denotemos por c_{ij} el costo por unidad de flujo que pasa por este arco. Entonces se desea

$$
\begin{cases}
\text{Minimizar} \quad \sum_{(i,j)\in A} c_{ij}\, x_{ij} & \\
\text{s.a.} & \\
\sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = 0 & i \in N \\
a_{ij} \leq x_{ij} \leq b_{ij} & (i,j) \in A
\end{cases}
$$

donde el vector de variables de decisión, $x = (x_{ij})$, se denomina flujo de la red. Este es el denominado problema de redes con flujos restringidos y costo mínimo.

Flujo Circulatorio

En este problema se dice que un flujo que satisface las restricciones en los nodos es un flujo circulatorio.

Flujo Factible

Asimismo, un flujo que satisface todas las restricciones del problema se dice que es un flujo factible.

Flujo Factible Básico

Por otra parte, un flujo factible que es una solución básica de las restricciones del problema de redes, se dice que es un flujo factible básico.

Flujo Factible Entero

Finalmente, un flujo factible, $x = (x_{ij})$, se dice que es entero si para cada arco (i,j) el flujo, esto es x_{ij}, es cero o un número entero positivo.

Una gran variedad de problemas prácticos pueden formularse como el de flujos restringidos con costo mínimo; ejemplos particulares son el de la ruta más corta, el del flujo máximo entre dos nodos de una red, el de transporte, el de distribución de bienes y otros. Es también conveniente mencionar que si los flujos mínimo y máximo permitidos en cada arco de la red son números enteros, entonces los flujos factibles básicos (y óptimos) son enteros. Como consecuencia de esta propiedad es frecuente plantear problemas lineales cuyas soluciones deben ser necesariamente enteras, en términos de redes.

2.2 El Problema Dual

Considerando el problema de redes formulado anteriormente en el que la red circulatoria es de la forma G = (N,A), donde N=(1,2,...,n) y A es un conjunto de arcos conocidos. Es sencillo verificar que este problema es equivalente a:

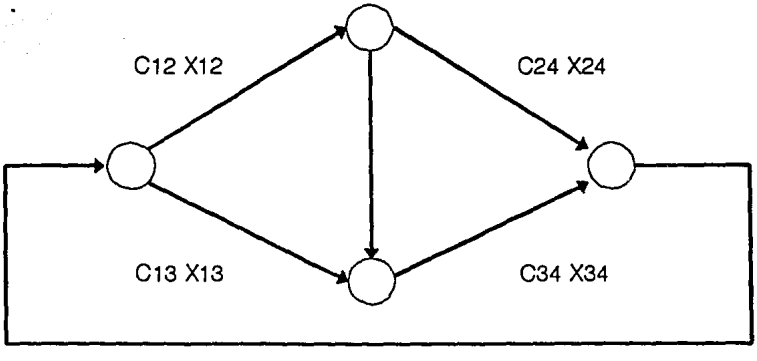
$$
\begin{cases}
\text{Minimice} & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\, x_{ij} \\
\text{s.a} & \\
 & \displaystyle\sum_{j=1}^{n} (x_{ij} - x_{ji}) = 0 \quad i=1,2,\ldots,n \\
 & a_{ij} \le x_{ij} \le b_{ij} \qquad i,j=1,2,\ldots,n
\end{cases}
$$

donde $a_{ij} = b_{ij} = 0$ si $(i,j) \notin A$. Note que en estos últimos arcos el flujo es siempre $x_{ij} = 0$. También note que la nueva formulación corresponde a crear arcos ficticios en la red cuyos flujos son igual a cero. Por otra parte, para estos arcos, el valor del costo unitario c_{ij} en la función objetivo, es completamente irrelevante.

A continuación se trata de obtener la formulación dual del problema primario. La formulación dual no es fácil de obtener y

por lo tanto se muestra con un ejemplo muy pequeño y de ahí tratar de generalizar.

Sea la red circulatoria de la figura:



El problema primario es:

Min $Z = C_{12}X_{12} + C_{13}X_{13} + C_{23}X_{23} + C_{24}X_{24} + C_{34}X_{34}$

s.a. i) Flujo que entra a un nodo es igual al flujo que sale del nodo. $\sum \varphi(k) = \sum \varphi(k)$

		Variable dual
Nodo 1	$-X_{12}-X_{13}+X_{41} = 0$	U_1
Nodo 2	$-X_{23}-X_{24}+X_{12} = 0$	U_2
Nodo 3	$-X_{34}+X_{13}+X_{23} = 0$	U_3
Nodo 4	$-X_{41}+X_{24}+X_{34} = 0$	U_4

ii) Flujo en un arco ≥ capacidad mínimo del arco

	Variable dual
$X_{12} \geq a_{12}$	V_{12}
$X_{13} \geq a_{13}$	V_{13}
$X_{23} \geq a_{23}$	V_{23}
$X_{24} \geq a_{24}$	V_{24}
$X_{34} \geq a_{34}$	V_{34}
$X_{41} \geq a_{41}$	V_{41}

iii) Flujo en un arco ≤ capacidad máxima del arco

	Variable dual
$X_{12} \leq b_{12}$	W_{12}
$X_{13} \leq b_{13}$	W_{13}
$X_{23} \leq b_{23}$	W_{23}
$X_{24} \leq b_{24}$	W_{24}
$X_{34} \leq b_{34}$	W_{34}
$X_{41} \leq b_{41}$	W_{41}

El problema dual es el siguiente:

$$\text{Max } Z = 0U_1 + 0U_2 + 0U_3 + 0U_4 + a_{12}V_{12} + a_{13}V_{13} + a_{23}V_{23} + a_{24}V_{24} + a_{34}V_{34} + a_{41}V_{41} - b_{12}W_{12} - b_{13}W_{13} - b_{23}W_{23} - b_{24}W_{24} - b_{34}W_{34} - b_{41}W_{41}$$

s.a.

$$U_1 - U_2 + V_{12} - W_{12} \leq C_{12}$$

$$U_{12} - U_3 + V_{23} - W_{23} \leq C_{23}$$

$$U_2 - U_4 + V_{24} - W_{24} \leq C_{24}$$

$$U_3 - U_4 + V_{34} - W_{34} \leq C_{34}$$

$$U_4 - U_1 + V_{41} - W_{41} \leq C_{41}$$

$V_{12}, V_{13}, V_{23}, V_{24}, V_{34}, V_{41} \geq 0$; $W_{12}, W_{13}, W_{23}, W_{24}, W_{34}, W_{41} \geq 0$

U_1, U_2, U_3, U_4 no restringidas en signo.

En términos generales, si la formulación primal de un problema de flujo máximo en una red a costo mínimo es:

$$
\begin{cases}
\text{Min } Z = \displaystyle\sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} X_{ij} \\
\text{s.a} \\
\displaystyle\sum_{j=1}^{n} (X_{ij} - X_{ji}) = 0 \qquad i=1,2,\ldots,n \\
a_{ij} \leq X_{ij} \leq b_{ij} \qquad i,j=1,2,\ldots,n
\end{cases}
$$

el problema anterior es equivalente a:

$$
\begin{cases}
\text{Min } Z = \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\, x_{ij} \\
\text{s.a.} \quad \displaystyle\sum_{j=1}^{n} (x_{ij} - x_{ji}) \geq 0 \qquad i=1,2,\ldots,n \\
\displaystyle\sum_{j=1}^{n} -(x_{ij} - x_{ij}) \geq 0 \qquad i=1,2,\ldots,n \\
x_{ij} \geq a_{ij} \\
-x_{ij} \geq -b_{ij} \\
x_{ij} \geq 0
\end{cases}
$$

En forma matricial:

$$\text{Min} \begin{bmatrix} c_{11}, c_{12}, \ldots, c_{nn} \end{bmatrix} \quad \begin{bmatrix} x_{11} \\ x_{12} \\ \cdot \\ \cdot \\ \cdot \\ x_{nn} \end{bmatrix}$$

s.a Restricciones en los nodos

$$\begin{bmatrix} R \\ -R \\ I_o \\ -I_o \end{bmatrix} \quad \begin{bmatrix} x_{11} \\ x_{12} \\ \\ x_{nn} \end{bmatrix} \quad \leq \quad \begin{bmatrix} 0 \\ 0 \\ a \\ -b \end{bmatrix}$$

Restricciones en los arcos

$$x_{ij} \geq 0$$

donde:

$$R = \begin{bmatrix} [0,1,1,..,] & [-1,0,0,..,] & \cdots & [-1,0,0,..,] \\ [0,-1,0,..,] & [1,0,1,..,] & \cdots & [0,-1,0,..,] \\ \vdots & \vdots & & \vdots \\ [0,0,..,-1] & [0,0,..,-1] & \cdots & [1,1,..,0] \end{bmatrix}_{n \times n}$$

$$Io = \begin{bmatrix} 0 & & & & & & & & & & & \\ & 1 & & & & & & & & & & \\ & & 1 & & & & & & & & & \\ & & & \ddots & & & & & & & & \\ & & & & \ddots & & & & & & & \\ & & & & & 0 & & & & & & \\ & & & & & & 1 & & & & & \\ & & & & & & & 1 & & & & \\ & & & & & & & & \ddots & & & \\ & & & & & & & & & 0 & & \\ & & & & & & & & & & \ddots & \\ & & & & & & & & & & & \ddots \end{bmatrix}_{n \times n}$$

El problema dual será entonces:

$$\text{Max } Z = [s,t,v,w] \begin{bmatrix} 0 \\ 0 \\ a \\ -b \end{bmatrix}$$

s.a.

$$[s,t,v,w] \begin{bmatrix} R \\ -R \\ I_o \\ -I_o \end{bmatrix} \leq \begin{bmatrix} c_{11} \\ c_{12} \\ \cdot \\ \cdot \\ \cdot \\ c_{nn} \end{bmatrix}$$

$$s,t,v,w \geq 0$$

problema que es equivalente a:

$$\text{Max } Z = av - bw$$

s.a. $$sR - tR + vI_o - wI_o \leq C$$

$$s,t,v,w \geq 0$$

haciendo u = s-t la restricción queda como:

$$uR + vI_o - wI_o \leq C$$

u no restringida en signo ya que si:

$$s > t \Rightarrow u > 0$$

$$s = t \Rightarrow u = 0$$

$$s < t \Rightarrow u < 0$$

Finalmente el dual queda como:

$$\text{Max } Z = a_{11} v_{11} + a_{12} v_{12} + \ldots + a_{nn} v_{nn} - b_{11} w_{11} - b_{12} w_{12} - \ldots - b_{nn} w_{nn}$$

s.a.
$$u_1 - u_2 + v_{12} - w_{12} \leq c_{12}$$
$$u_1 - u_3 + v_{13} - w_{13} \leq c_{13}$$
$$\vdots$$
$$u_{n-1} - u_n + v_{n-1,n} - w_{n-1,n} \leq c_{n-1,n}$$

$u_1, u_2, \ldots, u_n$ no restringido $v_{11}, v_{12}, \ldots, v_{nn}, w_{11}, w_{12}, \ldots, w_{nn} \geq 0$

que se puede expresar como:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} v_{ij} - b_{ij} w_{ij}$$

s.a.
$$u_i - u_j + v_{ij} - w_{ij} \leq c_{ij}$$
$$u_i \text{ no restringida} \quad ; \quad v_{ij} \; w_{ij} \geq 0$$

2.3. Condiciones de optimalidad del problema de redes.

En el problema de redes equivalente y su dual se tiene que, si un flujo es factible y v, w y u son vectores de variables duales factibles, entonces estas soluciones son óptimas sí y sólo si se satisfacen las condiciones de complementariedad, esto es, se cumple que:

$$\lambda^* [Ax^* - b] = 0 \quad . \ . \ . \quad (1)$$

$$X^* [c - \lambda^* A] = 0 \quad . \ . \ . \quad (2)$$

Desarrollando (1) y haciendo u = s-t se tiene

$$u_1[(x_{12} + x_{13} + \ldots + x_{1n}) - (x_{21} + x_{31} + \ldots + x_{n1})] + u_2[(x_{21} + x_{23} + \ldots + x_{2n}) - (x_{12} + x_{32} + \ldots + x_{n2})] + \ldots + u_n[(x_{n1} + x_{n2} + \ldots + x_{nn}) - (x_{1n} + x_{2n} + \ldots + x_{nn})] + v_{12}(x_{12} - a_{12}) + v_{13}(x_{13} - a_{13}) + \ldots + v_{nn}(x_{nn} - a_{nn}) + w_{12}(b_{12} - x_{12}) + w_{13}(b_{13} - x_{13}) + \ldots + w_{nn}(b_{nn} - x_{nn}) = 0 \quad . \ . \ . \quad (3)$$

Los primeros n términos de la ecuación (3) se satisfacen para todo flujo factible por la propiedad de conservación de flujo en la red $(x_{ij} - x_{ji}) = 0$. Además para que se cumpla la igualdad (3) es necesario que:

$$v_{ij}(x_{ij} - a_{ij}) = 0 \quad y \quad w_{ij}(b_{ij} - x_{ij}) = 0$$

Desarrollando (2):

$$(c_{12} - u_1 + u_2 - v_{12} + w_{12})\, x_{12} = 0$$

$$(c_{13} - u_1 + u_3 - v_{13} + w_{13})\, x_{13} = 0$$

$$\vdots$$

$$(c_{n-1'n} - u_{n-1} + u_n - v_{n-1',\, w_{n-1'n}})\, x_{n-1'n} = 0$$

Resumiendo (1) y (2) podemos decir que para el problema de redes con flujos restringidos una solución factible y óptima debe cumplir que:

$$v_{ij}(x_{ij} - a_{ij}) = 0$$

$$w_{ij}(b_{ij} - x_{ij}) = 0$$

$$x_{ij}(c_{ij} - u_i + u_j - v_{ij} + w_{ij}) = 0$$

para toda i, j = 1, 2,...,n

Es sencillo verificar que esta condición de complementariedad es equivalente a que se cumpla una de las condiciones:

$$x_{ij} = a_{ij} \;;\; w_{ij} = 0 \;;\; v_{ij} = c_{ij} - u_i + u_j \geq 0$$

$$a_{ij} < x_{ij} < b_{ij} \;;\; v_{ij} = w_{ij} = c_{ij} - u_i + u_j = 0$$

$$x_{ij} = b_{ij} \;;\; v_{ij} = 0 \;;\; w_{ij} = -c_{ij} + u_i - u_j \geq 0$$

Para toda i, j = 1, 2,...,n

En este punto conviene observar que para los arcos tales que $a_{ij} = b_{ij} = 0$ (como es el caso de aquellos arcos que no

pertenecen a la red original) las condiciones de complementariedad se satisfacen, esto es, los arcos ficticios del problema de redes equivalente satisfacen estas condiciones.
Una forma de verificar la condición de complementariedad con un mínimo de información, es la siguiente:

Sea $x = (x_{ij})$ un flujo factible del problema de redes y $u = (u_i)$ el vector de potenciales en las redes, entonces, para calcular los vectores $v = (v_{ij})$ y $w = (w_{ij})$ se usan las fórmulas:

$$v_{ij} = \max [0, \bar{c}_{ij}] \qquad w_{ij} = \max [0, -\bar{c}_{ij}]$$

donde $\bar{c}_{ij} = c_{ij} - u_i + u_j$ es denominado el costo relativo del arco (i, j). Como consecuencia de estas formulas es sencillo verificar que si x y u forman parte de las soluciones óptimas del problema de redes y su dual, entonces los vectores x, u, v y w son vectores factibles y óptimos, esto es, satisfacen las condiciones de complementariedad.

Como conclusión de la discusión anterior podemos establecer que un flujo circulatorio y factible $x = (x_{ij})$, es óptimo, sí y sólo sí existe un vector de potenciales $u = (u_i)$ tal que para cada arco de la red se satisface alguna de las condiciones o estados siguientes:

$$\bar{c}_{ij} > 0 \quad \Rightarrow \quad x_{ij} = a_{ij}$$

$$\bar{c}_{ij} = 0 \quad \Rightarrow \quad a_{ij} \leq x_{ij} \leq b_{ij}$$

$$\bar{c}_{ij} < 0 \quad \Rightarrow \quad x_{ij} = b_{ij}$$

Por otra parte, si el flujo circulatorio x y el vector de potenciales u no son óptimos, se tiene que algunos de sus arcos se encuentran en las condiciones o estados siguientes:

$$\bar{c}_{ij} > 0 \quad ; \quad x_{ij} < a_{ij}$$

$$\bar{c}_{ij} > 0 \quad ; \quad x_{ij} > a_{ij}$$

$$\bar{c}_{ij} = 0 \quad ; \quad x_{ij} < a_{ij}$$

$$\bar{c}_{ij} = 0 \quad ; \quad x_{ij} > b_{ij}$$

$$\bar{c}_{ij} < 0 \quad ; \quad x_{ij} < b_{ij}$$

$$\bar{c}_{ij} < 0 \quad ; \quad x_{ij} > b_{ij}$$

2.4 Propiedades de la matriz A, del problema de flujo con costo mínimo.

Es de interés particular la estructura de la matriz A. Cada columna de A tiene dos elementos diferentes de cero, +1, y -1, y debido a su estructura, la matriz A es de rango n-1. Físicamente cada renglón de A corresponde a una ecuación de conservación de flujo, en cierto nodo. Otra propiedad de A que es importante, es que cada subdeterminante de A, tiene valor de 0, ó ±1, esta es la llamada propiedad de unimodularidad. Es esta propiedad de unimodularidad, la que garantiza que la solución óptima sea entera, si el vector b, es un vector entero. Una matriz se dice

totalmente unimodular, si cada subdeterminante de A, es igual a 0, ó ±1. La condición de que A es totalmente unimodular, es suficiente para la existencia de soluciones óptimas enteras, la dificultad es demostrar, que la condición es también necesaria. La demostración se debe a Veinott y Dantzing. [1]

Rango de la matriz A.

Para mostrar que la matriz A tiene rango m-1, necesitamos seleccionar una submatriz de A de (m-1)x(m-1) que sea no singular.

Sea T cualquier árbol de expansión de la red G. El árbol T consiste de m nodos y m-1 arcos de G que no forman un ciclo. Considere la submatriz A_T de A de mx(m-1) asociada con los nodos y arcos de T. Puesto que m≥2, T tiene a lo menos un nodo terminal, digamos el nodo k, con exactamente un arco que incide sobre ese nodo. En tal caso el k-ésimo renglón de A_T contiene un elemento diferente de cero. Permutando los renglones y las columnas de A_T de tal manera que, el elemento diferente de cero del k-ésimo renglón, sea colocado en la primera columna y primer renglón. Entonces:

$$A_T = \left[\begin{array}{c|c} \pm 1 & 0 \\ \hline p & A_T' \end{array}\right]$$

Borrando el primer renglón y la primera columna de A_T y considerando la matriz A_T' de (m-1)x(m-2), correspondiente a la gráfica T' de T, a la cual se le ha quitado el nodo k y el arco incidente. T' es también un árbol. Debe de contener a lo menos

[1] Veinott, Dantzing, Integral Extreme Points. SIAM, Review, Vol. 10, No. 3, julio de 1968.

un nodo terminal, digamos el nodo *l*. Permutando los renglones y las columnas de A_T' de tal forma que el elemento diferente de cero en el renglón *l*, quede en el primer renglón, primera columna, podemos escribir A_T como:

$$A_T = \begin{bmatrix} \pm 1 & 0 & 0 \\ p_1 & \pm 1 & 0 \\ p_2 & q & A_T'' \end{bmatrix}$$

Podemos continuar de esta manera exactamente m-1 veces. Borrando la última columna de A_T' obtenemos una matriz de (m-1)x(m-1) que es triangular inferior, con elementos en la diagonal diferentes de cero y que por lo tanto es no singular. Entonces el rango de A es m-1.

Teorema. Una matriz es unimodular si las siguientes cuatro condiciones se satisfacen:

Condición 1. Cada columna contiene a lo más, dos elementos diferentes de cero.

Condición 2. Cada elemento es 0 ó±1. La matriz A puede ser descompuesta en dos conjuntos disjuntos de renglones R_1 y R_2 tal que:

2a) Dos elementos diferentes de cero en una columna con el mismo signo, no están contenidos en el mismo conjunto de renglones.

2b) Dos elementos diferentes de cero en una columna con signos diferentes, están contenidos en el mismo conjunto de renglones.

Prueba: Es suficiente con probar que cualquier matriz cuadrada A, que satisface las cuatro condiciones anteriores, tiene su determinante igual a 0 ó ±1. La prueba es por inducción sobre el tamaño de la matriz. Para una matriz de uno por uno, el teorema es verdadero por la condición 2. Considere ahora que el teorema es verdadero para una matriz de (n-1)x(n-1), y A es una matriz de nxn.

Si alguna columna de A tiene todos sus elementos igual a cero, entonces det(A) = 0. Si alguna tiene exactamente un elemento diferente de cero, entonces, expandiendo para esta columna det(A) = ± det(A'), donde A' es el cofactor de los elementos diferentes de cero, y tiene un determinante igual a 0 ó ±1, por la hipótesis de inducción.

Consideremos que cada columna de la matriz A, contiene exactamente dos elementos diferentes de cero. De las condiciones 1 y 2 se sigue que:

$$\sum_{i \in R_1} a_{ij} = \sum_{i \in R_2} a_{ij} \qquad j=1,\ldots,n$$

Esto implica que det(A) = 0. Note que el argumento es válido, aún cuando alguno de los conjuntos R_1 ó R_2 es vacío.

Teorema. La matriz de los coeficientes del sistema, del problema de flujo con costo mínimo, es totalmente unimodular.

Prueba. Sea B cualquier submatriz kxk de A. Se prueba por inducción que det(B) = 0 ó ±1, y entonces que la matriz A es

totalmente unimodular. Obviamente si k=1, entonces det(B) = 0 ó ±1. Supongamos ahora que det(B) = 0, ±1, para cualquier matriz cuadrada de tamaño k-1.

Sea B una submatriz kxk de A. Si alguna columna de B es cero, entonces det(B) = 0. Si cada columna de B tiene un +1 y un -1, entonces det(B) = 0 (los renglones de B son linealmente dependientes, y sumándolos obtenemos el vector cero). Suponga ahora, que hay al menos una columna con un elemento diferente de cero. Entonces det(B) = ± det(B'), donde B' es la submatriz de B formada quitando esa columna y el renglón con elementos diferentes de cero. Por la hipótesis de inducción det(B') = 0 ó ±1, y entonces det(B) = 0 ó ±1, con lo que el argumento de inducción queda completo.

Teorema. (Soluciones enteras). Sea el problema de redes en forma circulatoria en el que los flujos mínimo y máximo permitidos en cada arco son números enteros. En este problema toda solución factible básica es entera. Asimismo, si existe una solución óptima, existe una solución óptima que es entera.

Prueba.
Sea $x = (x_{ij})$ una solución factible básica del problema de redes. Note que los valores de las variables básicas quedan determinadas en forma única si las variables no básicas han sido identificadas y tienen el valor cero.
Asociemos con esta solución una red :definida como G' = (N',A') donde N' = N y

$$A'=\{(i,j)\in A \ / \ x_{ij} > 0 \ ; \text{ no } x_{ij} \notin Z\}$$

Es claro que si el conjunto A' es vacio, el teorema queda demostrado.

Suponga que A' es un conjunto no vacio. En este caso se prueba, que G' tiene un ciclo y se contradice la hipótesis de que x es una solución factible básica. Se hace notar que en cada nodo, si existe un arco con flujo que entra, entonces existe un arco con flujo que sale, pues todos los nodos son de traspaso. Asimismo, si el flujo total que entra al nodo es un número entero, lo mismo es cierto del flujo total que sale.

Considere la gráfica dirigida G' = (n',A') definida anteriormente, seleccione un arco $a_1 \in A'$ y etiquete los nodos de este arco, ésto es, $a_1 = (i_1,i_2)$. Note que existe un arco $a_2 \neq a_1$ tal que $a_2 \in A'$ y que $a_2 = (i_2,i_3)$ ó bien $a_2 = (i_3,i_2)$. Si $i_3 = i_1$, se tiene un ciclo. Por otra parte, si $i_3 \neq i_2$ suponga, por simplicidad, que $a_2 = (i_2,i_3)$. Entonces, seleccione un arco $a_3 \neq a_2$ tal que $a_3 \in A'$ y que $a_3 = (i_4,i_3)$ ó $a_3 = (i_3,i_4)$. Si el nodo i_4 fue anteriormente etiquetado se tiene un ciclo. De otra manera podemos generar más arcos y etiquetar sus nodos hasta encontrar que un nodo ha sido reetiquetado. Esto se logra en un número finito de pasos. Sea i_r el nodo tal que $i_r = i_k$ para algún k<r. Entonces se tiene un ciclo con los nodos i_r, $i_{k+1},\ldots,i_r$.

Los arcos de este ciclo pertenecen al conjunto A' y puede dividirse en dos conjuntos

$$C_D = \{(i,j);(i,j) = (i_t,i_{t+1}), \text{ para algún } t\in\{k,k+1,\ldots,r-1\}\}$$

$C_I = \{(i,j);(i,j) = (i_{t+1},i_t)$, para algún $t \in \{k,k+1,\ldots,r-1\}\}$

Esto es, C_D y C_I son los conjuntos de arcos que siguen el sentido directo e inverso de la numeración de nodos consecutivos del ciclo respectivamente. De donde $x' = (x'_{ij})$ dado como

$$x'_{ij} = \begin{cases} x_{ij}+E & (i,j) \in C_D \\ x_{ij}-E & (i,j) \in C_I \\ x_{ij} & \text{De otra manera} \end{cases}$$

es un flujo circulatorio factible para $E > 0$ suficientemente pequeño. Sin embargo, esto contradice la hipótesis que $x = (x_{ij})$ es una solución factible básica. Por lo tanto, el conjunto A' es vacío y la prueba termina.

2.5 El Método de Solución del Problema de Redes.

El algoritmo "out-of-kilter" es similar al algoritmo "primal-dual", en el sentido de que el algoritmo empieza con una solución factible en el dual, pero no necesariamente con una solución factible del primal e interacciona entre el primal y el dual hasta que el óptimo es alcanzado. Este algoritmo se puede ver como una generalización del algoritmo primal dual para los problemas de flujo en redes.

2.5.1 Formulación del problema de redes con costo mínimo

Por conveniencia se muestra el problema de redes con costo mínimo en la siguiente forma:

$$\left\{\begin{array}{l} \text{Min} \quad \displaystyle\sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij}\, x_{ij} \\ \\ \text{s.a.} \\ \\ \displaystyle\sum_{j=1}^{m} x_{ij} - \sum_{k=1}^{m} x_{ki} = 0 \qquad i=1,\ldots,m \\ \\ x_{ij} \geq a_{ij} \quad i,j=1,\ldots,m \\ \\ x_{ij} \leq b_{ij} \quad i,j=1,\ldots,m \end{array}\right.$$

en donde las sumas y desigualdades son consideradas solamente para los arcos existentes en la red.

La primera restricción es de "conservación de flujo", las restricciones siguientes son de "capacidad" y dan la cota superior e inferior para cada arco en la red. Un flujo que satisface estas restricciones se denomina "flujo factible". Se considera a_{ij}, b_{ij} y c_{ij} enteros, y que $0 \leq a_{ij} \leq b_{ij}$.

Observamos en las restricciones de conservación de flujo que los valores del miembro derecho son ceros, de donde podemos concluir que el flujo en la red no tiene un punto inicial o un punto final, sino que circula continuamente a través de toda la red. Podemos mencionar entonces que la conservación de flujo en la red será a través de circuitos.

2.5.2 La estructura del dual en un problema de redes con costo mínimo y sus propiedades.

Si asociamos una variable dual u_i a cada "ecuación de conservación de flujo", una variable dual w_{ij} a cada restricción $x_{ij} \leq b_{ij}$ (la cual consideraremos como $-x_{ij} \geq -b_{ij}$ para propósitos de la formulación del dual), y una variable dual v_{ij} a cada restricción del tipo $x_{ij} \geq a_{ij}$, la formulación del problema dual en un problema de redes con costo mínimo esta dada por:

$$
\begin{cases}
\text{Max } Z = \displaystyle\sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij}\, v_{ij} - b_{ij}\, w_{ij} \\
\text{s.a.} \\
u_i - u_j + v_{ij} - w_{ij} = c_{ij} \qquad i,j=1,\ldots,m \\
v_{ij},\ w_{ij} \geq 0 \qquad i,j=1,\ldots,m \\
u_i \text{ no restringida} \qquad i=1,\ldots,m
\end{cases}
$$

en donde las sumas y las restricciones se consideran solamente para los arcos existentes en la red.

El problema dual tiene una estructura muy interesante. Supongamos que se selecciona **cualquier** conjunto u_i's (consideremos a través de todo el desarrollo que las u_i's son enteros). Entonces la retricción dual para un arco (i,j) es:

$$v_{ij} - w_{ij} = c_{ij} - u_i + u_j$$

$$v_{ij},\ w_{ij} \geq 0$$

la cual puede satisfacerse por:

$$v_{ij} = \text{Max } \{0,\ c_{ij} - u_i + u_j\}$$

$$w_{ij} = \text{Max } \{0,\ -(c_{ij} - u_i + u_j)\}$$

de donde el problema dual siempre tiene una solución factible para cualquier conjunto dado de u_i's. De hecho los valores anteriormente seleccionados de v_{ij} y w_{ij} nos da los valores óptimos de v_{ij} y w_{ij} para un conjunto fijo de u_i's.

2.5.3 Condiciones de holgura complementaria

Las condiciones de complementariedad por la optimalidad son:

$$(x_{ij} - a_{ij})\ v_{ij} = 0 \quad i,j=1,\ldots,m$$

$$(b_{ij} - x_{ij})\ w_{ij} = 0 \quad i,j=1,\ldots,m$$

Definamos $\bar{c}_{ij} = c_{ij} - u_i + u_j$. Entonces por la definición de v_{ij} y w_{ij} obtenemos:

$$v_{ij} = \text{Max } \{0, \bar{c}_{ij}\}$$

$$w_{ij} = \text{Max } \{0, -\bar{c}_{ij}\}$$

dado un conjunto de u_i's podemos calcular $c_{ij} = \bar{c}_{ij} - u_i + u_j$. Observamos que las condiciones de holgura complementaria se cumplen solamente si:

$$\bar{c}_{ij} < 0 \Rightarrow w_{ij} > 0 \Rightarrow x_{ij} = b_{ij} \qquad i,j=1,\ldots,m$$

$$\bar{c}_{ij} > 0 \Rightarrow v_{ij} > 0 \Rightarrow x_{ij} = a_{ij} \qquad i,j=1,\ldots,m$$

$$\bar{c}_{ij} = 0 \Rightarrow a_{ij} \leq x_{ij} \leq b_{ij} \qquad i,j,=1,\ldots,m$$

cualquier flujo que satisfaga las tres condiciones anteriores será óptimo. El problema entonces radica en investigar para cuales valores de las u_i's y x_{ij}'s las tres condiciones anteriores se satisfacen.

Arco Conformable y No Conformable

Un arco (i,j) se dice que esta en un estado "conformable", si cumple con las condiciones de optimalidad. Un arco (i,j) se dice que está en un estado no conformable, si no cumple con las condiciones de optimalidad.

Para hacer que un arco (i,j) no conformable, pase a un estado conformable, debemos incrementar la x_{ij} o disminuir el valor de $\bar{c}_{ij}$ cambiando las u_i's. Esto es lo que hace el algoritmo out-of-

kilter exactamente. Durante la fase primal del algoritmo modificamos las x_{ij}'s para que el arco pase a un estado conformable. Durante la fase dual modificamos las u_i's para alcanzar un estado conformable.

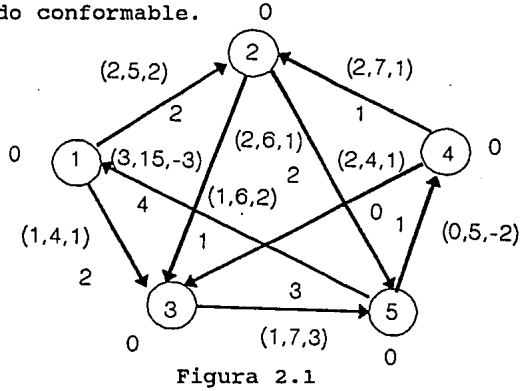


Figura 2.1

Consideremos la figura 2.1 , donde los números de los arcos indican respectivamente a_{ij}, b_{ij} y c_{ij}. El flujo inicial (arbitrario) en cada arco también se indica como el número sin paréntesis. Los precios duales asignados arbitrariamente a cada nodo, son todos cero y están indicados sobre el mismo nodo. Los costos negativos representan ganancias.

Se calcula $\bar{c}_{ij} = c_{ij}-u_i+u_j$, para cada arco (i,j). Como todas las $u_i=u_j=0$, en esta iteración $\bar{c}_{ij}=c_{ij}$ para todo arco (i,j). Los arcos (1,2), (2,3) y (2,5) están en orden pues satisfacen las condiciones siguientes:

$$\bar{c}_{ij} > 0 \quad \Rightarrow \quad x_{ij} = a_{ij}$$

$$\bar{c}_{ij} = 0 \quad \Rightarrow \quad a_{ij} \leq x_{ij} \leq b_{ij}$$

$$\bar{c}_{ij} < 0 \quad \Rightarrow \quad x_{ij} = b_{ij}$$

Así, por ejemplo, $C_{12} = 2 > 0$, lo que implica que el flujo X_{12} debe satisfacer $X_{12} = a_{12} = 2$. X_{12} está en orden, porque en efecto, $X_{12} = a_{12}$. $C_{23} = 2 > 0$, lo que implica que el flujo X_{23} debe satisfacer $X_{23} = a_{23} = 1$, X_{23} está en orden, porque $X_{23} = a_{23}$. $C_{25} = 1 > 0 \Rightarrow X_{25} = a_{25} = 2$, X_{25} está en orden es decir es conformable porque $X_{25} = a_{25}$.

Se analizará arbitrariamente el arco (1,3) que está fuera de orden es decir es no conformable, $c_{13} = 1 > 0$ lo que indica que X_{13} debería ser igual que $a_{13} = 1$. Pero $X_{13} = 2 > a_{13} = 1$, y por lo tanto, está fuera de orden es no conformable, se debe reducir el valor de X_{13} al valor $a_{13} = 1$. Esto es precisamente lo que intenta hacer el algoritmo Out-Of-Kilter.

Los estados conformables y no conformables para un arco en una red están dados en la siguiente forma:

	$\bar{c}_{ij} < 0$	$\bar{c}_{ij} = 0$	$\bar{c}_{ij} > 0$
$x_{ij} < a_{ij}$	no conformable	no conformable	no conformable
$x_{ij} = a_{ij}$	no conformable	conformable	conformable
$a_{ij} < x_{ij} < b_{ij}$	no conformable	conformable	no conformable
$x_{ij} = b_{ij}$	conformable	conformable	no conformable
$x_{ij} > b_{ij}$	no conformable	no conformable	no conformable

Figura 2.2

Note que un arco está en el estado conformable si $a_{ij} \leq x_{ij} \leq b_{ij}$, y las condiciones de complementariedad se cumplen.

Conforme vayamos modificando el flujo en un arco (i,j), el arco se mueve hacia arriba o hacia abajo de una columna particular de la figura anterior, dependiendo de si x_{ij} se incrementa o disminuye. Conforme modificamos los valores de las u_i's, el arco se mueve hacia la izquierda o hacia la derecha a través del renglón particular.

Una descripción gráfica de los estados de un arco se muestra a continuación.

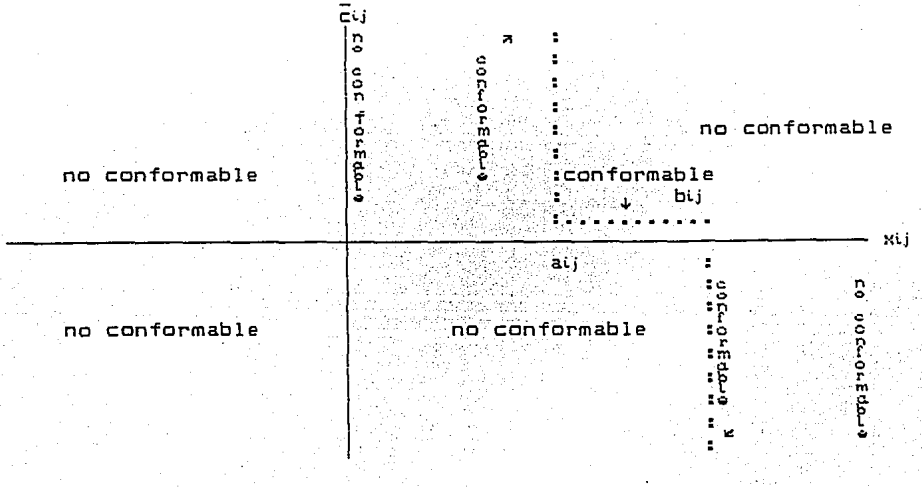


Figura 2.3

Con el objetivo de asegurarnos que el algoritmo convergerá, es necesaria alguna medida de la "distancia" para alcanzar la optimalidad; si podemos construir un algoritmo que reduzca periódicamente (en un número finito de iteraciones) la distancia para alcanzar la optimalidad, entonces el algoritmo eventualmente convergerá (realmente es necesario un argumento más fuerte acerca de la reducción de la distancia para alcanzar la optimalidad, pero como veremos, esa reducción es en números enteros por lo cual no habrá problema para que sea finita).

2.5.4 Números Kilter para un arco.

Hay muchas medidas diferentes de distancia para el problema out-of-kilter. En la siguiente figura se muestra una medida de distancia que llamaremos el número Kilter k_{ij}, para un arco (i,j).

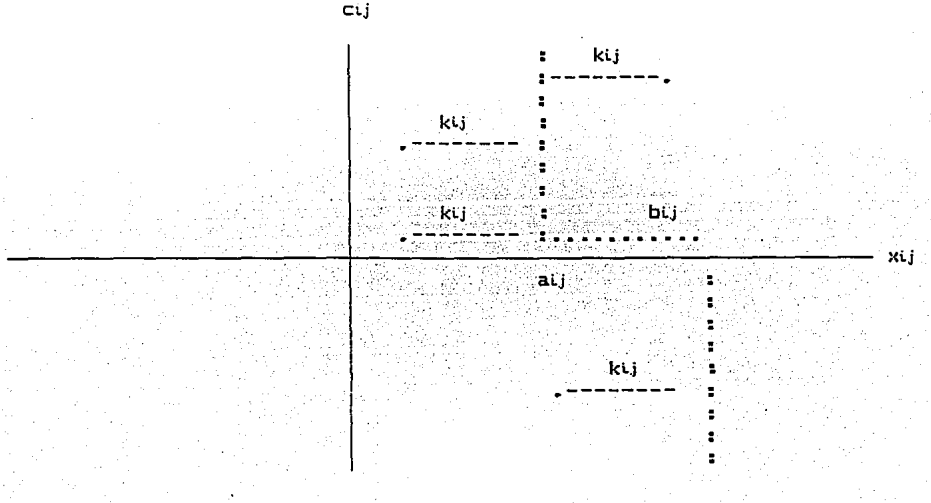


Figura 2.4

El **número de Kilter** se define como el mínimo cambio de flujo en el arco, que es necesario para llevarlo a un estado conformable. Note que puesto que todos los términos involucran valores absolutos, el número Kilter para un arco es no-negativo. También note que si el arco está en un estado conformable el número Kilter asociado es **cero**, y si el arco está en un estado no conformable, el número Kilter asociado es estrictamente positivo.

Note que si $\bar{c}_{ij} < 0$, entonces el arco (i,j) está en un estado conformable solamente si el flujo es igual a b_{ij}, y el número Kilter $| x_{ij} - b_{ij} |$ nos indica que tan lejos está el flujo x_{ij} del caso ideal b_{ij}. Similarmente si $\bar{c}_{ij} > 0$, entonces el número Kilter $| x_{ij} - a_{ij} |$ nos indica la distancia para alcanzar el flujo ideal a_{ij}. Finalmente, si $\bar{c}_{ij} = 0$, entonces el arco (i,j) esta en estado conformable si $a_{ij} \leq x_{ij} \leq b_{ij}$, en particular, si $x_{ij} > b_{ij}$, entonces el arco pasará a un estado conformable si el flujo disminuye en $| x_{ij} - b_{ij} |$, y si $x_{ij} < a_{ij}$, entonces el arco pasará al estado conformable si el flujo se aumenta en $|x_{ij} - a_{ij}|$.

En la siguiente figura se muestran las desviaciones (mínimo cambio de flujo requerido sobre el arco para que sea conformable) necesarias para que un arco (i,j) pase de un estado no conformable a un estado conformable.

	$\bar{c}_{ij} < 0$	$\bar{c}_{ij} = 0$	$\bar{c}_{ij} > 0$
$x_{ij} > a_{ij}$	$\lvert x_{ij} - b_{ij} \rvert$	$\lvert x_{ij} - a_{ij} \rvert$	$\lvert x_{ij} - a_{ij} \rvert$
$x_{ij} = a_{ij}$	$\lvert x_{ij} - b_{ij} \rvert$	0	0
$a_{ij} < x_{ij} < b_{ij}$	$\lvert x_{ij} - b_{ij} \rvert$	0	$\lvert x_{ij} - a_{ij} \rvert$
$x_{ij} = b_{ij}$	0	0	$\lvert x_{ij} - a_{ij} \rvert$
$x_{ij} < b_{ij}$	$\lvert x_{ij} - b_{ij} \rvert$	$\lvert x_{ij} - b_{ij} \rvert$	$\lvert x_{ij} - a_{ij} \rvert$

Figura 2.5

Un método para asegurar la convergencia finita del algoritmo out-of-kilter es demostrando lo siguiente:

- El número Kilter de un arco nunca se incrementa.
- En un número finito de iteraciones el número Kilter del arco es disminuido (en un número entero).

2.5.5 Estrategia del algoritmo out-of-Kilter

Como se mencionó anteriormente, el algoritmo out-of-Kilter se puede ver como una generalización del algoritmo primal-dual. En este aspecto los pasos generales del algoritmo son los siguientes:

Paso 1. Iniciar con un flujo tal como $x_{ij} = 0$, y una solución factible del dual, por ejemplo $u_i=0$. Identifique los estados y calcule los números Kilter.

Paso 2. Si la red tiene un arco no conformable, inicie la fase

primal del algoritmo, durante la cual se elige un arco no conformable y se trata de construir un nuevo flujo, en tal forma que el número Kilter de los demás arcos no se vea afectado (no aumente) y que el arco seleccionado sea mejorado (disminuya).

Paso 3. Cuando se determina que esa "mejora" no puede realizarse durante la fase primal, el algoritmo construye una nueva solución dual, de tal manera que el número Kilter no se vea afectado (no aumente), y se va al paso 2.

Paso 4. Iterando entre los pasos 2 y 3, el algoritmo eventualmente construye una solución óptima, o determina que no existe solución factible.

La fase primal: cambio de flujo.

Durante la fase primal, el algoritmo out-of-Kilter, intenta disminuir el número Kilter de un arco no conformable, cambiando o modificando el flujo, en tal forma que el número Kilter de los demás arcos no se vea afectado (no aumente).

Examinando la figura 2.3, observamos que el flujo debe ser modificado de tal manera que los estados no conformables se acerquen a los estados conformables. Por ejemplo, para el estado no conformable $x_{ij} > b_{ij}$ y $\bar{c}_{ij} < 0$, podemos disminuir x_{ij} tanto como $| x_{ij} - b_{ij} |$. Si disminuimos el flujo más allá de esa cantidad, el arco pasará a un estado no conformable (cosa que no queremos que suceda), también, no permitiremos ningún aumento de flujo en esta x_{ij}.

Un análisis similar de los otros estados conformables, nos proporcionan los siguientes resultados que aparecen en la figura 2.6.

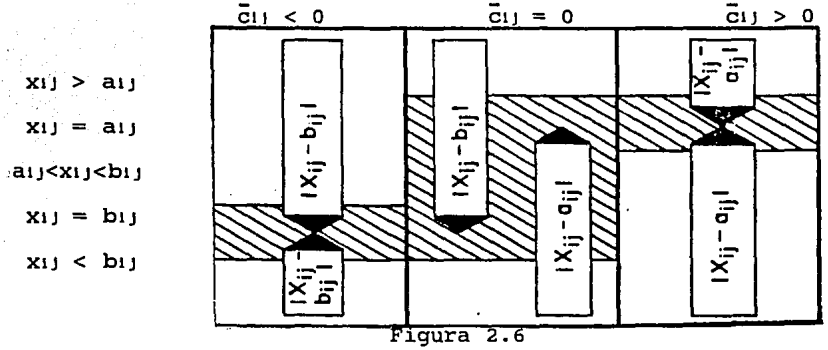


Figura 2.6

Varios estados de la figura anterior merecen una atención especial. El estado no conformable $x_{ij} > b_{ij}$ y $\bar{c}_{ij} = 0$, indica que el flujo deber de ser disminuído en $| x_{ij} - a_{ij} |$, refiriéndose a la figura 2.3, vemos que solamente necesitamos disminuir la x_{ij} en $| x_{ij} - b_{ij} |$, que es una cantidad menor, para alcanzar el estado conformable. Sin embargo como se puede ver en la misma figura, podemos continuar disminuyendo el flujo x_{ij} hasta en una cantidad $| x_{ij} - a_{ij} |$ y el arco permancerá en el estado conformable. Frecuentemente, es deseable hacer esto para ayudar a otros arcos a alcanzar el estado conformable.

También un arco en un estado conformable $a_{ij} < x_{ij} < b_{ij}$ y $\bar{c}_{ij} = 0$, puede aumentar o disminuir su flujo permaneciendo en el estado conformable. Esta situación se muestra en la siguiente figura.

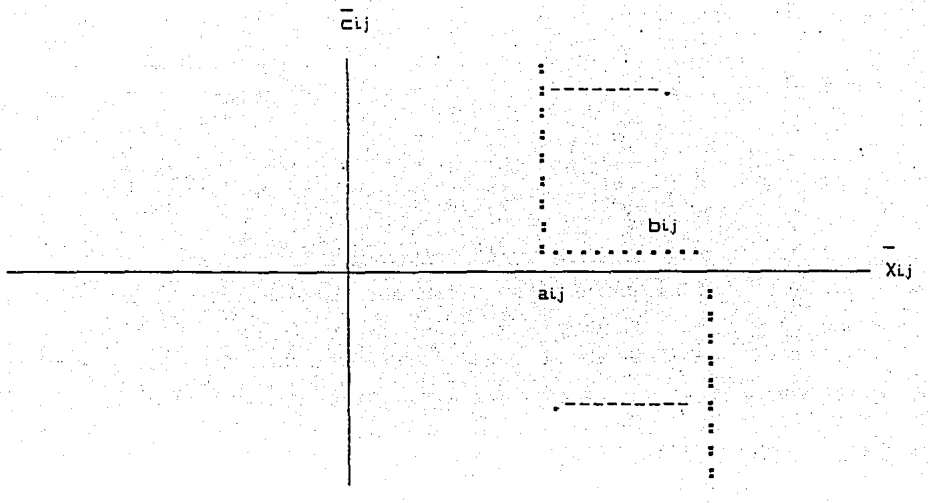


Figura 2.7

Hemos determinado cuánto podemos variar un determinado flujo, sobre un arco, debemos ahora determinar que combinación de flujos podemos hacer para que exista la "consevación de flujo".

Sea $\bar{x}$ el vector de flujos (actuales), entonces, la restricción de conservación flujo del problema primal, se puede escribir como: $A\bar{x} = 0$, donde A es la matriz de incidencia nodos-arcos. Si $\mathbb{D}$ es el vector de modificación de flujo, entonces debemos tener:

$$A(\bar{x} + \mathbb{D}) = 0 \quad \text{ó} \quad A\mathbb{D} = 0$$

Si $A\mathbb{D} = 0$, para $\mathbb{D} \neq 0$, entonces cada columna de A tiene exactamente un +1 y un -1 (Puesto que A es una matriz de incidencia nodo-arco), y los componentes diferentes de cero de $\mathbb{D}$ deben corresponder a un ciclo (no dirigido) o conjunto de ciclos. El flujo deberá ser modificado a través del ciclo o conjunto de ciclos, en tal forma que se satisfaga la restrición de "conservación de flujo".

Dado un arco no conformable, debemos construir un ciclo que contenga a este arco. Este ciclo debe tener la propiedad de que cuando se le de una orientación y cuando se adicione un flujo, ningún arco sea afectado (aumentado) en su número Kilter.

fase dual: cambio de variable dual.

Cuando no es posible construir un ciclo que contenga al arco no conformable específico, entonces debemos de modificar las $\bar{c}_{ij}$'s, en tal forma que ningún número Kilter sea afectado (aumentado), y que nos permita encontrar un ciclo que contenga al arco no conformable en cuestión.

Puesto que $\bar{c}_{ij} = c_{ij} - u_i + u_j$, debemos modificar las u_i's para poder modificar las $\bar{c}_{ij}$'s. Sea (p,q) un arco, no conformable, y sea $\bar{I}$ el conjunto de nodos que pueden ser alcanzados desde el nodo q a través de alguna trayectoria. Sea $I= N-\bar{I}$, donde $N=1,2,\ldots,n$. Note que ni $\bar{I}$ ni I, pueden ser vacios puesto que $q \in \bar{I}$ y $p \in I$ cuando pasamos a la fase dual. Queremos modificar las u_i's de tal forma que ningún número Kilter se vea afectado, y el conjunto $\bar{I}$ vaya creciendo periódicamente. Si otro nodo pasa a formar parte de $\bar{I}$, en un número finito de intervalos, entonces "p" pasará a formar parte de $\bar{I}$ y se creará un ciclo. Se

considera implícitamente que el conjunto $\bar{I}$ nunca se hará más pequeño. Para asegurar que esto sucede, debemos modificar las u_i's en tal forma que todos los arcos con ambas terminaciones en $\bar{I}$ sean retenidos.

Considere $\bar{c}_{ij} = c_{ij} - u_i + u_j$. Si u_i y u_j son modificadas en la misma cantidad, entonces $\bar{c}_{ij}$, permanece sin cambiar. Entonces podemos asegurar, que el conjunto $\bar{I}$, contendrá al menos todos los mismos nodos después del cambio en la variable dual, si cambiamos todas las u_i's en $\bar{I}$ en la misma cantidad θ. Suponga que dejamos las u_i's en I sin cambiar. Entonces los únicos arcos que serán afectados, serán los arcos de $\bar{I}$ a I y de I a $\bar{I}$. Específicamente, si $\theta > 0$ y modificamos las u_i's de acuerdo a:

$$u_i = \begin{cases} u_i + \theta & i \in \bar{I} \\ u_i & i \in I \end{cases}$$

entonces:

$$\bar{c}'_{ij} = \bar{c}_{ij} \quad \text{si } i \in \bar{I},\ j \in \bar{I}$$

ó

$$i \in I,\ j \in I$$

Ahora, si $i \in \bar{I}$ y $j \in I$ obtenemos:

$$\bar{c}'_{ij} = c_{ij} - (u_i + \theta) + u_j = \bar{c}_{ij} - \theta$$

También; si $i \in I$ y $j \in \bar{I}$ obtenemos:

$$\bar{c}'_{ij} = c_{ij} - u_i + (u_j + \theta) = \bar{c}_{ij} + \theta$$

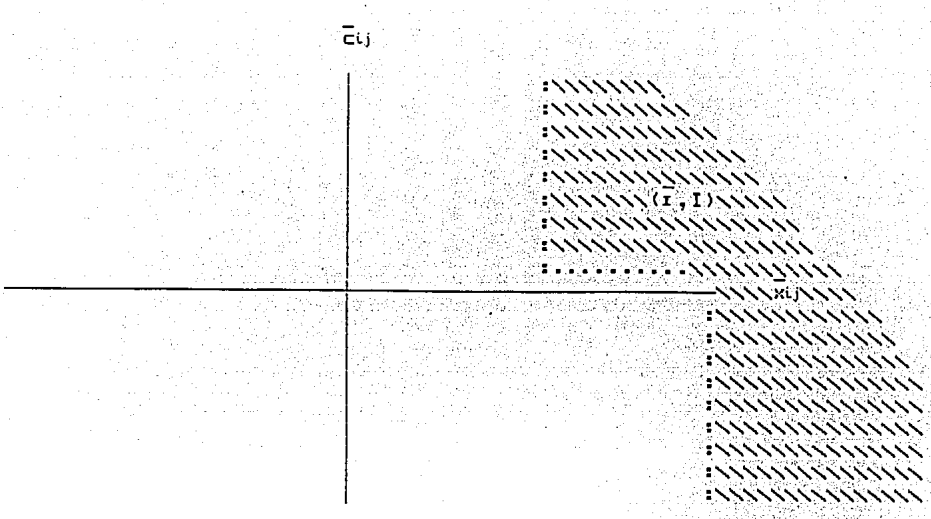
Entonces todos los arcos que van de $\bar{I}$ a I tendrán sus $\bar{c}_{ij}$'s disminuídas en θ, y aquellos arcos de I a $\bar{I}$, tendrán sus $\bar{c}_{ij}$'s incrementadas en θ.

Debemos determinar θ, en tal forma que el número Kilter de los demás arcos no se vea afectado (empeorado), y el estado de algunos arcos sea modificado. Primero debemos identificar los arcos que pueden estar en el conjunto $(\bar{I},I)$, y en el conjunto $(I,\bar{I})$.

(La notación (x,y) representa el conjunto $S=\{(x,y): x \in X, y \in Y\}$).

Examinando la figura 2.6, vemos que el conjunto $(\bar{I},I)$ no puede contener un arco asociado con el estado conformable $x_{ij} < a_{ij}$ y $\bar{c}_{ij} > 0$, puesto que tal arco podría formar parte de un ciclo, con el resultado de que si i puede ser alcanzado (a través de una trayectoria), desde q, entonces j, puede ser alcanzado desde q, y entonces $j \in \bar{I}$ (lo cual es una contradicción).

Examinando los estados conformables restantes, encontramos que los únicos candidatos para ser miembros de $(\bar{I},I)$, son aquellos que se muestran en la siguiente figura:



Poniendo estos estados en forma tabular:

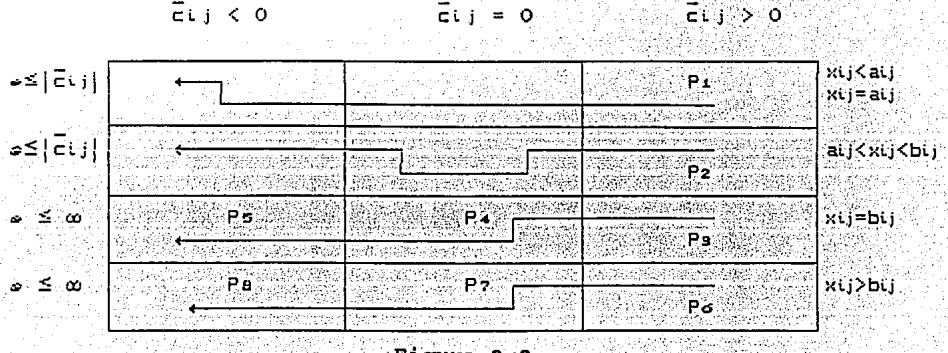


Figura 2.8

Recuerde que estos arcos de $\bar{I}$ a I, tendrán sus $\bar{c}_{ij}$'s disminuidas, entonces estos arcos cambiarán su estado de derecha a izquierda como se indica en la figura "2.8".

Examinando el arco $\bar{I}$ a I, que está en el estado $x_{ij} > b_{ij}$, $\bar{c}_{ij} > 0$ vemos en la figura 2.4, que mientras θ se incrementa, k_{ij} permanece constante y posteriormente k_{ij} disminuye desde $k_{ij}=| x_{ij}- a_{ij}|$ a $k_{ij} = | x_{ij} - b_{ij} |$, entonces para este arco podemos incrementar θ, tanto como queramos y el número de Kilter del arco nunca se incrementará. Para estos arcos podemos establecer un límite superior de ∞ para θ, como se indica en la figura "2.8".

Cualquier arco de $\bar{I}$ a I, que este en el estado $x_{ij} = b_{ij}$ y $c_{ij} > 0$ conforme θ se incrementa, k_{ij} disminuirá (a cero), y después su número Kilter permanecerá constante, nuevamente el límite superior de variación de θ, para asegurar que el número Kilter no se verá afectado, será de ∞. (∞ es un límite superior sobre el cambio permitido en θ para asegurar que ningún número de Kilter empeorará).

Sin embargo, examinando un arco de $\bar{I}$ a I, en el estado conformable $a_{ij}<x_{ij}<b_{ij}$, y $\bar{c}_{ij} > 0$, vemos que el número Kilter asociado Xij,primero disminuye (a cero), y luego empieza a incrementarse. Con el fin de eliminar el incremento en k_{ij} para el arco, debemos poner un límite de $| \bar{c}_{ij} |$ en θ. Similarmente debemos poner un límite $| \bar{c}_{ij} |$ en θ para los arcos en el estado $x_{ij}=a_{ij}$ y $\bar{c}_{ij} > 0$.

Este análisis justifica las entradas en la tabla de la figura 2.8.

En la siguiente figura se muestra el efecto del incremento en θ para los arcos que son miembros de $(\bar{I},I)$.

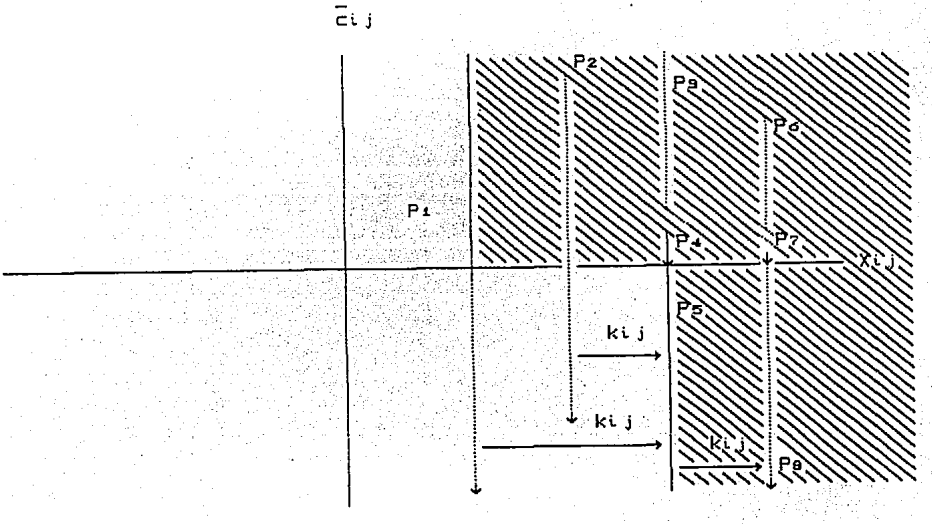


Figura 2.9

Un análisis similar de los arcos de I a $\bar{I}$, justifica la información en las siguientes tres figuras

ESTA TESIS NO DEBE
SALIR DE LA BIBLIOTECA

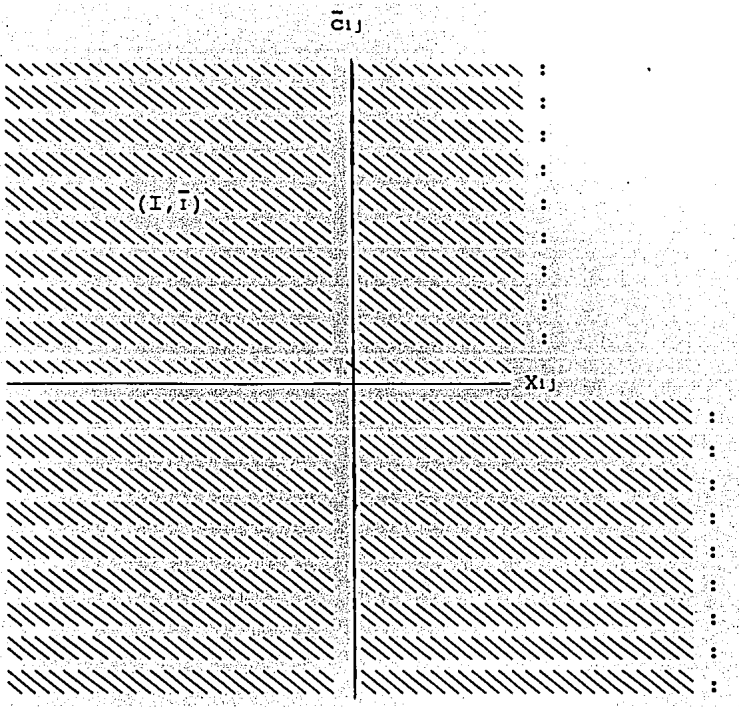


Figura 2.10

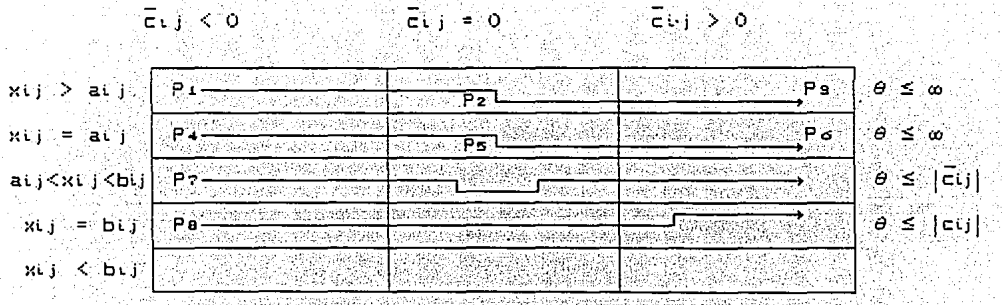


Figura 2.11

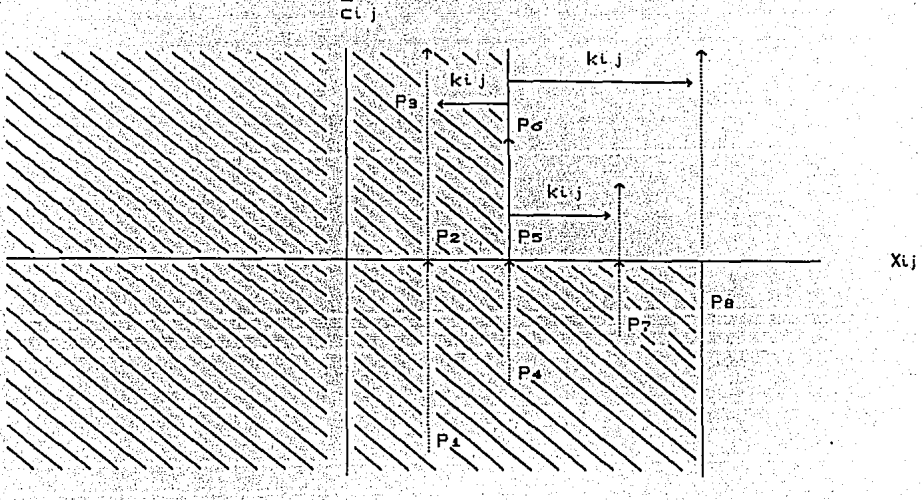


Figura 2.12

En cuanto a lo que concierne a la variación (aumento) de los números Kilter, las figuras 2.8, 2.10 y 2.12, indican que solo necesitamos calcular θ basado en los arcos de $\bar{I}$ a I con $x_{ij}<b_{ij}$ y los arcos de I a $\bar{I}$ con $x_{ij}>a_{ij}$. Sin embargo, si procedemos a definir un método de cálculo de θ, basado solamente en estas consideraciones, tendremos dificultades en la interpretación del significado del valor $\theta=\infty$. Esto se simplifica, si en vez de desigualdades estrictas de flujo (esto es, $x_{ij}<b_{ij}$, $x_{ij}>a_{ij}$), admitimos desigualdades débiles (esto es, $x_{ij}\leq b_{ij}$, $x_{ij}\geq a_{ij}$). La razón para tal desviación, parece ser contrario a la intuición, será clara cuando establezcamos la convergencia del algoritmo.

La discusión anterior concierne a los límites de θ basados en la consideración de los números de Kilter, y en las propiedades de convergencia (todavía no establecidas), lo que nos conduce a el siguiente procedimiento formal para el cálculo de θ.

Definamos S_1 y S_2 por:

$$S_1 = \{ (i,j) : i\in\bar{I},\ j\in I,\ \bar{c}_{ij}>0,\ x_i\leq b_{ij}\}$$

$$S_2 = \{ (i,j) : i\in I,\ j\in\bar{I},\ c_{ij}<0,\ x_{ij}\geq a_{ij}\}$$

Sea

$$\theta_1 = \underset{(i,j)\in S_1}{\text{Min}} \{ |\bar{c}_{ij}| \}$$

$$\theta_2 = \underset{(i,j)\in S_2}{\text{Min}} \{ |\bar{c}_{ij}| \}$$

$$\theta = \text{Min } \{\theta_1, \theta_2\}$$

donde $\theta=\infty$ si S_1 es vacío. Entonces θ es estrictamente positivo. También, θ es un entero positivo o ∞.

Estas dos posibilidades se discuten brevemente a continuación:

CASO 1: $0<\theta<\infty$

En este caso hacemos los cambios apropiados en u_i (esto es, $u_i'=u_i+\theta$ si $i\in\bar{I}$, y $u_i'=u_i$ si $i\in I$) y pasamos a la fase primal del algoritmo.

CASO 2: $\theta=\infty$

En este caso el problema primal no tiene solución factible, esto termina la fase dual del algoritmo out-of-Kilter, y proporciona el fundamento de la totalidad del algoritmo out-of-Kilter.

Como una ilustración consideremos el ejemplo de la figura 2.1. Aquí tenemos :

$$S_1 = \{(1,2),(1,3)\}$$
$$S_2 = \{(5,1)\}$$
$$\theta_1 = \text{Mín } |2,1| = 1$$
$$\theta_2 = 3$$
$$\theta = \text{Mín } |1,3| = 1$$

Esto da origen al siguiente cambio en las variables duales:

$$u_2' = u_2 + \theta = 1$$
$$u_3' = u_3 + \theta = 1$$

$$u'_4 = u_4 + \theta = 1$$
$$u'_5 = u_5 + \theta = 1.$$

2.5.6 La no-factibilidad del problema cuando $\theta=\infty$.

Supongamos que durante alguna aplicación de la fase dual del algoritmo out-of-Kilter, tenemos el caso donde $\theta=\infty$. Cuando esto ocurre, debemos tener $S_1=S_2=\phi$. Puesto que $S_1 = \phi$, revisando la definición de S_1, concluimos que $i\in\bar{I}$, y $j\in I$, lo que implica uno de los siguientes casos:

i) $\bar{c}_{ij} > 0$ y $x_{ij} > b_{ij}$

ii) $\bar{c}_{ij} = 0$

iii) $\bar{c}_{ij} < 0$

De la figura "2.8", y puesto que $i\in\bar{I}$ y $j\in I$, posiblemente ii) o iii) pueden suceder solamente si $x_{ij} \geq b_{ij}$. Entonces $S_1=\phi$ puede suceder solamente si $X_{ij}=b_{ij}$, para $i\in\bar{I}$ y $j\in I$.

Similarmente, $S_2 = \phi$ solamente si $i\in I$ y $j\in\bar{I}$, implica que $x_{ij} \leq a_{ij}$ Entonces $S_1=S_2=\phi$ implica:

$$x_{ij} \geq b_{ij} \text{ si } i\in\bar{I} \text{ y } j\in I \quad . \quad . \quad . \quad (1)$$

y

$$x_{ij} \leq a_{ij} \text{ si } i\in I, \text{ y } j\in\bar{I} \quad . \quad . \quad . \quad (2)$$

Sumando estas desigualdades obtenemos:

$$\sum_{\substack{i\in\bar{I}\\ j\in I}} x_{ij} - \sum_{\substack{i\in I\\ j\in\bar{I}}} x_{ij} > \sum_{\substack{i\in\bar{I}\\ j\in I}} b_{ij} - \sum_{\substack{i\in I\\ j\in\bar{I}}} a_{ij} \quad . \quad . \quad . \quad (3)$$

Puesto que el flujo dado por x_{ij}'s es "conservatorio", y notando que el conjunto de nodos consistente de $\bar{I} \cup I$ y $\bar{I} \cap I = \phi$, entonces la ecuación de conservación se puede escribir como:

$$\sum_{j\in\bar{I}} x_{ij} + \sum_{j\in I} x_{ij} - \sum_{j\in\bar{I}} x_{ji} - \sum_{j\in I} x_{ji} = 0 \qquad i=1,\ldots,m$$

sumando estas ecuaciones sobre $i\in\bar{I}$ obtenemos:

$$\sum_{\substack{i\in\bar{I}\\ j\in\bar{I}}} x_{ij} + \sum_{\substack{i\in\bar{I}\\ j\in I}} x_{ij} - \sum_{\substack{j\in\bar{I}\\ i\in\bar{I}}} x_{ji} - \sum_{\substack{i\in\bar{I}\\ j\in I}} x_{ji} = 0$$

Notando que: $\sum_{\substack{i\in\bar{I}\\ j\in\bar{I}}} x_{ij} = \sum_{\substack{j\in\bar{I}\\ i\in\bar{I}}} x_{ji}$, y que $\sum_{\substack{i\in\bar{I}\\ j\in I}} x_{ji} = \sum_{\substack{i\in I\\ j\in\bar{I}}} x_{ij}$, la

ecuación se reduce a:

$$\sum_{\substack{i\in\bar{I}\\ j\in I}} x_{ij} + \sum_{\substack{i\in I\\ j\in\bar{I}}} x_{ij} = 0 \qquad \ldots (3')$$

sustituyendo en la desigualdad (3) obtenemos:

$$0 > \sum_{\substack{i\in\bar{I}\\ j\in I}} b_{ij} - \sum_{\substack{i\in I\\ j\in\bar{I}}} a_{ij} \qquad \ldots (4)$$

Supongamos por contradicción que hay un flujo factible representado por $\hat{x}_{ij}$ para $i,j,=1,\ldots,m$. Entonces $b_{ij} \geq x_{ij}$ y $-a_{ij} \geq -\hat{x}_{ij}$; la desigualdad (4) nos da:

$$0 > \sum_{\substack{i\in\bar{I}\\ j\in I}} b_{ij} - \sum_{\substack{i\in I\\ j\in\bar{I}}} a_{ij} \geq \sum_{\substack{i\in\bar{I}\\ j\in I}} \hat{x}_{ij} - \sum_{\substack{i\in I\\ j\in\bar{I}}} \hat{x}_{ij} \qquad \ldots (5)$$

pero como el flujo $\hat{x}_{ij}$ representa un flujo factible, debe ser conservativo.

En forma similar a la ecuación (3'), es claro que el miembro derecho de la desigualdad (5) debe ser igual a cero. Entonces (5) implica que 0>0, lo cual es imposible. Esta contradicción demuestra que si $\theta=\infty$ no puede haber flujo factible.

Nota: Si S_1 y S_2 se hubieran definido mediante desigualdades estrictas para X ($X_{ij} < b_{ij}$ y $X_{ij} > a_{ij}$ respectivamente), no hubiera podido producirse la desigualdad requerida en (3).

2.5.7 Convergencia del algoritmo out-of-Kilter

Para el propósito del siguiente argumento de convergencia finita hacemos la consideración de que los vectores c,b,y a tienen valores enteros.

Hay varias propiedades del algoritmo, que deben ser mencionadas:

- Primero, cada vez que un circuito es construido conteniendo un arco no conformable, el número Kilter de ese arco y de los demás arcos, se reducen en un número entero. Solamente podemos construir un número finito de circuitos, que contengan arcos no conformables, antes de alcanzar la solución óptima.

- Segundo, después de cada cambio de variable dual, el estado de cada arco, que tiene ambas terminales en $\bar{T}$, permanece sin cambio. Si (p,q), es no conformable, entonces, después de un cambio de variable dual, cada nodo en $\bar{T}$, está todavía en $\bar{T}$.

Existen dos posibilidades:

Una posibilidad es que, un nuevo nodo k, entre al conjunto $\bar{I}$, cada vez que esto ocurra, el conjunto $\bar{I}$ crece, para al menos un nodo. Esto puede ocurrir, a lo mucho en un número finito de veces, antes de que el nodo "p", pase a ser miembro de $\bar{I}$, y se obtenga un circuito que contenga a (p,q). Entonces, si el algoritmo no es finito, debe de existir la posibilidad, de que en un número infinito de cambios de variable dual, el conjunto $\bar{I}$ no se incremente, ó $\theta=\infty$. Mostraremos, que esto no puede ocurrir.

Supongamos que después de un cambio de variable dual, no hay nuevos nodos que pasen a ser miembros de $\bar{I}$; esto es, $\bar{I}$ no se incrementa. Entonces al pasar a la siguiente fase dual, tenemos los mismos conjuntos $\bar{I}$, I, y las mismas x_{ij}'s. Además cada arco de $\bar{I}$ a I, ha incrementado su $\bar{c}_{ij}$, cada arco de I a $\bar{I}$, ha disminuido su $\bar{c}_{ij}$. Entonces, después del cambio de variable dual, los nuevos conjuntos S_1' y S_2', satisfacen:

$$S_1' \subset S_1 \quad y \quad S_2' \subset S_2 \qquad . \quad . \quad . \quad (1)$$

Seleccionando un valor de θ (finito), al menos un arco ha sido sacado del conjunto S_1 o de S_2. Entonces alguna de las "inclusiones" anteriores es propia.

Ahora; si S_1 y S_2 pudieran decrecer a lo más de un número finito de veces antes de que $S_1 \cup S_2 = \phi$ y $\theta = \infty$, en cuyo caso el algoritmo termina. Esto completa el argumento de que al algoritmo out-of-Kilter es finito.

2.5.8 Resumen del algoritmo out-of-Kilter

Paso 0. Empiece con un flujo circulatorio $x=(x_{ij})$ y un vector de potenciales $u=(u_i)$ (arbitrarios).

Paso 1. Si existe un arco (s,t) no conformable, ir al paso 2; de otra manera, el flujo circulatorio es factible y óptimo.

Paso 2. Aplique el método de las etiquetas para formar un circuito que contenga a (s,t), donde t es el origen y s es el destino, si el arco (s,t) está en uno de los estados no conformables;

$$\bar{c}_{ij} > 0 \; ; \; x_{ij} < a_{ij}$$

$$\bar{c}_{ij} = 0 \; ; \; x_{ij} < a_{ij}$$

$$\bar{c}_{ij} < 0 \; , \; x_{ij} < b_{ij}$$

de otra manera, sea s el origen y t el destino.

Paso 3.(modificación del flujo). Existe un ciclo que contiene a (s,t).

Aumenta en α el flujo en (s,t) donde α es

min $[\alpha_s, a_{st}-x_{st}]$ si (s,t) esta en el estado $\bar{c}_{ij}>0$; $x_{ij}<a_{ij}$.
min $[\alpha_s, b_{st}-x_{st}]$ si (s,t) esta en el estado $\bar{c}_{ij}=0$; $x_{ij}<a_{ij}$, o en el estado $\bar{c}_{ij}<0$, $x_{ij}<b_{ij}$.

min $[\alpha_s, x_{st}-a_{st}]$ si (s,t) esta en el estado $\bar{c}_{ij}>0$; $x_{ij}>a_{ij}$, o en el estado $\bar{c}_{ij}=0$; $x_{ij}>b_{ij}$.

min $[\alpha_s, x_{st}-b_{st}]$ si (s,t) esta en el estado $\bar{c}_{ij}<0$; $x_{ij}>b_{ij}$.

Si el arco (s,t) es conformable, regrese a 1. De otra manera, borre las etiquetas y empiece nuevamente en 2.

Método de las etiquetas

Paso 0. Denote con k el origen y con p el destino. Se desea encontrar un ciclo que contenga (k,p). Etiquete el nodo k con $(-,\infty)$.

Paso 1. Sea i un nodo con etiqueta (h,α_i). Etiquete con (i,α_j) cada nodo j que satisfaga alguna de las condiciones.

λ; $(i,j)\in A$, $\bar{c}_{ij} > 0$; $x_{ij} < a_{ij}$

μ; $(i,j)\in A$, $\bar{c}_{ij} \leq 0$; $x_{ij} < b_{ij}$

γ; $(j,i)\in A$, $\bar{c}_{ij} \geq 0$; $x_{ji} > a_{ji}$

ρ; $(j,i)\in A$, $c_{ij} < 0$; $x_{ji} > b_{ji}$

donde α_j es igual a $\min[\alpha_i, a_{ij} - x_{ij}]$, $\min[\alpha_i, b_{ij} - x_{ij}]$, $\min[\alpha_i, x_{ji} - a_{ji}]$, $\min[\alpha_i, x_{ji} - b_{ji}]$, en cada uno de los respectivos casos λ, μ, γ y ρ.

Paso 2. Si el destino p es etiquetado, considere la trayectoria de k a p en que cada nodo es la etiqueta del siguiente. Aumente,

en α_p el flujo en la trayectoria de k a p; específicamente, aumente (disminuya) en α_p el flujo en un arco (i,j) si el arco está en el sentido opuesto de la circulación del flujo en el arco (k,p).

Paso 3. Si el destino no es etiquetado, termine.

Método de cambio de potencial

Denote por $\bar{I}$ y I ; es conjunto de nodos etiquetados y no etiquetados al final del método de las etiquetas. Sean los conjuntos:

$$S_1 = \{(i,j) \quad ; \quad i \in \bar{I}, \quad j \in I, \; c_{ij} > 0, \; x_{ij} \leq b_{ij}\}$$

$$S_2 = \{(i,j) \quad ; \quad i \in I, \quad j \in \bar{I}, \; c_{ij} < 0, \; x_{ij} \geq a_{ij}\}$$

Si $S_1 = S_2 = \phi$ hagamos $\theta = +\infty$. Asimismo termine, pues el problema no tiene solución factible. De otra manera, sea

$$\theta = \min \left[\min_{S_1} (\bar{c}_{ij}), \min_{S_2} (-\bar{c}_{ij}) \right]$$

que es un número finito. Reemplace el vector de potenciales u por un nuevo vector u' donde

$$u_i' = u_i \quad \text{si} \quad i \in I \quad ; \quad u_i' = u_i + \theta \text{ si } i \in \bar{I}$$

Es conveniente puntualizar que en este caso, la correspondiente matriz de costos relativos es dada por

$$\bar{c}_{ij}' = \begin{cases} \bar{c}_{ij} - \theta & \text{si } (i,j) \in (\bar{I},I) \\ \bar{c}_{ij} + \theta & \text{si } (i,j) \in (I,\bar{I}) \\ \bar{c}_{ij} & \text{de otra manera} \end{cases}$$

CAPITULO III

PROBLEMAS Y APLICACIONES

los problemas de redes surgen en una gran variedad de situaciones. La representación de redes se utiliza ampliamente en áreas tan diversas como campos de distribución de recursos, secuenciación, inventarios, asignación de recursos, localización de instalaciones, planeación de producción, planeación financiera, diseño de rutas de vehículos, por citar algunos. En general una representación de redes nos permite visualizar las relaciones entre los componentes de los sistemas que se usa casi en todas las áreas científicas, sociales y económicas.

Gran variedad de problemas de la programación lineal pueden ser formulados mediante lo que denominaremos **estructuras de transporte.** Estas estructuras pueden subdividirse en varias subcategorías las cuales provienen de alguna modificación, a veces elemental de la estructura de transporte. Estas subcategorías son: las estructuras de transbordo o transporte con nodos intermedios, las estructuras de asignación, estructura de transporte con capacidad limitada y estructuras de transporte generalizadas.

Todas estas estructuras pueden ser resueltas por el método simplex, aunque esto podría resultar ineficiente, debido a que para cada caso particular han sido desarrollados algoritmos que justifican métodos de solución más eficientes.

- Un ejemplo representativo sobre problemas de redes de optimización ha sido la que se llevó a cabo con la Compañía "Citgo", cuyo giro es el refinamiento y comercialización de petróleo. En 1983 la Southland Corporation adquirió "Citgo" la administración de esta compañía visualizó la necesidad de crear

un sistema de modelado para ayudar a "Citgo" a superar las presiones de los precios cambiantes de petróleo crudo y un aumento de 30 veces los costos de capital de trabajo.

El equipo de investigación de la compañia desarrolló mediante el uso de redes un sistema para apoyar decisiones; el modelo utilizado para cada producto de la compañía "Citgo" fue el problema de flujos restringidos a costo mínimo, analizado en el capítulo anterior. Cada modelo tiene cerca de 3000 ecuaciones (nodos) y 15,000 variables (arcos), lo cual representa un problema de tamaño medio para los estándares actuales de aplicación de los modelos de optimización de redes.

El sistema toma en cuenta todos los aspectos del negocio, además ayuda a la administración en todas las decisiones, esto es desde la producción en refinerías hasta los precios que debe pagar o cobrar. Fue esencial utilizar la representación de redes, debido a el flujo de bienes en actividades tales como la compra de petróleo crudo de los proveedores, el embarque a las refinerías, el refinamiento de los diferentes productos y el embarque de estos productos a los centros de distribución y terminales de almacenamiento para su venta posterior.

Este sistema ha permitido a la compañía reducir su inventario en $116 millones de dólares, lo cual significa un ahorro en los intereses anuales de $14 millones de dólares, así como mejoras en la toma de decisiones de lo que es la coordinación, costeo y compra lo cual equivale a $2.5 millones de dólares anuales.

El capítulo se desarrolla como sigue; primeramente se mostrará la forma matemática de transformar el problema Ax=d a la forma Ax=0, conduciendonos así de una manera natural a la solución del problema del transporte con el algoritmo Out-of-Kilter. Se

muestra también la solución de un problema prototipo de transporte. Cabe mencionar que este algoritmo proporcionará la solución del problema del transporte, pero también existen además otros algoritmos que lo resuelven eficientemente, como el algoritmo Stepping Stone de Dantzig, y que se han desarrollado métodos, como el de la esquina noroeste y el método de Vogel, que proporcionan una solución factible del problema del transporte.

Otro grupo de problemas de la programción lineal, que pueden ser resueltos por el algoritmo Out-Of-Kilter, son los modelos asociados a las redes de optimización, estos modelos incluyen: problemas de flujo máximo, problemas de ruta corta (o más larga), ruta crítica, problemas de flujo restringido en una red con costo mínimo, redes de actividad, etc.

Se mostrará los problemas de flujo máximo y ruta más corta (o más larga), y se ilustra su método de solución con el algoritmo Out-Of-Kilter. Además se resuelven problemas utilizando ruta crítica dandoles solución mediante un programa elaborado en pascal.

3.1 Transformación del Problema Ax=d a la forma Ax=0

Cualquier problema de flujo en redes con costo mínimo, puede ser transformado a la forma Out-Of-Kilter.

Sea el problema:

$$\text{Minimice } Cx$$

$$\text{S.A.} \qquad Ax=d$$

$$a \le x \le b$$

donde A es la matriz de incidencia nodos-arcos. Si $d_i>0$.

Defina una variable $X_{m+1,i}$, tal que, $X_{m+1,i}=d_i$. Para asegurar que $X_{m+1,i}=d_i$, hacemos $a_{m+1,i}=b_{m+1,i}=d_i$.

Similarmente, si $d_i<0$, entonces definamos una variable $X_{i,m+1}$ tal que $X_{i,m+1}=-d_i$. Para asegurar esto, hacemos $a_{i,m+1}=b_{i,m+1}=-d_i$. Entonces:

$$\sum_{j=1}^{m} X_{ij} - \sum_{j=1}^{m} X_{ij} = X_{m+1,i} \quad \text{si } d_i>0$$

$$\sum_{j=1}^{m} X_{ij} - \sum_{j=1}^{m} X_{ij} = -X_{i,m+1} \quad \text{si } d_i<0$$

Las variables $X_{i,m+1}$, $X_{m+1,i}$, se pueden interpretar como un flujo desde el nodo i, al nuevo nodo m+1, y un flujo desde el nodo m+1, al nodo i respectivamente.

Igualando a cero las restricciones, obtenemos:

$$\sum_{j=1}^{m} X_{ij} - \sum_{j=1}^{m+1} X_{ji} = 0 \quad \text{si } d_i > 0$$

$$\sum_{j=1}^{m+1} X_{ij} - \sum_{j=1}^{m} X_{ij} = 0 \quad \text{si } d_i < 0$$

Sumando ambas ecuaciones, obtenemos:

$$\sum_{j=1}^{m} X_{m+1,j} - \sum_{j=1}^{m} X_{j,m+1} = 0$$

con lo que hemos transformado el problema a la forma circulatoria.

3.2 El Problema del Transporte

Una clase importante de problemas de programación lineal son los problemas de transporte. Esta clase y su extensión, la clase de los problemas de flujo en redes, poseen una estructura especial que 1) permite el desarrollo de algoritmos simples y eficientes 2) facilita una mayor intuición y un mejor entendimiento de las técnicas de programación lineal y del método simplex.

Ejemplos de este tipo de problemas se presentan en áreas de planeación y programación de la producción.. En lo que a producción se refiere es necesario manejar una gran cantidad de planeación con el fin de compaginar la capacidad del proceso de

producción con la demanda del producto final. Un ejemplo en el cual se presenta esta situación es en la producción de juegos para televisión, en el cual la demanda de juegos de video se presenta durante las ventas de navidad, período en el cual se rebasa la capacidad de producción. Para resolver este problema es necesario utilizar inventarios, considerados como forma de almacenamiento de capacidad de producción. En general existen costos asociados con el mantenimiento de inventarios, que resultan de inversiones de capital, costos de almacenamiento, costos por deterioro, etc. Los costos de mantenimiento podemos denotarlos como porcentajes de los costos de producción o como costos unitarios durante los lapsos de tiempo en que se mantenga o conserve el inventario. El problema ahora es cómo utilizar la capacidad disponible para satisfacer las demandas presente y futura a un costo mínimo, considerando también los costos de producción y los de mantenimiento.

Podemos plantear el problema como uno de transporte si lo consideramos como un problema de transportar la producción, en donde las capacidades de producción en cada período de tiempo se convierten en las ofertas, y los requerimientos por período se convierten en las demandas.

Definición del problema de transporte

Conside m puntos origen localizados en un mapa, donde el origen i tiene una oferta a_i unidades de un producto particular. Además, existen n destinos, donde el destino j tiene una demanda de b_j unidades del producto. Asociado a cada arco (i,j), hay un costo unitario c_{ij} de transporte del producto. El problema, es determinar la distribución del producto, en tal forma que satisfaga la demanda y se minimice el costo total de transporte.

Sea x_{ij} el número de unidades transportadas a lo largo del arco (i,j), del orígen i al destino j. Supóngase, además que la oferta total es igual a la demanda total, es decir

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$$

Si la oferta total excede a la demanda total, entonces se puede introducir un destino ficticio o artificial con demanda $b_{n+1}=\sum_i a_i-\sum_j b_j$, y $C_{i,n+1} = 0$ para $i=1,\ldots,m$. Suponiendo que la oferta total es igual a la demanda total, el modelo de programación lineal para el problema de transporte resulta ser el siguiente:

Minimizar

$$Z = C_{11}X_{11}+ \ldots +C_{1n}X_{1n}+C_{21}X_{21}+ \ldots +C_{2n}X_{2n}+ \ldots +C_{m1}X_{m1}+C_{mn}X_{mn}$$

Sujeta a:

$$\begin{aligned}
&X_{11} +\ldots+ X_{1n} && && && = a_1\\
& && X_{21} +\ldots+ X_{2n} && && = a_2\\
& && && X_{m1} +\ldots+ X_{mn} && = a_m\\
&X_{11} + && X_{21} && \ldots && + X_{m1} && = b_1\\
& X_{1n} && + && X_{2n} && && + X_{mn} && = b_n
\end{aligned}$$

$$X_{11}, \ldots, X_{1n}, X_{21}, \ldots, X_{2n}, \ldots, X_{m1}, \ldots, X_{mn} \geq 0$$

El problema de transporte se puede escribir en forma matricial si se hace

$$X = (X_{11}, X_{12}, \ldots, X_{1n}, X_{21}, \ldots, X_{2n}, \ldots, X_{mn})^T$$
$$C = (C_{11}, C_{12}, \ldots, C_{1n}, C_{21}, \ldots, C_{2n}, \ldots, C_{mn})$$
$$b = (a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n)^T$$
$$A = (a_{11}, a_{12}, \ldots, a_{1n}, a_{21}, \ldots, a_{2n}, \ldots, a_{mn})$$

en donde

$$a_{ij} = e_i + e_{m+j}$$

y e_i y e_{m+j} son vectores unitarios en E^{M+N} con unos en la i-ésima y la (m+j)-ésima posición respectivamente; a_i es un escalar, a_{ij} es un vector, con estas definiciones el problema será:

Minimice cx

S.A.

$$Ax = b$$
$$x \geq 0$$

La matriz A, con dimensión (m+n)*mn, tiene la siguiente forma especial.

$$A = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \vdots & . & & . \\ 0 & 0 & & 1 \\ I & I & \ldots & I \end{bmatrix}$$

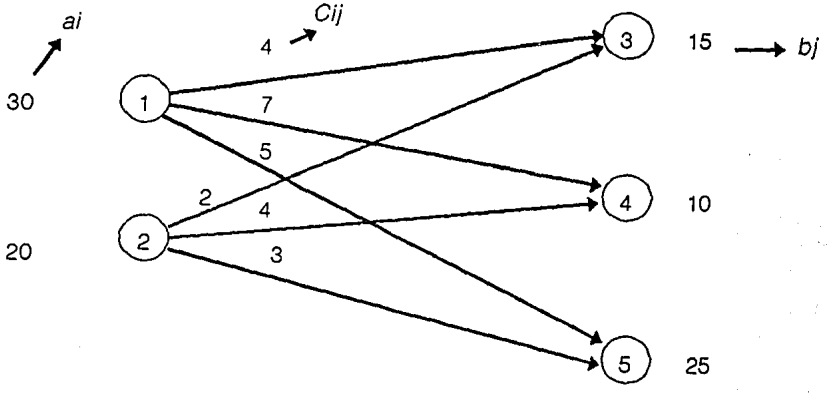
en donde 1 es un vector n-vector renglón de componentes todas iguales a 1, e I es una matriz identidad nxn. La matriz A es la que da al problema de transporte su estructura especial.

Ejemplo prototipo:

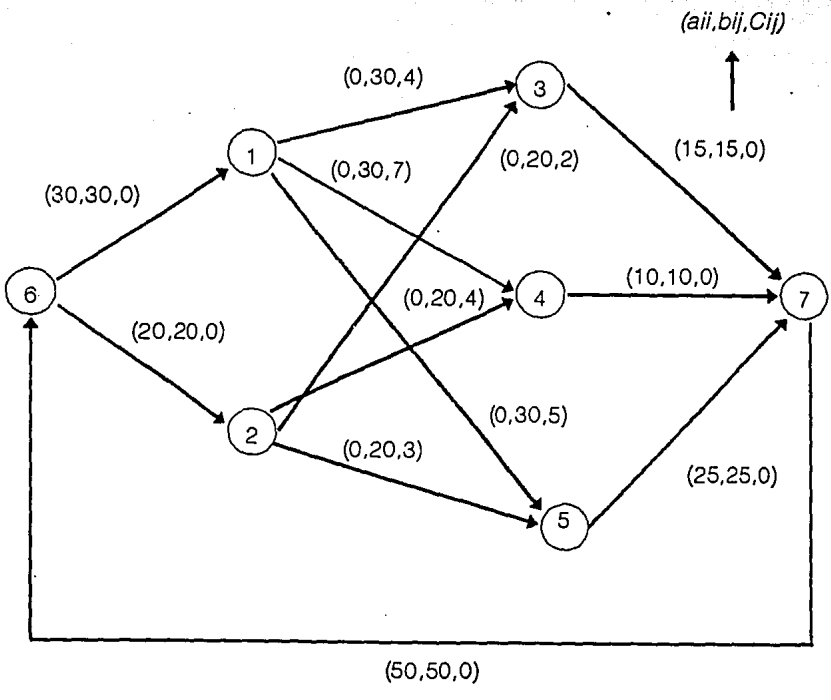
Como ejemplo del problema de transporte considere dos orígenes y tres destinos. Los datos aparecen a continuación:

ORIGEN		DESTINO 1	2	3	ai
	1	$C_{11}=4$	$C_{12}=7$	$C_{13}=5$	30
	2	$C_{21}=2$	$C_{22}=4$	$C_{23}=3$	20
	bj	15	10	25	

El problema presentado en forma de red, es como sigue:



El problema puede ser resuelto con el algoritmo Out-Of-Kilter introduciendo las siguientes modificaciones en la red:



Cabe mencionar que el algoritmo "Out-Of-Kilter" proporciona la solución del problema de transporte pero existen además otros algoritmos que lo resuelven algunos son: Stepping-stone (Piedra de Paso), Solución Numérica de Houthakker, Método de Wagener, Primal-Dual y el Primal de Balinsky y Gomory. Se presenta el algoritmo de Piedra de Paso en la rutina A-III (Anexos). También se han desarrollado métodos como Regla del Extremo Noroeste,Columna Mínimo, Renglón Mínimo, Matriz Miníma, Métodos de Vogel y Algoritmo de Russell que proporcionan una solución

básica factible inicial del problema del transporte. De estos algoritmos el que proporciona una mejor solución es el Método de Vogel, el cual se presenta en la rutina A-IV (Anexos)

3.3 El Problema de Asignación

El problema de asignación, es un tipo especial de problema de la programación lineal, en donde; se trata de determinar la mejor manera de asignar una serie de recursos a un grupo de actividades que proporcionan diferentes valores. Los problemas típicos de esta naturaleza incluyen asignar trabajadores a máquinas, equipos de trabajo a proyectos, etc. En estos problemas las variables de decisión X_{ij} representan la asignación del recurso i-ésimo a la actividad j-ésima, cuyos únicos valores son 0 ó 1; hay un costo asociado c_{ij} a cada asignación. La función objetivo será determinar, como deben ser hechas todas las asignaciones, para minimizar el costo total.

La formulación del problema de asignación es como sigue:

$$\text{Minimice} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} X_{ij}$$

$$\text{S.A.} \quad \sum_{j=1}^{n} X_{ij} = 1 \qquad i=1,\ldots,m$$

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

$$X_{ij} \geq 0 \qquad i=1,\ldots,m \quad j=1,\ldots,n$$

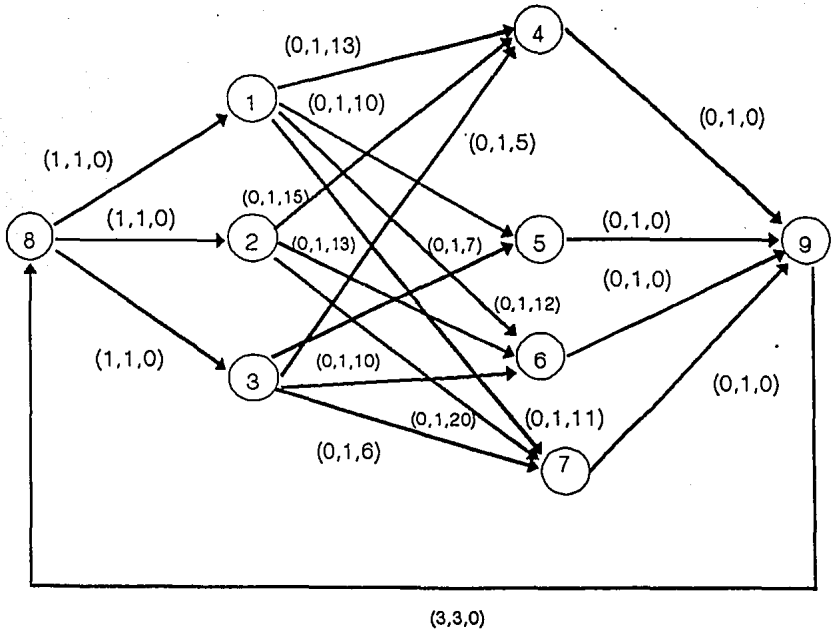
Ejemplo prototipo:

La compañía x, ha comprado 3 nuevas máquinas de diferentes tipos. Hay cuatro lugares disponibles en donde una máquina pudiera ser instalada. Algunos de estos lugares disponibles son más deseables que otros, para cada máquina en particular, debido a su proximidad a los centros de trabajo. El objetivo es asignar las nuevas máquinas, a los lugares disponibles para minimizar el costo total de manejo de materiales.

COSTO DE MANEJO DE MATERIALES

	LUGAR 1	2	3	4
1	13	10	12	11
2	15	--	13	20
3	5	7	10	6

El problema puede ser resuelto con el algoritmo Out-Of-Kilter, mediante el siguiente planteamiento.



Debido a la estructura tan particular de este tipo de modelos, existe un algoritmo sumamente eficiente para resolverlo conocido como algoritmo Húngaro de Asignación, que se presenta en la rutina A-V (Anexos).

3.4 El Problema de Flujo Máximo

En ciertos problemas estamos interesados en los valores que se generan a través de cierto flujo que pasa por una red. Este valor puede estar dado en términos de dinero, distancia, tiempo o alguna otra medida. Existen problemas en los que el valor del flujo no es tan importante como la cantidad de flujo que pasa a través de la red. Los gasoductos y las líneas de transmisión de electricidad son ejemplos de esta situación. En general tendremos, una red dirigida y conexa que tiene un solo nodo fuente y un solo nodo destino y el resto son nodos de transbordo. Dada la capacidad en los arcos, lo que nos interesa es determinar el flujo factible que pasa através de una red, que maximiza el flujo total, desde el nodo fuente al nodo destino, donde es necesario suponer que existen restricciones de capacidad en los arcos. Si no fuera así, el flujo máximo que pasaría a través de la red sería infinito.

Considérese una red con m nodos y n arcos através del cual fluye un solo tipo de bien o unidad. Con cada arco (i,j) se asocia sobre el flujo una cota inferior de $\ell_{ij} = 0$ y una cota superior de C_{ij}. En todo el desarrollo que sigue se supondrá que las C_{ij} (capacidades de los arcos) son enteros. En el problema de flujo máximo no intervienen costos. En la red, se desea encontrar la cantidad máxima de flujo del nodo 1 al nodo m.

Representándose por f la cantidad de flujo en la red del nodo 1 al nodo m, el problema de flujo máximo en una red puede establecerse como sigue:

Maximizar f

$$\text{S.A.} \quad \sum_{j=1}^{m} X_{ij} - \sum_{k=1}^{m} X_{ki} \begin{cases} f \text{ si } i=1 \\ 0 \text{ si } i \neq 1*m \\ -f \text{ si } i=m \end{cases}$$

$$0 \geq X_{ij} \leq b_{ij} \quad i,j = 1,2,\ldots,m$$

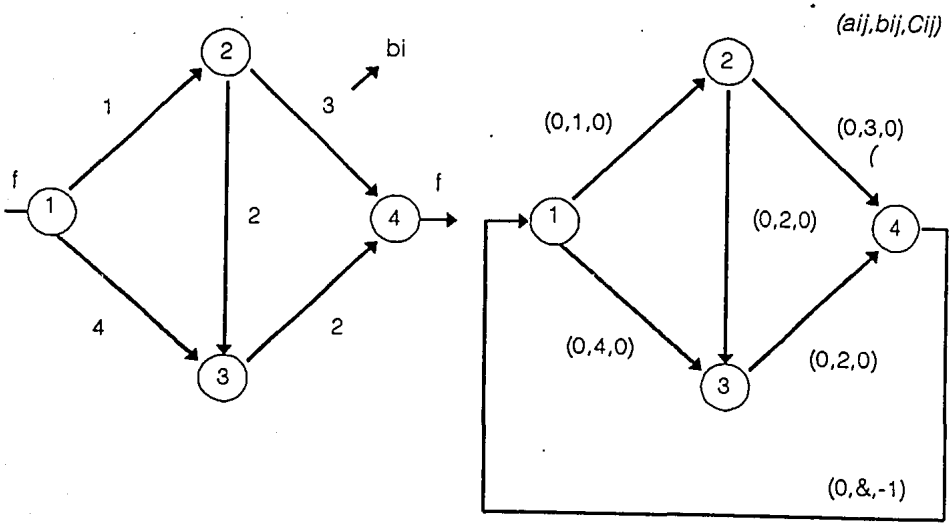
donde las sumas y las desigualdades se consideran sobre los arcos existentes en la red. Esta es la denominada formulación nodos-arcos para el problema de flujo máximo, puesto que la matriz de restricciones es una matriz de incidencia nodos-arcos. Notamos que f es una variable y sea A la matriz incidencia nodos-arcos, podemos escribir el problema de flujo máximo en forma matricial

$$\begin{cases} \text{Maximice } f \\ \text{S.A.} \\ (e_m - e_1)f + Ax = 0 \\ x \leq b \\ x \geq 0 \end{cases}$$

Esto proporciona la formulación directa del problema de flujo máximo en la forma Out-Of-Kilter, en donde el flujo es circulatorio.

Recordando que el algoritmo Out-Of-Kilter minimiza la función objetivo, asignamos un costo de cero a cada una de las variables de flujo, excepto $X_{m,1} \equiv f$, a la cual asignamos un costo de -1.

En la siguiente figura se muestra un ejemplo de un problema de flujo máximo y su problema equivalente en la forma Out-Of-Kilter.



Cabe mencionar que existen otros algoritmos más eficientes para resolver el problema de flujo máximo en una red, estos son: algoritmo de las Etiquetas y el de Ford y Fulkerson. Se muestran sus diagramas de flujo en las rutinas A-VI y A-VII (Anexos respectivamente).

3.5 El Problema de Ruta más Corta

El problema de ruta más corta, consiste en encontrar la trayectoria más corta del nodo 1 al nodo m. El costo de la trayectoria es la suma de los costos sobre los arcos en la trayectoria.

El problema de ruta más corta en el contexto de redes, consiste en enviar una unidad de flujo, del nodo 1 al nodo m, con costo mínimo.

La formulación matemática de este problema es como sigue:

$$\text{Minimice} \quad \sum_{i=1}^{m} \sum_{j=1}^{m} c_{ij} X_{ij}$$

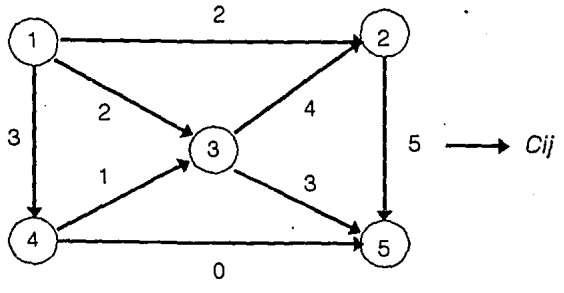
$$\text{S.A.}$$

$$\sum_{j=1}^{m} X_{ij} - \sum_{k=1}^{m} X_{ki} = \begin{cases} 1 & \text{si } i=1 \\ 0 & \text{si } i \neq 1 \text{ ó } m \\ -1 & \text{si } i=m \end{cases}$$

$$X_{ij} \geq 0 \qquad i=1,2,\ldots,m$$

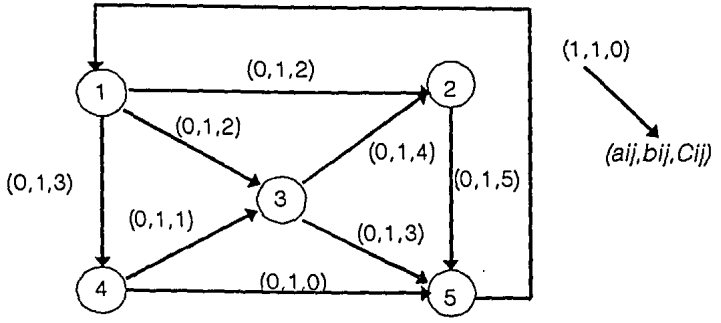
El problema de ruta más corta puede ser reformulado para ser resuelto con el algoritmo Out-Of-Kilter, usando la transformación del problema Ax=d a la forma Ax=0.

Para ilustrar como se resuelve este problema con el algoritmo Out-Of-Kilter, consideremos la red:



En donde los números sobre los arcos son las distancias, y se desea encontrar la ruta más corta entre el nodo 1 y el nodo 5. Primero añadimos un arco (5,1) con límite inferior de flujo de uno, límite superior de uno, y costo por unidad de flujo de cero. Los demás arcos de la red, tendrán un límite inferior de cero, límite superior de uno y los costos por unidad de flujos iguales a c_{ij}.

A continuación se muestra la red anterior ya transformada para ser resuelta con el algoritmo Out-Of-Kilter.



Para encontrar la ruta más larga entre los nodos 1 y m, añadimos el arco (m,1) con límite inferior de flujo de cero, límite superior de uno y costo igual a cero. Para los demás arcos (i,j), límite inferior de cero, límite superior de uno y costo de $c_{ij}=-c_{ij}$.

El algoritmo de Dijkstra rutina A-VIII (Anexos); que resuelve el problema de flujo a costo mínimo, se puede aplicar al problema de ruta más corta con mejores resultados.

En el ejemplo anterior, los arcos representan las distancias y los valores correspondientes a cada arco representan el costo de esas distancias, en este caso el objetivo consistió en encontrar la secuencia de distancias que minimizaran el costo total relacionado.
Otra posibilidad es considerar al valor asociado de cada arco como el tiempo requerido para realizar dicha actividad, el objetivo será encontrar la secuencia de actividades con las cuales se minimiza el tiempo total requerido.

Existen otras versiones de la ruta más corta las cuales consisten en :

- Encontrar la ruta más corta del orígen a todos los demás nodos de la red.
- Encontrar la ruta más corta desde todos los nodos a todos los demás nodos.

3.6 Técnicas de Previsión y Evaluación de Proyectos y Ruta Crítica (PERT/CPM).

Los problemas que involucran un considerable número de actividades tienen que planearse para que exista una programación controlada en el uso de los recursos durante el tiempo que se requieran. Un análisis de tiempos y costos, puede proporcionar parámetros valiosos para el administrador de un proyecto.

Existen diversas técnicas que permiten planear, programar y controlar una serie de actividades complejas dentro de los procesos administrativos o de producción, entre ellas pueden citarse como más importantes:

- El Método del Camino Crítico (CPM).
- Las Técnicas de Previsión y Evaluación de Proyectos (PERT).

Existen múltiples aplicaciones en las que estas técnicas han sido de gran ayuda, por ejemplo:

- La construcción de un edificio.
- La planeación e introducción de un nuevo producto al mercado.
- La instalación y corrección de algún defecto en un sistema de computadoras.
- Proyectos de investigación y diseño de ingeniería.
- La programación de la construcción y reparación de navíos.
- El proceso de secuencia en trámites administrativos, etc.

Una de las características que particularizan a las técnicas de planeación es el uso de gráficas o redes de actividades.

PERT

Primero definiremos algunos conceptos relevantes en la terminología del PERT:

Actividad: Es el trabajo necesario para cumplir un evento particular; cada actividad requiere tiempo y recursos, tiene un inicio y una terminación bien definidos y puede darse en cualquier posición del proyecto.

Evento: Es un punto en el tiempo y significa la terminación de alguna actividad y/o inicio de otra.

Varias actividades pueden predecer a un evento y dirigirse a otro. Normalmente se presentan las actividades con una flecha orientada cuya longitud no tiene significado y no es proporcional al tiempo de duración de las mismas. Cada evento está simbolizado por un nodo o círculo. Las actividades se trabajan con un tiempo de ejecución cero para facilitar la enumeración de los nodos, y se les llama actividades ficticias.

La numeración de nodos es tal que cada actividad i es siempre menor que j, ($i < j$). Los números de cada arco son progresivos. Esta numeración de nodos es importante para entender la lógica de la red de optimización y el cálculo de la misma.
Una red describe la secuencia de actividades que son necesarias para completar un proyecto. Para desarrollar la red de la técnica de evaluación y revisión de proyectos (PERT) se necesitan seguir tres pasos:

i) **Análisis de actividades**

Esta etapa involucra la delineación detallada de las operaciones y los métodos de trabajo que se deben llevar a cabo para la realización del proyecto; esencialmente el proyecto en su totalidad está subdividido en actividades. El análisis de actividades es el proceso de reducir el proyecto a sus componentes operativos para formar una lista completa de las actividades esenciales.

ii) **Diagrama de flechas**

Este desarrolla una presentación gráfica que representa las interdependencias y relaciones de precedencia entre las actividades de un proyecto. Requiere un conocimiento amplio de saber cuáles actividades deben cumplirse antes de otras actividades, cuáles pueden ejecutarse paralelamente, cuales empiezan inmediatamente después de *x* actividades, etc.

iii) **Determinación de los tiempos de actividades**

Después de haber obtenido la red es necesario estimar los tiempos de cada actividad: los modelos determinísticos requieren un solo tiempo estimado mientras que los probabilísticos necesitan 3 tipos de tiempo para cada actividad; además se supone que los tiempos de las actividades tienen una distribución tipo Beta (Fig. 3.1).

La distribución de los tiempos en PERT se supone que es una distribución continua de tipo Beta y se miden no en el eje vertical (probabilidad), sino por el área bajo la curva, y se supone que cada parte hacia la media es igual a tres desviaciones estándar, por lo que toda el área bajo la curva será 6 desviaciones estándar.

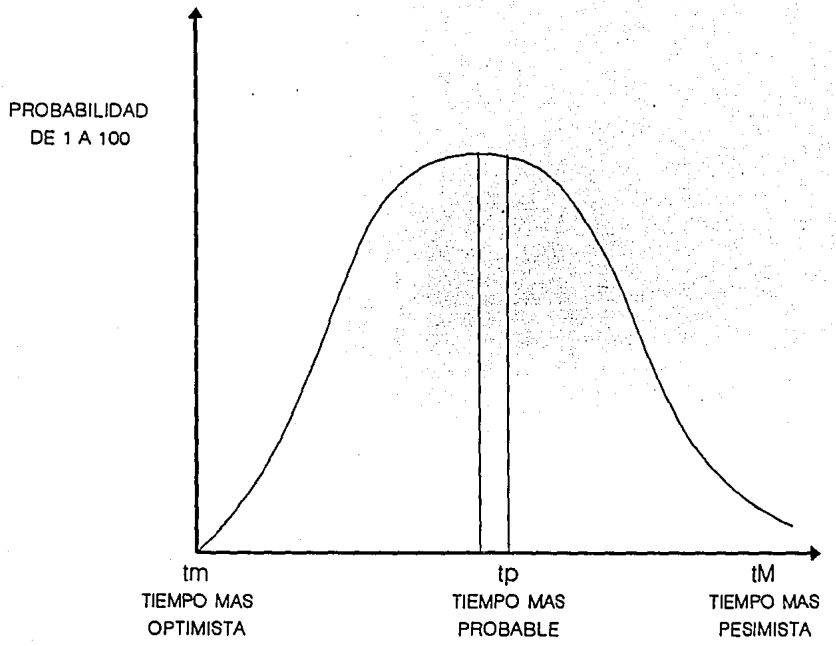


Figura 3.1

Tiempos:

1. **tm:** Tiempo mínimo u optimista. Es el tiempo mínimo que una actividad puede requerir.

2. **tM:** Tiempo máximo o pesimista. Es el tiempo máximo requerido por una actividad.

3. **tp:** Tiempo más probable. Es el tiempo que se requerirá si la actividad ocurre varias veces bajo las mismas circunstancias, es decir, es una estimación de la moda de la distribución del tiempo.

En una red pueden aparecer los tres tiempos o uno sólo, que es el valor esperado.

Normalmente se utiliza un solo tiempo cuando hay datos históricos adecuados de la duración de las actividades o cuando no hay datos históricos confiables, ocuando se está desarrollando por primera vez un proyecto y requiere estimar los tiempos mínimo, máximo y probable de cada actividad.

La distribución Beta tiene la ventaja de proporcionar los tres tiempos mencionados (tm,tM y tp), además la facilidad para medir la incertidumbre en las estimaciones.

Después de estimar los tiempos más pesimista, más optimista y más probable, se deben combinar para determinar uno solo. Esto se logra mediante el promedio ponderado.

$$t = \frac{a + 4m + b}{6}$$

donde:

t = Tiempo esperado
a = Tiempo más pesimista
b = Tiempo más optimista
m = Tiempo más probable.

Como tenemos tres tiempos para cada actividad, se puede calcular una desviación estándar para ella:

$$\frac{b - a}{6}$$

RUTA CRITICA

Determinar la ruta más larga de una red es conocer la ruta crítica de ella. Para determinar la ruta crítica es necesario calcular el tiempo de terminación más rapido, el tiempo de terminación más tardío y las holguras disponibles.

Tiempo de terminación más rápido (TTMR). Es el tiempo necesario para iniciar las actividades de un evento dado. Si dos o más actividades parten de un evento el TTMR de ése es el mayor de todos, es decir, es la actividad que requiere más tiempo para su terminación, ya que no se pueden empezar las nuevas actividades mientras no se haya terminado por completo la etapa anterior. Para encontrar el tiempo de terminación más rápido se debe iniciar el cálculo con la primera actividad, recorriendo toda la gráfica hasta el final de la misma, acumulando los TTMR de las fases precedentes.

TTMT = tiempo de inicio más temprano TIT + tiempo esperado (T)
TTMR = TIT + T.

El tiempo de terminación más tardío es el mayor tiempo que se requiere para que una actividad siguiente pueda realizarse; el cálculo se hace del último evento hacia el primero, acumulando los **TTMR** en el recorrido de toda la gráfica. Se toma como base el total acumulado hasta el evento y de ahí se van restandoo las actividades precedentes.

TTMT = Ultimo tiempo de terminación - tiempo esperado
TTMT = UTT - T

Cuando dos o más actividades parten de un evento se elige como TTMR el mínimo entre ellos, ya que nos indicará cuál es la actividad que requiere menos tiempo.

Después de realizar los dos pasos anteriores para calcular el tiempo de terminación más rápido y el tiempo de terminación más tardío, se pueden calcular las holguras.

La ruta crítica del programa será aquella en que las holguras de todas y cada una de las actividades sea cero.

A la cantidad de tiempo que una actividad puede retrasarse sin afectar el tiempo de ocurrencia más rápido de otra actividad se le llama **holgura libre**, o también se le puede conocer como exceso de tiempo.

La ruta crítica nos permite manejar las holguras de manera que la combinación de las actividades no críticas nos lleve a ahorros de tiempos y costos, o también a recortar el tiempo de realización del proyecto al menor costo posible.

La notación que utilizaremos simplificará lo más posible la red, para que en una sola se tengan todas las posibles situaciones que se puedan presentar por la o las rutas críticas, los tiempos de las actividades, los costos de las mismas, la comprensión de la red, etc. (Fig. 3.2). Por ejemplo, si tenemos el proyecto de producir y lanzar al mercado un artículo, de manera general, en el cual se pueden citar las siguientes actividades:

I 1) Localización de planta.
II 2) Diseño de instalaciones.
III 3) Concurso y cotización.
IV 4) Construcción o remodelación.
V 5) Adaptaciones finales para instalación de maquinaria.
VI 6) Adquisición de maquinaria.
VII 7) Instalación.
VIII 8) Pruebas preliminares.
IX 9) Contratación de personal.
X 10) Compra de materia prima.
XI 11) Iniciación de operaciones.

Las actividades 1,2,3,4,5 y 11 se suceden en este orden, lo mismo que las actividades 7,8 y 10.

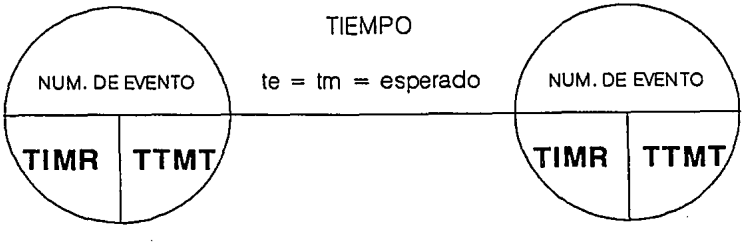


Figura 3.2

Poniendo los eventos numerados de mayor a menor y haciendo pequeños arreglos para evitar el cruce de actividades, la red nos quedaría como se indica en la figura 3.3.

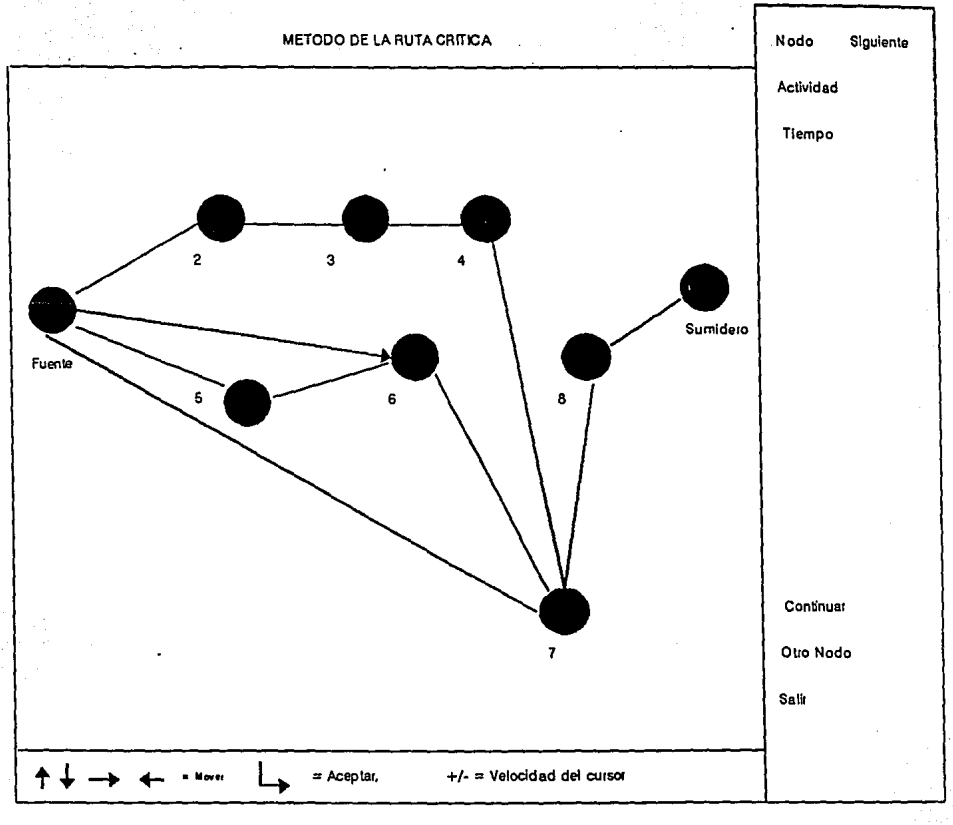

METODO DE LA RUTA CRITICA
Nodo Siguiente
Actividad
Tiempo
2
3
4
Sumidero
Fuente
5
6
8
7
Continuar
Otro Nodo
Salir
= Aceptar,
+/- = Velocidad del cursor

Figura 3.3

Calculando los TIMR y TTMR utilizando el programa llamado CPM (el manejo del programa se presenta en la sección de anexos) tenemos la siguiente figura 3.4 donde las únicas actividades críticas, sin holguras son: 1 (fuente)-2, 2-3, 3-4, 4-7, 7-8, y 8-9 (sumidero), lo que nos indica que es la ruta crítica.

METODO DE LA RUTA CRITICA

Nodos		Tiempo	Actividad	T.msPrx. InicioAct	T.msPrx. Fin Act	T.mslej. InicioAct	T.mslej. Fin Act	Holgura Total	Holgura Indep.
1	2	4	I	0	4	0	4	0	0
1	5	3	VII	0	3	4	7	4	0
1	6	2	IX	0	2	6	8	6	2
1	7	3	VI	0	3	7	10	7	7
2	3	2	II	4	6	4	6	0	0
3	4	2	III	6	8	6	8	0	0
4	7	2	IV	8	10	8	10	0	0
5	6	1	VIII	3	4	7	8	4	0
6	7	2	X	4	6	8	10	4	4
7	8	5	V	10	15	10	15	0	0
8	9	3	XI	15	18	15	18	0	0

Figura 3.4

Las condiciones que deben cumplirse para que la ruta sea crítica deben ser:

a) Los tiempos inicial y final en cada evento tienen que estar iguales.

b) La diferencia entre el tiempo inicial y el final anterior al primero debe ser al tiempo de las actividades entre el TIMR y TTMT anterior.

Tiempo actividad j = $TIMR_j$ - TTMT - 1

CONCLUSIONES

En el desarrollo de este trabajo se ha mostrado que redes de ciertos tipos surgen en una amplia variedad de contextos, pudiéndose considerar como problemas de Programación Lineal.

El análisis de redes proporciona técnicas útiles (en especial las técnicas de optimización) para el diseño y operación de sistemas de redes. Por su naturaleza los problemas de redes son con frecuencia difíciles de resolver, pero se han hecho grandes avances en el desarrollo de poderosas técnicas de modelado y de metodologías que incluso, amplían el panorama hacia nuevas aplicaciones. De hecho los algoritmos más recientes han permitido resolver algunos problemas de redes grandes y complejos.

Los problemas lineales que resultan en el estudio de redes de flujo, han sido analizados en un marco metodológico que permite unificar los diversos enfoques y métodos de solución existentes en campos tan diversos como la investigación de operaciones, ingeniería eléctrica y optimización combinatoria.

El aspecto unificador del análisis de los problemas lineales a el denominado problema de redes con flujos restringidos y costo mínimo es que pueden reducirse a sus casos particulares, conocidos como problemas básicos de redes de flujo, permitiendo la generación de métodos de solución combinando métodos aparentemente desconectados. Este aspecto no interfiere con el hecho de que se sigan usuando y generando métodos de solución particulares.

Los problemas de redes de flujo tratados en este trabajo cuentan con un análisis de existencia y factibilidad de soluciones así como procedimientos para determinar tales soluciones (si existen), además se proporcionan diversos algoritmos para

resolverlos.

Cabe mencionar que el algoritmo de las desviaciones (Out-Of-Kilter) resulta fácil de aplicar lo que constituye una de las técnicas más aplicadas en diversos campos de la ingeniería.

Conviene señalar que los modelos de redes de flujo presentados han sido motivados con ejemplos para mostrar su aplicabilidad, como pudo apreciarse, éstos modelos persiguen un fin específico: servir de auxiliares en el difícil proceso de la toma de decisiones. En esta acción, se auxilia de representaciones matemáticas con diversos grados de complejidad, con las que busca determinar cursos de acción viables que permitan al decisor satisfacer al máximo sus necesidades. Sin embargo, en su aplicación, el uso de las computadoras se vuelve imprescindible, pues es difícil concebir procesos de resolución manuales para problemas reales. No obstante, es importante tener en cuenta que como tales, estas herramientas auxiliares, no representan la panacea. El factor humano, juega un papael preponderante; es el hombre quien, a fin de cuentas, decidirá y pondrá en práctica los planes de trabajo. La intuición de este y su compenetración con los problemas reales, lo ayudarán a obtener con mayor éxito las metas que persigue.

Las matemáticas al igual que las computadoras son herramientas que en su afán por simplificar y universalizar un lenguaje común, ha creado el hombre. Por esta razón resulta importante que, como tales, sean utilizadas para el fin que por el que fueron concebidas, que nos familiaricemos más con su empleo y que no sean consideradas como el manjar de un pequeño círculo de intelectuales. en ello el proceso de aprendizaje juega un papel preponderante como factor motivacional, que ayudará al interesado a profundizar más sobre el conocimiento en el que incursiona.

BIBLIOGRAFIA

- Ackoff, Russel L., y M.W Sasieni, **Fundamentos de Investigación de Operaciones**, Limusa, México, 1979.

- Bazaraa Mokhtar. S., Jarvis J.J., **Linear Programming and Network Flows**, John Wiley & Sons, New York, N.Y., 1977.

- Busacker P.G. y Saaty T.L. **Finitegraphs and networks**, McGraw-Hill, 1965.

- Berge, Claude, **Theory of Graphs and Applications**, John Wiley and Sons,Inc., New York, N.Y., 1962.

- Cadena Landeta Alberto, **Algoritmos para resolver problemas de redes**, Tesis de Posgrado, División de Estudios de Posgrado, Facultad de Ingeniería de la U.N.A.M., 1982

- Dantzing, G.B., **Linear Programming and Extensions**, Princeton University Press, Princeton, N.J., 1963.

- Davis/Mckeown, **Modelos Cuantitativos para Administración**, Iberoamérica, Belmont, California, 1986.

- E. Hobson, D. L. Fletcher, W,O. Stadlin **Network Flow Linear Programming Technique and their Application to Fuel Scheduling and Contigency Analysis**, IEEE Transaction on Power Apparatus and Systems, Vol. 7, 1984.

- Ford, L.R., y D.R. Fulkerson, **Flows in Networks**, Princeton, N.J., 1962.

- Hu T. C., **Integer Programming and Network Flows**, Addison Wesley, 1969.

- Hiller F. y G. Lieberman, **Introducción a la Investigación de Operaciones**, McGraw-Hill, 1989.

- Jauffred, Francisco J., y A.M. Bonett, **Métodos de Optimización y Programación Lineal-Gráficas**, Representaciones y Servicios de Ingeniería, México, 1976.

- Lara Rosano Felipe, **Análisis Topológico de la Confiabilidad de Redes de Flujo Sujetos a Restricciones de Calidad.** Tesis de Posgrado, División de Estudios de Posgrado, Facultad de Ingeniería U.N.A.M. , 1973.

- Parra Ugalde Mario, **Redes de Flujo con Costos Convexos y Cóncavos**, Tesis de Posgrado, División de Estudios de Posgrado, de la Facultad de Ingeniería U.N.A.M. , 1982.

- Prawda Juan, **Métodos y Modelos de Investigación de Operaciones, Tomo I, Modelos Determinísticos**, Limusa, México, 1987.

- Veinott, JR. and Dantzing G.B:, **Integer Extreme Points**, SIAM 10,3, 1968.

- Von Lanzenaver, Christhoph Haehling, **Cases in Operations Research**, Holden Day, San Francisco, 1975.

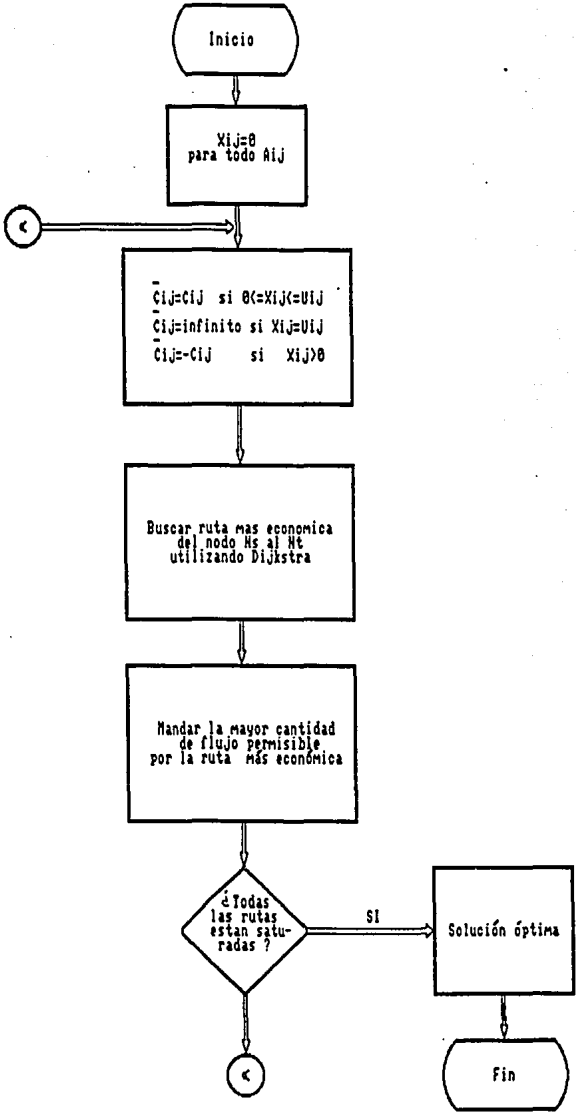
- Wagner M. H. **Principles of Operation Research**, Prentice Hall, 1975.

- Williams, S. Jewll, **Optimal Flow Through Networks with Gains**, Operation Research, Vol. 10 1962.

ANEXO

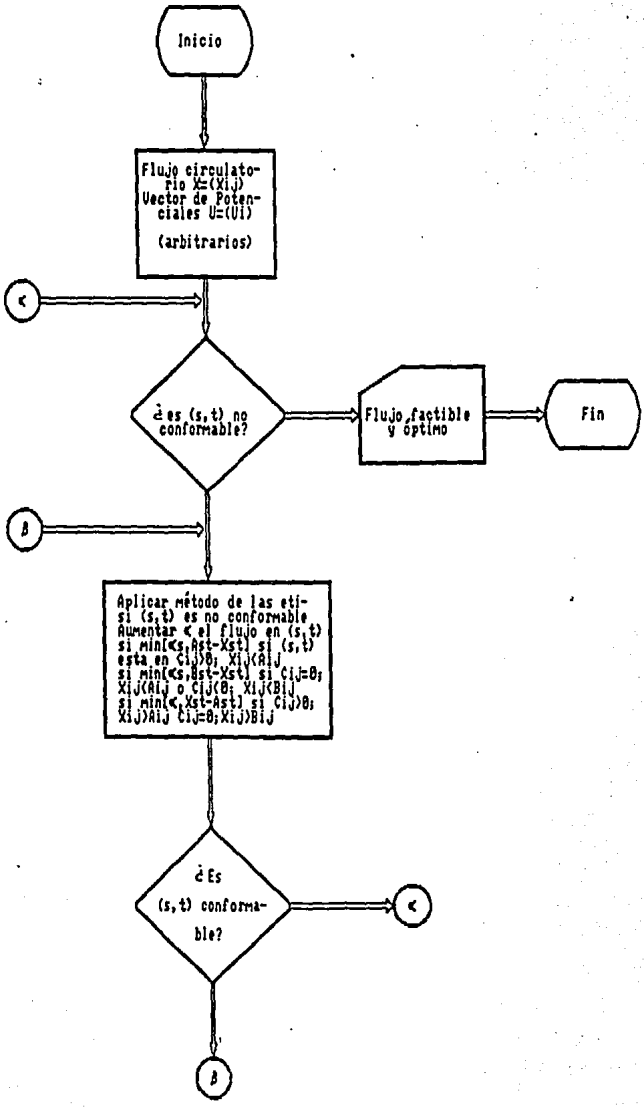
A

DIAGRAMA DE FLUJO DEL
ALGORITMO DE BUSACKER Y GOWEN



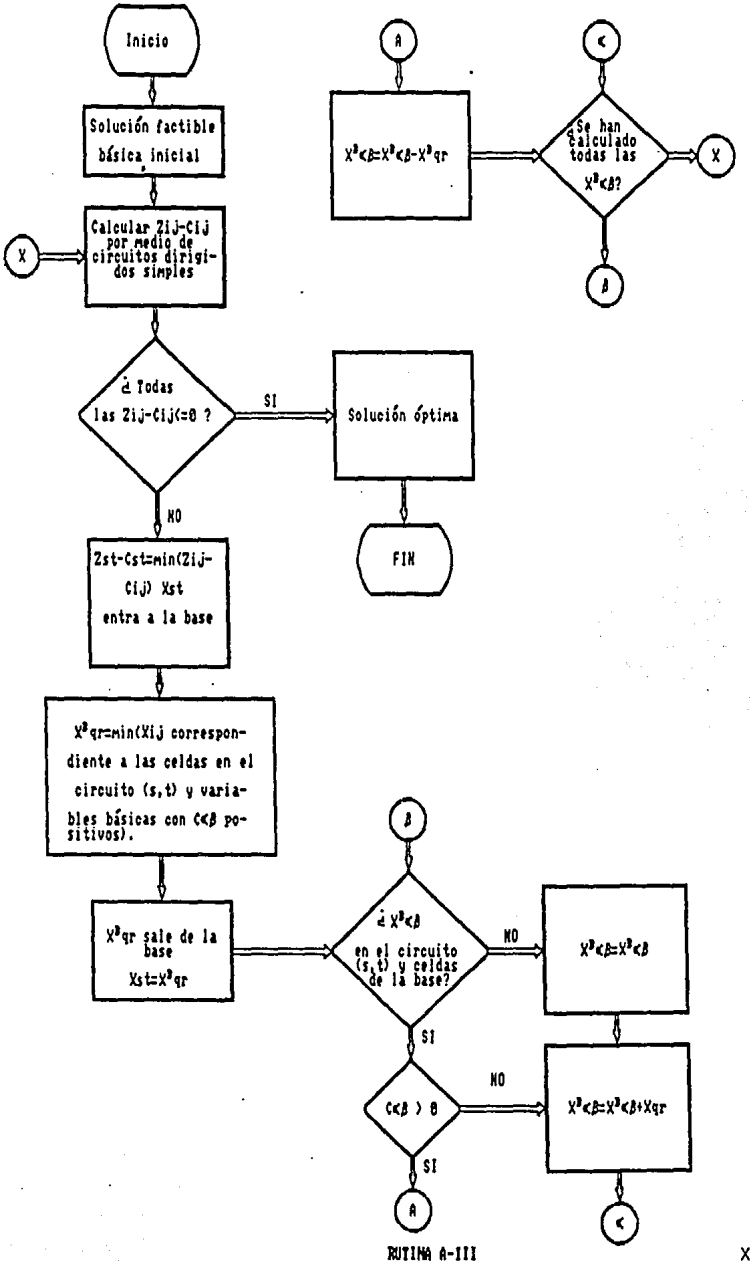
RUTINA A-I

DIAGRAMA DE FLUJO DEL ALGORITMO
OUT-OF-KILTER



RUTINA A-II

DIAGRAMA DE FLUJO DEL ALGORITMO DE PIEDRA DE PASO



RUTINA A-III

DIAGRAMA DE FLUJO DEL METODO DE VOGEL

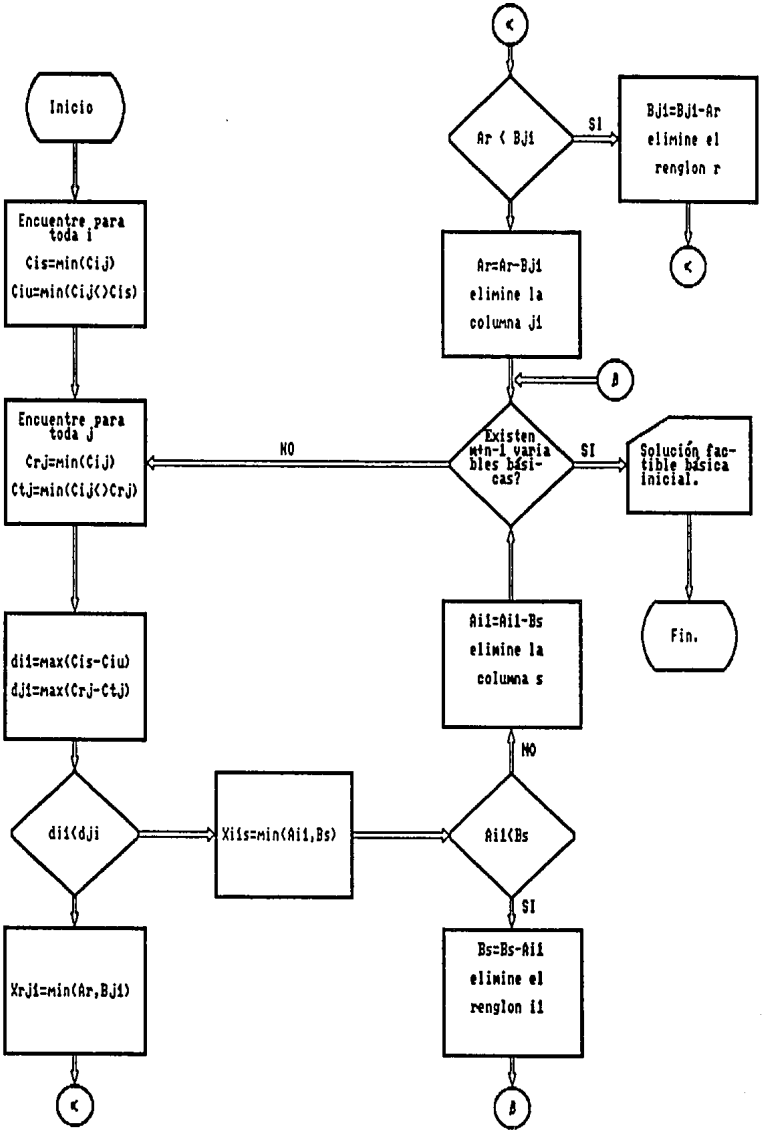
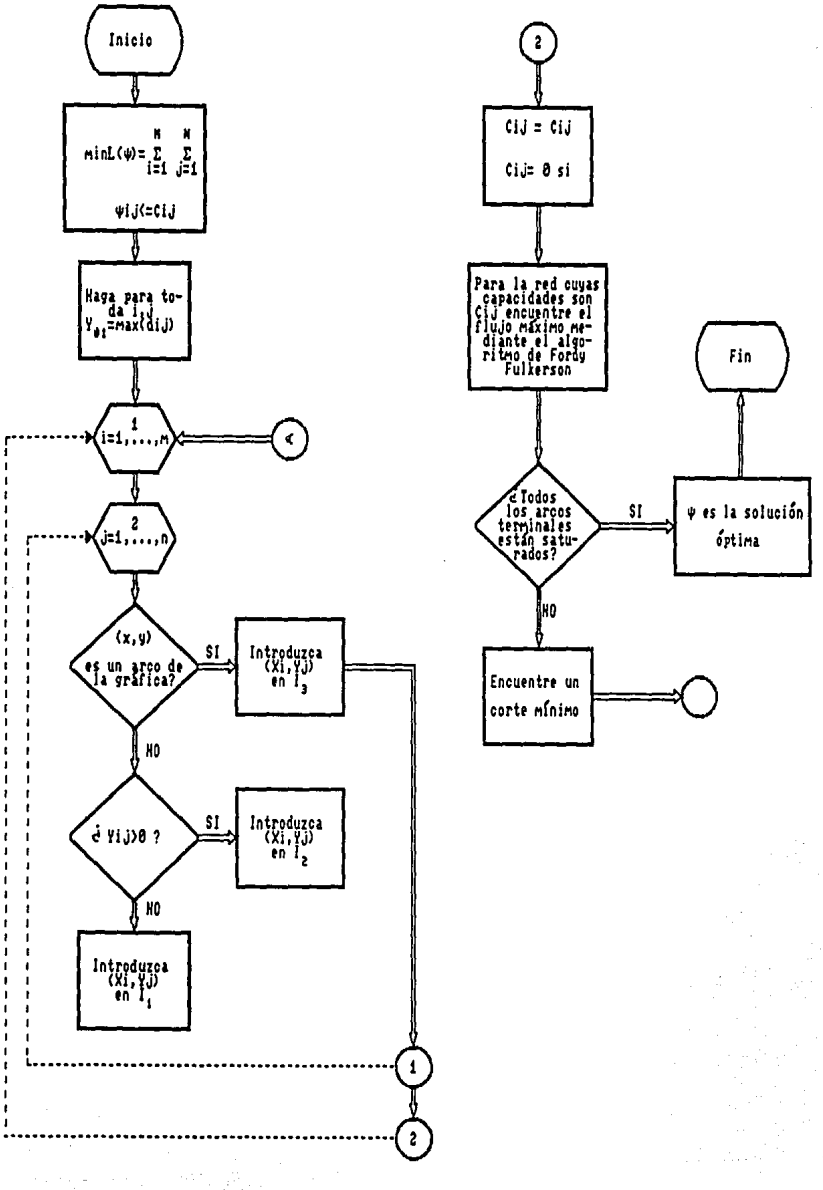
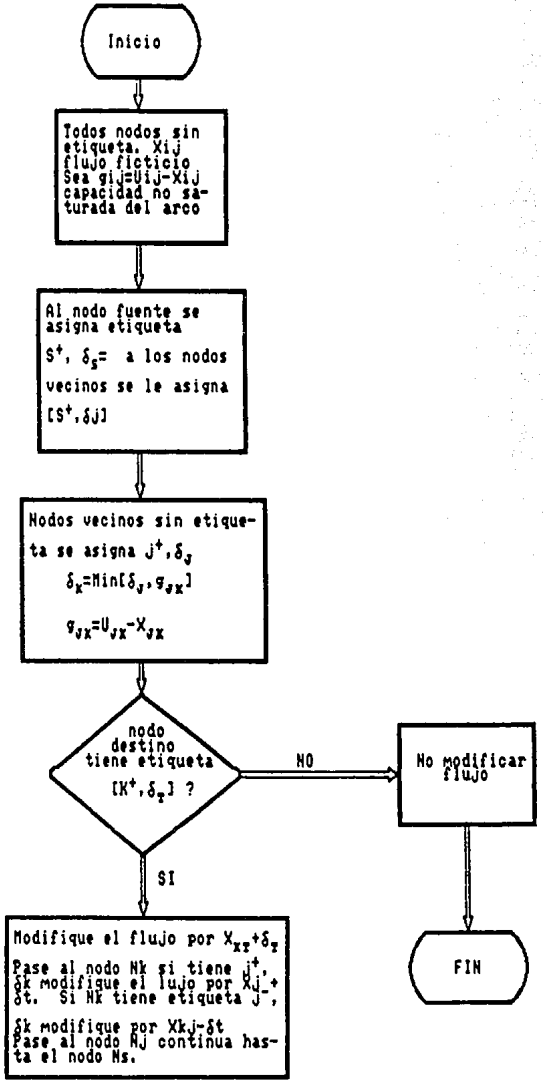


DIAGRAMA DE FLUJO DEL ALGORITMO HUNGARO



RUTINA A-V

DIAGRAMA DE FLUJO DEL ALGORITMO
DE ETIQUETAS



RUTINA A-VI

DIAGRAMA DE FLUJO DEL ALGORITMO DE FORD Y FULKERSON

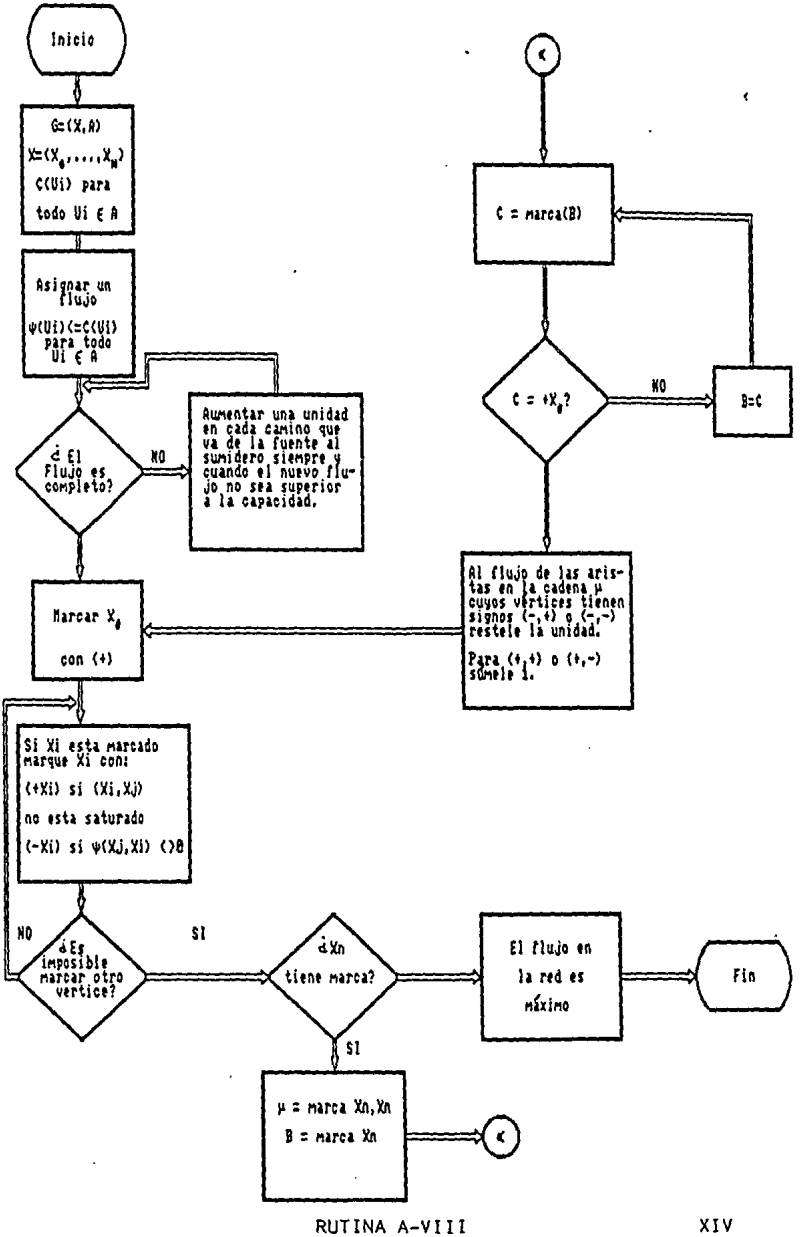
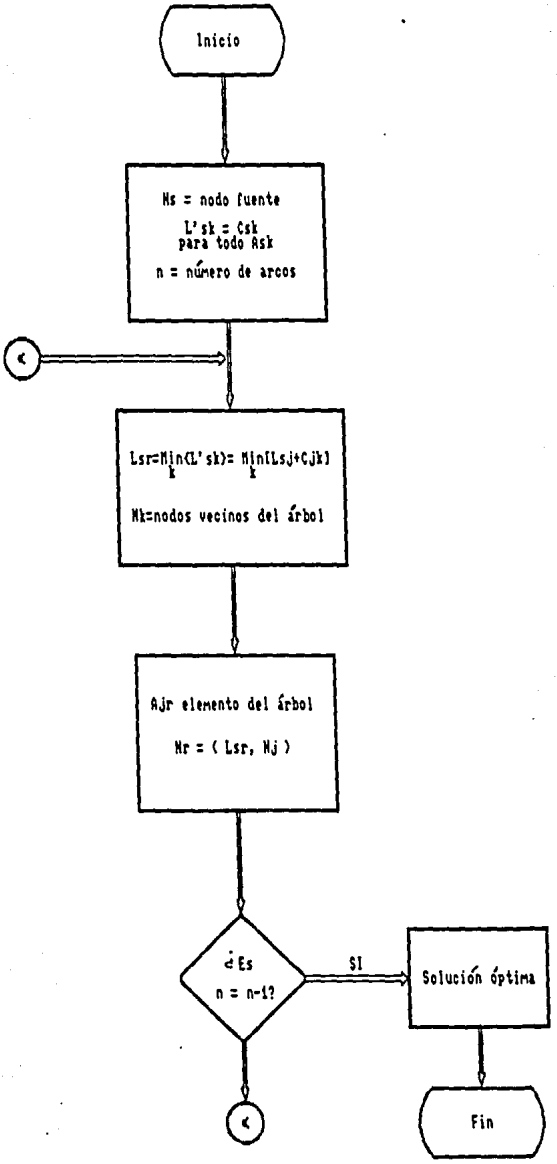


DIAGRAMA DE FLUJO DEL
ALGORITMO DE DIJKSTRA



RUTINA A-VIII

ANEXO

B

Anexos

Sección B

Manual de Usuario para el manejo del programa de la Ruta crítica

Funcionamiento del Programa

B.1 Requerimientos de Equipo

El programa de la Ruta crítica (CPM), recomienda lo siguiente como requerimiento mínimo para lograr un correcto funcionamiento:

* Computadora PC compatible con IBM, Sistema Operativo MS-DOS versión 3.1 en adelante; ya sea XT 8086 ó una AT 80286 o posterior.

* 640 Kb en memoria RAM.

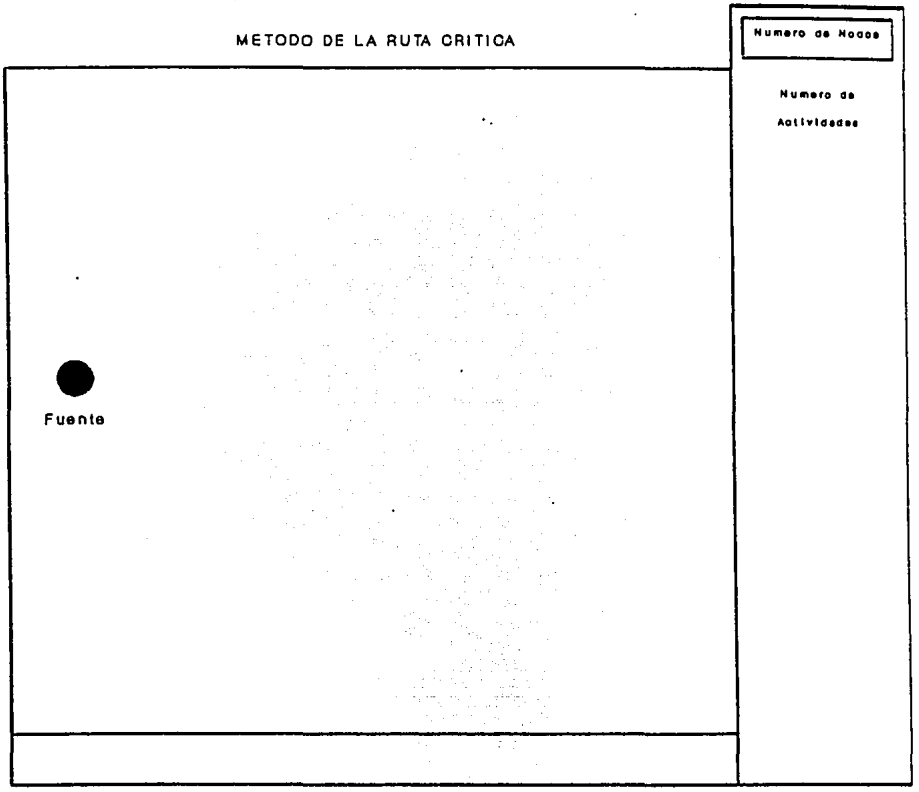
* Monitor CGA, VGA, o SUPER VGA.

* Unidad de disco flexible $5^1/_4$

* El archivo Config.sys debe tener como mínimo 40 Files y 35 Buffers.

B.2 Entrada al Programa

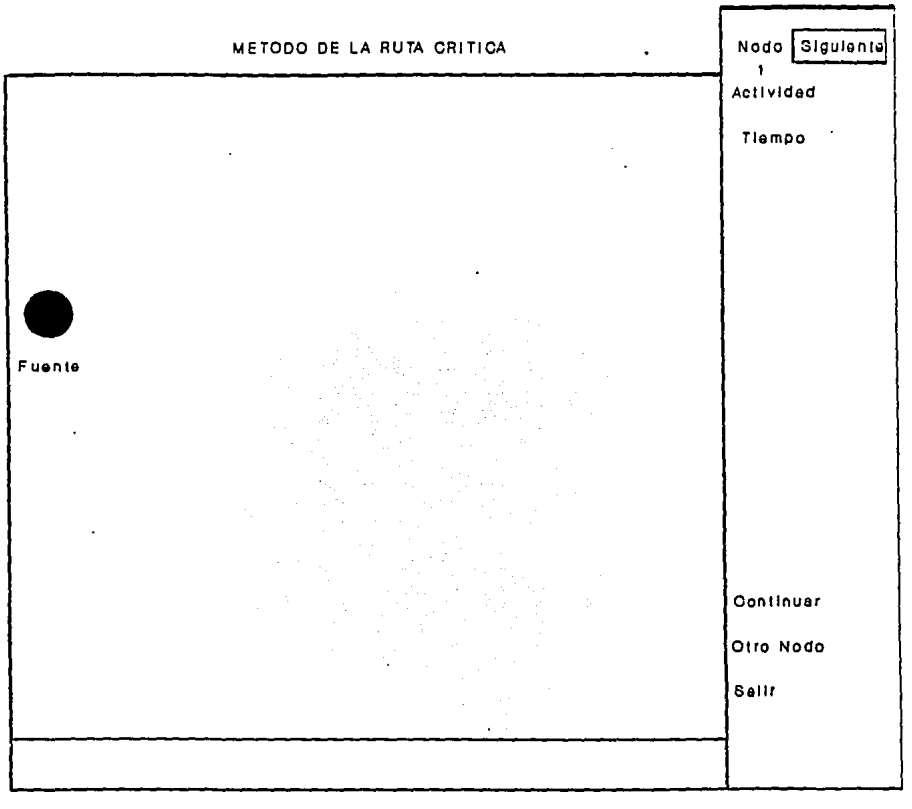
Para tener acceso al programa de la Ruta Crítica, es necesario teclear CPM en la unidad en que esté cargado el programa, al hacer esto aparecerá en la pantalla la siguiente presentación:



B.3 Modo de operación en pantalla

Como se observa en la pantalla se presenta encerrado en un rectángulo el dato que se debe proporcionar, el primer dato que pide es el número de nodos del problema; en el cual el mayor número de nodos que acepta el programa son 20, aunque esto se puede modificar fácilmente en el programa, entrando al editor de PASCAL versión 5.5 y modificando la variable Max_Nodos e indicando el número de nodos deseado. El programa valida los datos, en las opciones en las que únicamente se pueden proporcionar como datos números, (como son número de nodos, número de actividades, tiempo), no acepta ningún otro valor, si lo que damos son letras y tecleamos ENTER el programa nos vuelve a solicitar el valor, hasta que se proporcione un número, una vez indicado no se puede modificar el valor; al dar ENTER el rectángulo se posiciona del siguiente dato que es requerido en este caso es el número de actividades, el cual no debe ser mayor de 30, aunque como en el caso anterior también puede ser modificado este valor con la variable Max_Acti. Además se encuentra dibujado el nodo número 1, el cual es la fuente de la red.

Una vez proporcionado el número de actividades aparece la siguiente pantalla:



El rectángulo ahora de encuentra posicionado en las opciones de la parte baja de la pantalla estas son: Continuar, Otro Nodo y Salir, para elegir alguna opción es suficiente con moverse através de éstas mediante flechas y presionar ENTER en la opción deseada.

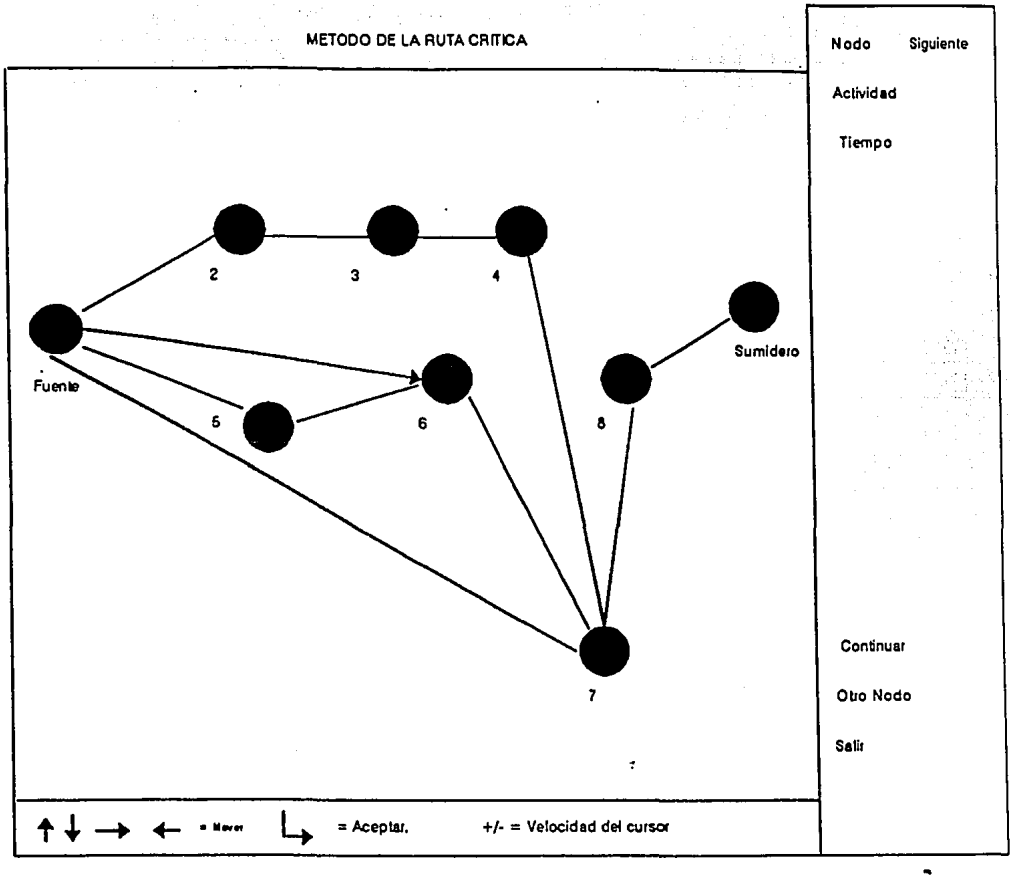
Si la opción es continuar el rectángulo ahora se encuentra en Siguiente, se proporciona el número de nodo que le sigue, y se continúa con el nombre de la actividad y el tiempo para realizar

esa actividad, para posicionarse en estas opciones de actividad y tiempo también se realiza con las flechas.

Dados estos datos aparece en el centro de la pàntalla el nodo que continúa , el cual el usuario debe colocar en la posición deseada para moverlo, únicamente se pueden utilizar las teclas que se observan en la pantalla ↓ ↑ →← = Mover, |→ = aceptar, +/- = velocidad el cursor; se debe tener cuidado de no cruzar los nodos sobre las actividades, o los nodos sobre los nodos pues el programa no redibuja la gráfica, una vez colocado el nodo se tecla ENTER para aceptar el nodo en la posición dada; el programa simultáneamente une los nodos con rectas, si la actividad es ficticia ésta es dibujada en forma discontínua, en caso contrario la línea es contínua.

Ahora si la opción elegida es Otro Nodo significa, por ejemplo que del nodo 1 ya no continúa ningún otro nodo, por lo tanto en la parte superior de la pantalla en la opción Nodo aparecerá el número 2 y se pide proporcionar el número de nodo que le sigue al nodo 2, de esta manera se prosigue hasta concluir con el número de nodos y actividades que se solicitaron.

A el último nodo el programa le da el nombre de sumidero, si durante la entrada de datos ya están dibujados los nodos requeridos únicamente el programa dibuja los arcos o actividades que los unen.
En la siguiente pantalla se presentan todos los nodos y arcos que se proporcionaron.



Proporcionados todos los datos, únicamente se teclea la opción ENTER en la cual aparece la siguiente pantalla donde se observan todos los datos proporcionados y los cálculos que realiza el programa, encerrando en rectángulos las actividades y nodos que forman parte de la ruta crítica.

METODO DE LA RUTA CRITICA

Nodos		Tiempo	Actividad	T.msPrx. InicioAct	T.msPrx. Fin Act	T.mslej. InicioAct	T.mslej. Fin Act	Holgura Total	Holgura Indep.
1	2	4	I	0	4	0	4	0	0
1	5	3	VII	0	3	4	7	4	0
1	6	2	IX	0	2	6	8	6	2
1	7	3	VI	0	3	7	10	7	7
2	3	2	II	4	6	4	6	0	0
3	4	2	III	6	8	6	8	0	0
4	7	2	IV	8	10	8	10	0	0
5	6	1	VIII	3	4	7	8	4	0
6	7	2	X	4	6	8	10	4	4
7	8	5	V	10	15	10	15	0	0
8	9	3	XI	15	18	15	18	0	0

Si no se desea concluir el problema se tiene la opción Salir con lo cual permite al usuario salir al sistema operativo.

B.4 Listado del Programa

Debido a que el programa puede ser modificado, se presenta el listado del mismo.

```
{  Programa  :   CPM.PAS                                                  }
{  Función   :   Método de la Ruta Crítica (Critical Path Method)         }
{  Fecha     :   29 Ene 1993        -     7 Abr 1993                      }
{                                                                         }

Program CPM;

Uses
  Crt, Graph;

Const
  Max_Nodos = 20;
  Max_Acti  = 30;
  CTab    = ^T;
  CRight  = ^D;
  CUp     = ^E;
  CLeft   = ^S;
  CDown   = ^X;
  CExit   = ^[;
  CEnt    = ^I;


Type
  Atributo  =  (NN,II,NR);
  Num_Nod   = 0..Max_Nodos;
  Num_Activ = 0..Max_Acti;
  VectN     = Array[1..Max_Nodos] Of Word;
  VectA     = Array[1..Max_Acti] Of Word;
  Cadena    = Array[1..Max_Acti] Of String[20];
  Coord     = Array[1..Max_Nodos] Of PointType;
  Conj      = Set Of Num_Nod;

Var
  NI, NF, ES, LS, LF, TF, FF, T, TE, RC, EF            : VectA;
  Nod                                                  : Integer;  {Num_Nod;}
  NumAct, ContAct                                      : Integer;  {Num_Activ;
```

```
  ACT, DIB, eerc                                    : Cadena;
  Posi                                              : Coord;
  op2, tit, numst, nis,nfs,ess,lss,lfs,tfs,ffs,ts,efs : String;
  Sigue, Ch, mat, Res, Res4, KK, op1, op3,
  op4, op5, op6, Con                                : Char;
  cumple, Done, Nueva, Acaba, Sale, Graf            : Boolean;
  Nodos                                             : Pointer;
  Gd, Gm, MaxX, MaxY, Rango, coef2,
  valor3, valor4                                    : Integer;
  Size, i, j, xx1, yy1, tam2, k, ijk, cuadsel,
  Conta, Max, Min, Inter, Vel_Cur                   : Word;
  Conj_Nodos                                        : Conj;

Procedure Ver(Atrib : Atributo);        {Este procedimiento detecta
Begin                                    si la computadora cuenta
  Case Atrib Of                          con tarjeta graficadora.
    NM : NormVideo;
    II : Begin
           TextColor(Black); TextBackGround(LightGray);
         End;
    WR : Begin
           TextColor(Black); TextBackGround(LightGray);
         End;
  End;
End;
Procedure Inicia_Grafica;               {Procedimiento con el cual
Var                                      se inicia el dibujo de la
  InGraphicsMode          : Boolean;     gráfica
  PathToDriver            : String;
  GraphMode, GraphDriver  : Integer;
Begin
  GraphDriver := Detect;
  InitGraph(GraphDriver, GraphMode, PathToDriver);
  If GraphResult <> grOK  Then  Graf := False
  Else  Graf := True;
end;
```

```
Procedure Beep;                          {Sonido con el cual indica
Begin                                    hasta que rango es posible
  Sound(500); Delay(25); NoSound;        dibujar los nodos, al
End;                                     sobrepasarlo se instala

Function ReadChar : Char;                {Función con la cual
Var                                      únicamente se pueden
  Ch  : Char;                            utilizar las teclas de
Begin                                    Return y flechas para
  Ch := ReadKey;                         dibujar la gráfica
  If Ch = #0 Then
    Case ReadKey Of
      #45: Ch := CExit;     { Alt-X }
      #72: Ch := CUp;       { Up    }
      #75: Ch := CLeft;     { Left  }
      #77: Ch := CRight;    { Right }
      #80: Ch := CDown;     { Down  }
    End
  Else
    If Ch = #9 Then  Ch := CTab    { Tab }
    Else
      If Ch = #13 Then  Ch := CEnt
      Else
        Ch := UpCase(Ch);
  ReadChar := Ch;
End;

Procedure Correcta( i,j : Word; inta, intb : Integer; Var opp1 : Integer);
Var
  cod, coef  : Integer;                  {Procedimiento con el cual
  opp2       : String;                   se detecta si los datos de
  opp : Char;                            la gráfica son correctos
                                         es decir datos de número
                                         de nodos y ardos
Begin
```

```
  cod :=1;
  Repeat
    coef := 0;
    opp2 := '';
    Repeat
      Repeat
        opp := ReadChar;
      Until opp in ['0'..'9',CEnt];
      If opp <> CEnt Then
      Begin
        opp2 := opp2+opp;
        OutTextXY(i+coef*6,j,opp);
        coef := coef+1;
      End;
    Until opp in [CEnt];
    Val(opp2,opp1,cod);
    If (opp1 < inta) Or (opp1 > intb) Then  cod := 1;
    If cod <> 0 Then
    Begin
      SetColor(0);
      OutTextXY(i,j,opp2+'  ');
      SetColor(GetMaxColor);
    End;
  Until cod = 0;
End;

Procedure Marco (Ren, Col, Ren2, Col2 : Integer);{Procedimiento con el
Begin                                              cual  se realizan  los
  Line(Ren,Col,Ren,Col2); Line(Ren,Col2,Ren2,Col2);    marcos  de  los
  Line(Ren2,Col2,Ren2,Col); Line(Ren2,Col,Ren,Col);          cuadros
End;

Procedure Dibuja_Nodos;                {Procedimiento con el  cual
Begin                                   se dibujan los nodos de la
  ClearDevice;                          gráfica
  For i:=1 To 7 Do  Circle(8,8,i);
```

```
  Size := ImageSize(1,1,16,16);
  GetMem(Nodos,Size);
  GetImage(1,1,16,16,Nodos^);
  ClearDevice;
End;

Procedure Pon_Nodos;                    {Procedimiento con lo cual
Begin                                    se dejan fijos los nodos
 Inicia_Grafica;                        en el lugar solicitado
 Dibuja_Nodos;
 MaxX := GetMaxX;  MaxY := GetMaxY; Rango := 100;
 Rectangle(1,10,MaxX-Rango-1,MaxY-1); Line(1,MaxY-15,MaxX-Rango-1,MaxY-15);
 SetTextStyle(SmallFont,HorizDir,4);
 OutTextXY(Trunc((MaxX-Rango)/2)-Trunc(TextWidth(' METODO DE LA RUTA CRITICA ')
          -3,' METODO DE LA RUTA CRITICA ');
 PutImage(4,Trunc(MaxY/2),Nodos^,NormalPut);   { 200 }
 Posi[1].x := 4; Posi[1].y := Trunc(MaxY/2);   { 200 }
 OutTextXY(4,Trunc(MaxY/2)+15,'Fuente');       { 200 }
 Rectangle(MaxX-Rango,1,MaxX-1,MaxY-1);
 OutTextXY(MaxX-Rango+5,7,'N#mero de Nodos');
 OutTextXY(MaxX-Rango+16,50,' N#mero de');
 OutTextXY(MaxX-Rango+16,63,'Actividades');
 Rectangle(MaxX-Rango+3,7,MaxX-4,20);
 Correcta(MaxX-Rango+38,20,2,Max_Nodos,Nod);
 SetColor(0);
 Rectangle(MaxX-Rango+3,7,MaxX-4,20);
 SetColor(GetMaxColor);
 Rectangle(MaxX-Rango+10,50,MaxX-Rango+92,76);
 Correcta(MaxX-Rango+38,76,Nod-1,Max_Acti,NumAct);
 SetColor(0);
 Rectangle(MaxX-Rango+10,50,MaxX-Rango+92,76);
 SetColor(GetMaxColor);
 SetFillStyle(EmptyFill,GetMaxColor);
 Bar(MaxX-Rango+2,7,MaxX-2,89);
 SetFillStyle(SolidFill,GetMaxColor);
 Conj_Nodos := []; j := 1; op2 := ''; ContAct := 0; cuadsel := TextHeight('A')+
```

```
Repeat
 OutTextXY(MaxX-Rango+2,Trunc(Trunc(MaxY/2)*0.035),'Modo');     { 7 }
 OutTextXY(MaxX-Rango+40,Trunc(Trunc(MaxY/2)*0.035),'Siguiente');     { 7 }
 OutTextXY(MaxX-Rango+10,Trunc(Trunc(MaxY/2)*0.25),'Actividad');  { 50 }
 OutTextXY(MaxX-Rango+10,Trunc(Trunc(MaxY/2)*0.45),'Tiempo');  { 90 }
 OutTextXY(MaxX-Rango+12,Trunc(MaxY/2),'Continuar');  { 200 }
 OutTextXY(MaxX-Rango+12,Trunc(Trunc(MaxY/2)*1.10),'Otro Modo'); { 220 }
 OutTextXY(MaxX-Rango+12,Trunc(Trunc(MaxY/2)*1.20),'Salir');     { 240 }
 tit := ''; coef2 := 0;  op6 := 'C'; ContAct := ContAct + 1;
 Conj_Modos := Conj_Modos + [j];
 Str(j,op2);
 OutTextXY(MaxX-Rango+10,Trunc(Trunc(MaxY/2)*0.10),op2);  { 20 }
 Rectangle(MaxX-Rango+36,Trunc(Trunc(MaxY/2)*0.035),MaxX-Rango+92,20);   { 7,
 Correcta(MaxX-Rango+58,Trunc(Trunc(MaxY/2)*0.10),j+1,Mod,valor3); { 20 }
 MI[ContAct] := j; MF[ContAct] := valor3;
 SetColor(0);
 Rectangle(MaxX-Rango+36,Trunc(Trunc(MaxY/2)*0.035),MaxX-Rango+92,20); { 7 ,
 SetColor(GetMaxColor);
 Rectangle(MaxX-Rango+6,Trunc(Trunc(MaxY/2)*0.25),MaxX-Rango+61,Trunc(Trunc(M
 op4 := ' ';
 Repeat
   op4 := ReadChar;
   If op4 <> CEnt Then
   Begin
     tit := tit+op4;
     OutTextXY(MaxX-Rango+7+coef2*6,Trunc(Trunc(MaxY/2)*0.315),op4);  { 63 }
     coef2 := coef2+1;
   End;
 Until op4 = CEnt;
 ACT[ContAct] := tit;
 SetColor(0);
 Rectangle(MaxX-Rango+6,Trunc(Trunc(MaxY/2)*0.25),MaxX-Rango+61,Trunc(Trunc(M
 SetColor(GetMaxColor);
 Rectangle(MaxX-Rango+6,Trunc(Trunc(MaxY/2)*0.45),MaxX-Rango+51,Trunc(Trunc(M
 Correcta(MaxX-Rango+30,Trunc(Trunc(MaxY/2)*0.515),0,3000,valor4);
 T[ContAct] := valor4;
```

```
SetColor(0);
Rectangle(MaxX-Rango+6,Trunc(Trunc(MaxY/2)*0.45),MaxX-Rango+51,Trunc(Trunc(M
SetColor(GetMaxColor);
 If  Not (valor3 In Conj_Modos) Then
  Begin
   PutImage(250,Trunc(Trunc(MaxY/2)*0.10),Modos^,NormalPut);   { 20 }
   Conj_Modos := Conj_Modos + [valor3];
   xx1 := 250; yy1 := Trunc(Trunc(MaxY/2)*0.10); tam2 := 15; op5 := ' '; Vel_
   SetTextStyle(DefaultFont,HorizDir,1);
   OutTextXY(10,MaxY-13,Chr(24)+' '+Chr(25)+' '+Chr(26)+' '+Chr(27)+' = Mover
   OutTextXY(150,MaxY-15,Chr(28)); OutTextXY(155,MaxY-13,Chr(16)+' = Aceptar,
   OutTextXY(255,MaxY-13,'+/- = Velocidad del cursor');
   SetTextStyle(SmallFont,HorizDir,4);
   Repeat
     op5 := ReadChar;
     PutImage(xx1,yy1,Modos^,XOrPut);
     Case op5 Of
       CUp    : If (yy1 >= (Trunc(Trunc(MaxY/2)*0.10)+Vel_Cur)) And (yy1 <= M
         { 20, 30 }
                  yy1 := yy1 - Vel_Cur
                Else   Beep;
       CDown  : If (yy1 >= Trunc(Trunc(MaxY/2)*0.10)) And (yy1 <= MaxY-tam2-T
         { 20, 30 }
                  yy1 := yy1 + Vel_Cur
                Else   Beep;
       CRight : If (xx1 >= 5) And (xx1 <= MaxX-tam2-Rango-30-Vel_Cur) Then
                  xx1 := xx1 + Vel_Cur
                Else   Beep;
       CLeft  : If (xx1 >= 5+Vel_Cur) And (xx1 <= MaxX-tam2-Rango-30) Then
                  xx1 := xx1 - Vel_Cur
                Else   Beep;
       '+'    : If Vel_Cur <= 45 Then   Vel_Cur := Vel_Cur + 5
                Else   Beep;
       '-'    : If Vel_Cur >= 10 Then   Vel_Cur := Vel_Cur - 5
                Else   Beep;
     Else
```

```
      If op5 <> CEnt Then  Beep;
    End;
     PutImage(xx1,yy1,Modos^,NormalPut);
     Until  op5 = CEnt;
   If valor3 = Mod Then  numst := 'Sumidero'
   Else   Str(valor3,numst);
   OutTextXY(xx1,yy1+15,numst);
   Posi[valor3].x := xx1; Posi[valor3].y := yy1;
   SetColor(0);
   SetTextStyle(DefaultFont,HorizDir,1);
   OutTextXY(10,MaxY-13,Chr(24)+' '+Chr(25)+' '+Chr(26)+' '+Chr(27)+' = Mover
   OutTextXY(150,MaxY-15,Chr(28)); OutTextXY(155,MaxY-13,Chr(16)+' = Aceptar,
   OutTextXY(255,MaxY-13,'+/- = Velocidad del cursor');
   SetTextStyle(SmallFont,HorizDir,4);
   SetColor(GetMaxColor);
End;
If T[ContAct] = 0 Then   SetLineStyle(DottedLn,0,NormWidth);
Line(Posi[j].x+7,Posi[j].y+7,Posi[valor3].x+7,Posi[valor3].y+7);
SetLineStyle(SolidLn,0,NormWidth);
If ContAct = NumAct Then  op6 := 'S'
Else
Begin
  Rectangle(MaxX-Rango+8,Trunc(MaxY/2),MaxX-Rango+64,Trunc(Trunc(MaxY/2)*1.0
  Repeat
    op1 := ReadChar;
    Case op1 Of
      CUp   : Case op6 Of
                'C' : Begin
                        op6 := 'S';
                        SetColor(0);
                        Rectangle(MaxX-Rango+8,Trunc(MaxY/2),MaxX-Rango+64
                        Trunc(Trunc(MaxY/2)*1.065)+cuadsel); { 200, 213 }
                        SetColor(GetMaxColor);
                        Rectangle(MaxX-Rango+8,Trunc(Trunc(MaxY/2)*1.20),M
                        Trunc(Trunc(MaxY/2)*1.265)+cuadsel);
                      End;
```

```
            'O' : Begin
                    op6 := 'C';
                    SetColor(0);
                    Rectangle(MaxX-Rango+8,Trunc(Trunc(MaxY/2)*1.10),M
                    Trunc(Trunc(MaxY/2)*1.165));
                    SetColor(GetMaxColor);
                    Rectangle(MaxX-Rango+8,Trunc(MaxY/2),MaxX-Rango+64
                    Trunc(Trunc(MaxY/2)*1.065));  { 200, 213 }
                  End;
            'S' : Begin
                    op6 := 'O';
                    SetColor(0);
                    Rectangle(MaxX-Rango+8,Trunc(Trunc(MaxY/2)*1.20),M
                    Trunc(Trunc(MaxY/2)*1.265));
                    SetColor(GetMaxColor);
                    Rectangle(MaxX-Rango+8,Trunc(Trunc(MaxY/2)*1.10),M
                    Trunc(Trunc(MaxY/2)*1.165));
                  End;
        End;
CDown : Case op6 Of
            'C' : Begin
                    op6 := 'O';
                    SetColor(0);
                    Rectangle(MaxX-Rango+8,Trunc(MaxY/2),MaxX-Rango+64
                    Trunc(Trunc(MaxY/2)*1.065));  { 200, 213 }
                    SetColor(GetMaxColor);
                    Rectangle(MaxX-Rango+8,Trunc(Trunc(MaxY/2)*1.10),M
                    Trunc(Trunc(MaxY/2)*1.165));
                  End;
            'O' : Begin
                    op6 := 'S';
                    SetColor(0);
                    Rectangle(MaxX-Rango+8,Trunc(Trunc(MaxY/2)*1.10),M
                    Trunc(Trunc(MaxY/2)*1.165));
                    SetColor(GetMaxColor);
                    Rectangle(MaxX-Rango+8,Trunc(Trunc(MaxY/2)*1.20),M
```

```
                    Trunc(Trunc(MaxY/2)*1.265));
                  End;
            'S' : Begin
                    op6 := 'C';
                    SetColor(0);
                    Rectangle(MaxX-Rango+8,Trunc(Trunc(MaxY/2)*1.20),M
                    Trunc(Trunc(MaxY/2)*1.265));
                    SetColor(GetMaxColor);
                    Rectangle(MaxX-Rango+8,Trunc(MaxY/2),MaxX-Rango+64
                    Trunc(Trunc(MaxY/2)*1.065));{ 200, 213 }
                  End;
        End;
  Else
    If op1 <> CEnt Then  Beep;
  End;
Until  (op6 In ['C','O','S']) And (op1 = CEnt);
If op6 = 'O' Then  j := j+1;
Case op6 Of
  'C' : Begin
        SetColor(0);
        Rectangle(MaxX-Rango+8,Trunc(MaxY/2),MaxX-Rango+64,
        Trunc(Trunc(MaxY/2)*1.065));  { 200, 213 }
        SetColor(GetMaxColor);
       End;
  'O' : Begin
        SetColor(0);
        Rectangle(MaxX-Rango+8,Trunc(Trunc(MaxY/2)*1.10),MaxX-Rango+64,T
        SetColor(GetMaxColor);
      End;
  'S' : Begin
        SetColor(0);
        Rectangle(MaxX-Rango+8,Trunc(Trunc(MaxY/2)*1.20),MaxX-Rango+64,
        Trunc(Trunc(MaxY/2)*1.265));  { 240, 253 }
        SetColor(GetMaxColor);
      End;
End;
```

```
      SetFillStyle(EmptyFill,GetMaxColor);
      Bar(MaxX-Rango+2,2,MaxX-2,MaxY-2);
      SetFillStyle(SolidFill,GetMaxColor);
    End;
  Until op6 = 'S';
  SetVisualPage(0);
End;

Procedure Gra;
Begin
  ClrScr;
  Pon_Modos;
End;

Procedure Resuelve;                    {Resuelve la ruta critica
Begin                                   utilizando el algoritmo
  Conta := 1; Max := 0;                 CPM}
  For i:=1 To NumAct Do
  Begin
    If NI[i] = 1 Then  ES[i] := 0
    Else
    Begin
      Max := 0;
      For j:=1 To i-1 Do
        If NI[i] = NF[j] Then
          If EF[j] > Max Then  Max := EF[j];
      ES[i] := Max;
    End;
    If NI[i] = Conta Then
    Begin
      TE[NI[i]] := Max;
      Conta := Conta + 1;
    End;
    EF[i] := ES[i] + T[i];
  End;
  Max := 0;
```

```
  For j:=1 To NumAct Do
    If Nod = NF[j] Then
      If EF[j] > Max Then  Max := EF[j];
  For j:=NumAct DownTo 1 Do
  Begin
    If NF[j] = Nod Then  LF[j] := Max
    Else
    Begin
      Min := Max+1;
      For i:=NumAct DownTo j+1 Do
        If NI[i] = NF[j] Then
          If LS[i] < Min Then  Min := LS[i];
      LF[j] := Min;
    End;
    LS[j] := LF[j] - T[j];
  End;
  TE[Nod] := LF[NumAct];
  For i:=1 To NumAct Do
  Begin
    TF[i] := LF[i] - EF[i];
    FF[i] := TE[NF[i]] - TE[NI[i]] - T[i];
  End;
End;

Procedure Sol_Texto;                    {Procedimiento para obtener
Begin                                    el cuadro de resultados de
  ClrScr;                                la ruta crítica
  ClrScr;
  Ver(II);
  GotoXY(22,1); Write(' METODO DE LA RUTA CRITICA ');
  Ver(NN);
  GotoXY(6,3); Write('Nodos');
  GotoXY(17,3); Write('T');
  GotoXY(23,3); Write('Actividad');
  GotoXY(35,3); Write('ES');
  GotoXY(42,3); Write('EF');
```

```
  GotoXY(49,3); Write('LS');
  GotoXY(56,3); Write('LF');
  GotoXY(63,3); Write('TF');
  GotoXY(70,3); Write('FF');
  ijk := 1;
  For i:=1 To NumAct Do
  Begin
    If TF[i] = 0 Then
    Begin
      Ver(II);
      DIB[ijk] := ACT[i];
      RC[ijk] := NI[i];
      ijk := ijk + 1;
    End;
    GotoXY(5,5+i); Write(NI[i]:2);
    GotoXY(9,5+i); Write(NF[i]:2);
    GotoXY(16,5+i); Write(T[i]:2);
    GotoXY(26,5+i); Write(ACT[i]);
    GotoXY(34,5+i); Write(ES[i]:2);
    GotoXY(41,5+i); Write(EF[i]:2);
    GotoXY(48,5+i); Write(LS[i]:2);
    GotoXY(55,5+i); Write(LF[i]:2);
    GotoXY(62,5+i); Write(TF[i]:2);
    GotoXY(69,5+i); Write(FF[i]:2);
    Ver(NN);
  End;
  RC[ijk] := Nod;
  Inter := 5;
  For i:=1 To ijk-1 Do
  Begin
    GotoXY(i*Inter,23); Write('O----O');
    GotoXY(i*Inter+2,24); Write(DIB[i]);
    GotoXY(i*Inter-1,22); Write(RC[i]:2);
  End;
  GotoXY(ijk*Inter-1,22); Write(RC[ijk]:2);
{  Readln;}
```

```
  ClrScr;
End;

Procedure Sol_Grafica;                    {Procedimiento con el cual
Begin                                      se dibuja toda la gráfica
  SetActivePage(0); ClearViewPort;
  Rectangle(1,10,MaxX-1,MaxY-1); Line(1,MaxY-15,MaxX-1,MaxY-15);
{  Rectangle(MaxX-Rango,1,MaxX-1,MaxY-1); }
  OutTextXY(Trunc(MaxX/2)-Trunc(TextWidth(' METODO DE LA RUTA CRITICA ')/2),
            -3,' METODO DE LA RUTA CRITICA ');
{  GotoXY(22,1); Write(' M todo de la Ruta Cr!tica '); }
  OutTextXY(14,Trunc(Trunc(MaxY/2)*0.065),'Nodos');  { 13 }
  OutTextXY(58,Trunc(Trunc(MaxY/2)*0.065),'Tiempo'); OutTextXY(106,Trunc(Trunc(M
  OutTextXY(184,Trunc(Trunc(MaxY/2)*0.065),'T. ■ s Pr"x.'); OutTextXY(184,Trunc(
  'Inicio Act.');{ 13 }
  OutTextXY(260,Trunc(Trunc(MaxY/2)*0.065),'T. ■ s Pr"x.'); OutTextXY(260,Trunc(
  'Fin de Act.');{ 13 }
  OutTextXY(336,Trunc(Trunc(MaxY/2)*0.065),'T. ■ s Lej.'); OutTextXY(336,Trunc(T
  'Inicio Act.'); { 13 }
  OutTextXY(406,Trunc(Trunc(MaxY/2)*0.065),'T. ■ s Lej.'); OutTextXY(406,Trunc(T
  'Fin de Act.'); { 13 }
  OutTextXY(476,Trunc(Trunc(MaxY/2)*0.065),'Holgura'); OutTextXY(476,Trunc(Trunc
  ' Total'); { 13 }  { 26 }
  OutTextXY(528,Trunc(Trunc(MaxY/2)*0.065),'Holgura'); OutTextXY(528,Trunc(Trunc
  ' Indep.');  { 13 } { 26 }
  ijk := 1;
  For i:=1 To NumAct Do
  Begin
    If TF[i] = 0 Then
    Begin
      Rectangle(5,Trunc(Trunc(MaxY/2)*0.15)+i*13,573,Trunc(Trunc(MaxY/2)*0.215)+
      D]B[ijk] := ACT[i];
      RC[ijk] := N][i];
      ijk := ijk + 1;
    End;
    Str(N][i]:2,nis); Str(NF[i]:2,nfs); Str(T[i]:6,ts); Str(ES[i]:6,ess);
```

```
    Str(EF[i]:6,efs); Str(LS[i]:6,lss); Str(LF[i]:6,lfs); Str(TF[i]:6,tfs);
    Str(FF[i]:6,ffs);
    OutTextXY(13,Trunc(Trunc(MaxY/2)*0.15)+i*13,nis); OutTextXY(30,Trunc(Trunc(M
    OutTextXY(58,Trunc(Trunc(MaxY/2)*0.15)+i*13,ts); OutTextXY(106,Trunc(Trunc(M
    ACT[i]);  { 30, 30 }
    OutTextXY(188,Trunc(Trunc(MaxY/2)*0.15)+i*13,ess); OutTextXY(264,Trunc(Trunc
    efs); { 30, 30 }
    OutTextXY(340,Trunc(Trunc(MaxY/2)*0.15)+i*13,lss); OutTextXY(410,Trunc(Trunc
    lfs);  { 30, 30 }
    OutTextXY(480,Trunc(Trunc(MaxY/2)*0.15)+i*13,tfs); OutTextXY(532,Trunc(Trunc
    ffs); { 30, 30 }
  End;
  RC[ijk] := Mod;
  Inter := 5;
{  For i:=1 To ijk-1 Do
  Begin
    GotoXY(i*Inter,23); Write('O----O');
    GotoXY(i*Inter+2,24); Write(DIB[i]);
    GotoXY(i*Inter-1,22); Write(RC[i]:2);
  End;
  GotoXY(ijk*Inter-1,22); Write(RC[ijk]:2);}
  Readln;
  SetVisualPage(0);
{  ReadLn;}
{  SetVisualPage(0);}
{  ReadLn;}
  ClearViewPort;
End;

Procedure Pon_Opciones;                    (Procedimiento con el cual
Begin                                       se marcan las  opciones y
      SetTextStyle(DefaultFont,HorizDir,1);  mediante que  teclas
      OutTextXY(10,MaxY-13,Chr(24)+' '+Chr(25)+' '+Chr(26)+' '+Chr(27)+' = Mover
      OutTextXY(150,MaxY-15,Chr(28)); OutTextXY(155,MaxY-13,Chr(16)+' = Aceptar,
      OutTextXY(255,MaxY-13,'+/- = Velocidad del cursor');
      SetTextStyle(SmallFont,HorizDir,4);
```

```
      Repeat
        op1 := ReadChar;
        Case op1 Of
          CUp   :
        Else
          If op1 <> CEnt Then  Beep;
        End;
      Until  (op6 In ['C','O','S']) And (op1 = CEnt);
End;

Begin
  Gra;
  If (ContAct = NumAct) Then
  Begin
    Resuelve;
    Pon_Opciones;
    Sol_Grafica;
    CloseGraph;
  End;
End.

{           Fin  del   Programa   CPM.PAS                          }
```