DIVISION DE ESTUDIOS DE POSGRADO
FACULTAD DE INGENIERIA

EL PROBLEMA DEL CARTERO CHINO Y SUS EXTENSIONES

MIGUEL ANGEL GUTIERREZ ANDRADE

TESIS

PRESENTADA A LA DIVISION DE ESTUDIOS DE

POSGRADO DE LA

FACULTAD DE INGENIERIA

DE LA

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

COMO REQUISITO PARA OBTENER

EL GRADO DE

MAESTRO EN INGENIERIA

(INVESTIGACION DE OPERACIONES)

CIUDAD UNIVERSITARIA

MEXICO, D.F., ABRIL DE 1988

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

R E S U M E N

Un problema clásico de la teoría de gráficas, es el denominado problema del cartero chino consistente en recorrer todos los arcos o aristas de una gráfica a costo mínimo. Dicho problema tiene aplicaciones inmediatas tales como: el trazado de rutas de camiones de servicio público para la recolección de desechos sólidos; en el reparto de artículos a una comunidad; en el recorrido de las plumas de un graficador para el trazado de mapas. En este trabajo se describe la implantación de un algoritmo polinomial para la determinación de la ruta óptima en una red o gráfica con arcos dirigidos o no dirigidos y con pesos o costos asociados al paso de cada arco. Una aplicación al problema de desechos sólidos se anexa.

C O N T E N I D O

I N T R O D U C C I O N

Una de las ramas de la ciencia que tiene un amplio campo de aplicación es la Teoría de Gráficas. Existen numerosas aplicaciones en ramas tan diversas como: ingeniería, química, planeación, educación, administración y sicología, entre otras. Quizá el uso extenso de las gráficas se deba en parte al dicho que dice: "Un dibujo dice más que mil palabras", ya que el lenguaje de las gráficas trata estrechamente con la filosofía de esta idea. A menudo, la manera natural de reflexionar cuando nos enfrentamos a un problema abstracto es dibujar en una hoja de papel puntos que puedan representar individuos, lugares, o cosas y conectarlos por medio de lineas o flechas que simbolicen una cierta relación. Esta representación de la realidad por medio de figuras tiene dos ventajas:

(1) Se puede expresar la estructura análitica de la situación dada.

(2) Desde un punto de vista práctico, presenta una visión global del problema; que da una guía valiosa para nuestra intuición y razonamiento.

El primer artículo sobre teoría de gráficas lo escribió Euler en 1736 sobre un problema de como atravesar una y sólamente una vez cada uno de los siete puentes de la ciudad de Königsberg. Una generalización de este primer problema de la Teoría de Gráficas es el Problema del Cartero Chino. El problema puede describirse como

sigue: un cartero debe recoger sus cartas en la oficina postal e ir entregándolas a lo largo de su ruta para finalmente regresar a la oficina postal. El cartero desea conservar su energía y por lo tanto quiere cubrir su zona recorriendo la ruta más corta posible. En términos gráficos, el problema del cartero es un problema de como cubrir todas las calles en su zona y regresar al punto de partida con la menor distancia total recorrida. Obviamente no únicamente un cartero se enfrenta a tal problema, este esquema es común en otro tipo de problemáticas tales como: un policía desea conocer el camino más eficiente para patrullar todas las calles de su delegación; un granjero desea conocer la mejor ruta para sembrar sus campos; un camión recolector de basura desea conocer la mejor forma de barrer su zona; para el trazado de una red de carreteras en un polo de desarrollo, etc.

La solución de este problema es sencilla, usando el método exhaustivo, cuando el número de calles o posibles rutas es pequeño. Sin embargo, el problema se vuelve complicado cuando dicho número aumenta. En tal caso es conveniente la introducción de técnicas de optimización que permitan la determinación óptima de recorrido que deba hacerse. En años recientes, diversos esfuerzos se han realizado en el área de Investigación de Operaciones para resolver problemas de determinación de rutas óptimas.

Antes de decidir que modelo o algoritmo hay que usar, se debe conocer cuál es el objetivo o criterio para la selección de la ruta más corta. Algunos pueden ser: cumplir con determinado rango de horario en cada sitio; minimizar la distancia total recorrida; lograr el menor tiempo de recorrido; o incluso cualquier combinación de los objetivos anteriores. Sea cual fuere la función objetivo que se establezca, así como las restricciones al sistema, lo más probable es que el número de opciones sea muy grande; por ejemplo, si tenemos una ruta en la que se deben recorrer 10 calles y cualquier sentido en que se recorran es válido, el número de combinaciones (soluciones factibles), es 20! (veinte factorial), cantidad que seria imposible de analizar en un tiempo razonable, aún por la computadora más veloz del mundo; ya que si se supone que la computadora pueda programarse para examinar las soluciones factibles de este problema a razón de mil millones de soluciones por segundo, entonces la computadora terminaría su tarea, en más de 77 años, y para 11 calles en alrededor de 35642 años. Con lo anterior se pretende destacar la importancia y necesidad de utilizar algoritmos y soluciones analíticas a los problemas de diseño de rutas a fin de obtener una o más soluciones "óptimas".

El problema de determinación de rutas óptimas puede ser atacado usando diversos enfoques. Uno de estos es en términos de programación entera, sin embargo, los correspondientes métodos de solución resultan inadecuados para casos prácticos. Un enfoque que

ha demostrado ser útil es el de teoría de gráficas, dado que sus métodos de solución resultan sencillos y fáciles de implantar en la computadora. En este enfoque conviene analizar el problema en cuestión, en términos de gráficas no dirigidas o dirigidas, dependiendo de la restricción que se tenga en el sentido de recorrer una calle o avenida (en una dirección o en ambas).

El problema del cartero chino puede transformarse de manera natural en un problema de optimización en una gráfica; donde las calles o caminos pueden transformarse en arcos, sus uniones y cruces en nodos y la distancia viajada por el cartero al ir recorriendo cada calle, en la longitud o peso asociado a este arco. De modo que para la gráfica asociada, el problema es identificar el ciclo que atraviesa cada arco al menos una vez y cuya distancia total es mínima.

Las gráficas más simples que se pueden estudiar en el problema del cartero chino, son aquellas en las que hay una ruta que atraviesa cada arco exactamente una vez. La longitud total de una ruta de este tipo, es la suma de todas las longitudes de los arcos en la gráfica y es evidente que no existe otra ruta de longitud más corta. Este caso especial no se ve afectado por la longitud de los arcos, ya que podemos alterar las longitudes de los arcos y la ruta seguirá siendo óptima. Para este caso especial no es necesario el uso de las distancias de cada arco. La ruta que

satisface estas restricciones se conoce como ruta de Euler o Euleriana. Para este tipo de gráficas, el problema de encontrar una ruta, se reduce a encontrar una ruta de Euler. Un problema más difícil es encontrar una ruta del cartero chino en gráficas donde no existan rutas de Euler o alguno o todos los arcos son dirigidos. En el caso que se tenga una gráfica en la que no existan rutas de Euler, el objetivo es seleccionar algunos arcos en la gráfica que deban recorrerse más de una vez de tal forma que se recorran todos los arcos en la gráfica al menos una vez y que hagan la distancia total recorrida mínima, así que se debe encontrar las distancias más cortas entre cada par de nodos de grado impar y aparejar o acoplar estos nodos de modo que el acoplamiento total sea de longitud mínima.

El objetivo de este trabajo es describir y analizar el problema del cartero chino para gráficas dirigidas y no dirigidas, con especial énfasis en algoritmos de solución polinomiales y su correspondiente implantación en PC, usando estructuras de datos recientes que resultan muy eficientes; también se incluye la aplicación a un caso práctico.

Conviene señalar que desde la década de los 50's, el problema de acoplamiento se ha estudiado desde un punto de vista técnico y complicado y los resultados obtenidos han aparecido en revistas especializadas. La importancia de los capítulos donde se

desarrollan los problemas de acoplamiento, radica en la sencillez de la presentación, la facilidad para implementar el algoritmo en una computadora y según extensos estudios muestran que estos algoritmos son muy rápidos y para propósitos prácticos, probablemente sean los algoritmos más rápidos de todos los algoritmos de acoplamiento conocidos; esta aseveración también se extiende al caso del algoritmo para resolver el problema del cartero chino en gráficas no dirigidas ya que la parte que consume más tiempo es la del acoplamiento entre nodos de grado impar.

Este trabajo se desarrolla como sigue: En el capítulo 1 se dan las definiciones básicas que se emplearán a lo largo de todo el trabajo; la forma de almacenar eficientemente en la computadora una gráfica y el concepto de complejidad computacional. El capítulo 2 trata el problema de acoplamiento de máxima cardinalidad. En el capítulo 3 se trata el problema de acoplamiento pesado necesario para la solución del problema del cartero chino en gráficas no dirigidas. En el capítulo 4 se plantea el problema del cartero chino en gráficas no dirigidas, es decir, para el caso en que se pueden transitar las calles (arcos) en ambos sentidos así como algunas alternativas de solución (algoritmos) y ejemplos ilustrativos. En el capítulo 5 se hace lo mismo que en el capítulo 4 pero considerando que la gráfica es dirigida, esto es, cuando las calles únicamente se pueden transitar en un sólo sentido. En el capítulo 6 se tienen la

conclusiones. Finalmente, en el apéndice, se dan las entradas de datos y algunas características de los programas desarrollados en PASCAL y FORTRAN de los algoritmos que se presentaron en el texto.

C A P I T U L O 1
ASPECTOS BASICOS

Los problemas de acoplamiento y del cartero chino tratados en el presente trabajo, caen dentro de las disciplinas conocidas como optimización combinatoria, teoría de gráficas y teoría de redes, tres áreas de las matemáticas aplicadas relativamente recientes que tienen diferente filosofía y forma de tratar los problemas. La diversidad de tales disciplinas y enfoques de análisis da origen a que no exista una notación y terminología homogénea. Para algunos autores un término puede significar algo completamente distinto del significado que le dan otros; asímismo, términos distintos pueden significar lo mismo. Debido a esto, el presente capítulo establece un mínimo de terminología y el significado que se dará en los capítulos posteriores.

El desarrollo del capítulo es como sigue: en la sección 1.1 se establecen algunas definiciones de los conceptos básicos empleados a lo largo del trabajo. En la sección 1.2 se definen y comentan las estructuras específicas que requieren los problemas de acoplamiento y cartero chino. En la sección 1.3 se tratan aspectos de la representación y manejo de gráficas en la computadora, dichas estructuras son las empleadas en los programas de computadora implantados en este trabajo. La sección 1.4 comenta el problema de la complejidad computacional de los algoritmos, esto

es, como se incrementa la memoria y los tiempos requeridos para resolver un problema en la computadora conforme se incrementa el tamaño del mismo.

1.1 Definiciones

Una gráfica $G = (N,A)$ consiste de un conjunto finito de **nodos** $N = \{v_1, v_2,\ldots,v_n\}$ y un conjunto finito de **arcos** $A = \{a_1, a_2,\ldots,a_m\}$. A cada arco a le corresponde un par de nodos distintos (u,v) a los que se dice que **incide**. Cada nodo de la gráfica se representa por un punto y cada arco por una linea que une sus dos nodos terminales (Vea figura 1.1). Una gráfica se dice que es **dirigida** o **digráfica** si la pareja de nodos (u,v) , también llamada arco, asociados a cada arco a es un par ordenado. El arco a se dice que está dirigido del nodo u al nodo v, y la dirección se representa por una flecha en el arco (Vea figura 1.2). Una gráfica se dice no dirigida si sus nodos terminales no están ordenados ,es decir, si sus arcos no tienen dirección. Una gráfica dirigida o no dirigida se dice **pesada** si se asigna un número real a cada arco de la gráfica. Este número generalmente se refiere como el **peso** del arco. En la práctica, este número puede representar conceptos tales como distancia, tiempo, costo, o cualquier otro atributo del arco.

Dos arcos se dicen **paralelos** si tienen el mismo par de nodos terminales (y adicionalmente, si tienen la misma dirección, en el

caso de una gráfica dirigida). A través de este trabajo, supondremos que las gráficas no tienen arcos en paralelo (a menos que no se diga lo contrario); así, se podrá hacer referencia a cada arco por medio de sus nodos terminales, por ejemplo en la figura 1.1 el arco a_3 también puede hacerse referencia como (v_2, v_5).

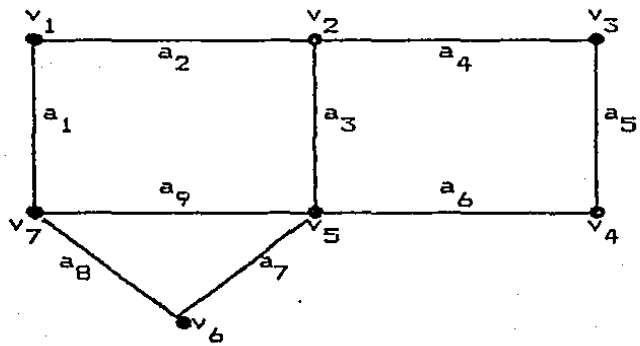


Figura 1.1 Gráfica no dirigida con 7 nodos y 9 arcos.

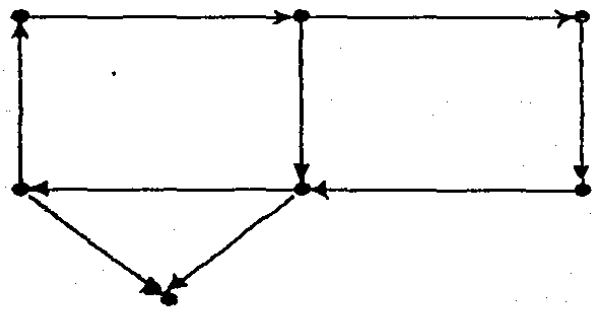


Figura 1.2 Gráfica dirigida

1.2 Estructuras especiales

Existen diversas clases de gráficas especiales que conviene mencionar así como algunas componentes de la misma que es necesario tipificar. En una gráfica dirigida dado un arco (u,v) (dirigido del nodo u al nodo v) se dice que v es un sucesor inmediato de u, y u es el predecesor inmediato de v. Si el arco (u,v) no es dirigido, entonces se dice que u y v son adyacentes. Una trayectoria simple o trayectoria P de un nodo v_1 a otro nodo v_k en una gráfica G es una sucesión de arcos (v_1,v_2), (v_2,v_3),..., (v_{k-2},v_{k-1}), (v_{k-1},v_k), algunas veces escrito, por simplicidad como $(v_1,v_2,\ldots,v_k)$, en donde todos los nodos $v_1,v_2,\ldots,v_k$ son distintos. En una gráfica dirigida, esta trayectoria se dice que está dirigida de v_1 a v_k; en una gráfica no dirigida se dice que es una trayectoria entre v_1 y v_k. El peso de la trayectoria P en la gráfica, es la suma de los pesos de los arcos de P. Por ejemplo, en la gráfica de la figura 1.3, la sucesión de arcos (5,2), (2,1), (1,3) es una trayectoria del nodo 5 al 3 que consta de tres arcos y tiene peso de 70. Un sólo arco (u,v) claramente es una trayectoria de u a v. Un ciclo está definido de la misma manera que una trayectoria, excepto que el primer nodo v_1 y el último nodo v_k son iguales. En la gráfica de la figura 1.3, la sucesión de arcos (2,1), (1,5), (5,2) es un ciclo. Una gráfica que no contiene ciclos se llama acíclica.

En general, en una gráfica existen muchas trayectorias de un nodo

s a otro nodo t. Entre todas las trayectorias de s a t, la que tiene el peso menor, se le llama **trayectoria más corta de s a t**. Claramente, pueden existir varias de estas trayectorias entre s y t.

Una **subgráfica** de una gráfica G = (N,A), es una gráfica cuyos nodos y arcos están en G. Una **gráfica no dirigida** G se dice que es **conectada** si para todo par de nodos v_i y v_j en G, existe al menos una trayectoria. Una **gráfica dirigida** G se dice que es **conectada** si es conectada la gráfica no dirigida obtenida de G al ignorar las direcciones de todos sus arcos.

Una gráfica conectada, no dirigida y acíclica se llama un **árbol** y puede mostrarse que existe solamente una trayectoria entre cada par de nodos en el árbol. Por lo tanto agregando un arco al árbol se forma exactamente un ciclo. Puede también mostrarse que un árbol con n nodos tiene exactamente n-1 arcos.

Un **árbol de expansión** T de una gráfica no dirigida y conectada es una subgráfica de G que es un árbol y contiene a todo nodo de G. Por ejemplo en la gráfica de la figura 1.3, un árbol de expansión se muestra con lineas más fuertes. El **peso del árbol de expansión** T en una red conectada, es la suma de los pesos de los arcos de T. El peso del árbol de expansión de la figura 1.3 es 131. En general, dada una gráfica conectada existen diversos árboles de expansión y a menudo se desea encontrar uno que tenga peso mínimo.

A tal árbol se le llama árbol de peso mínimo.

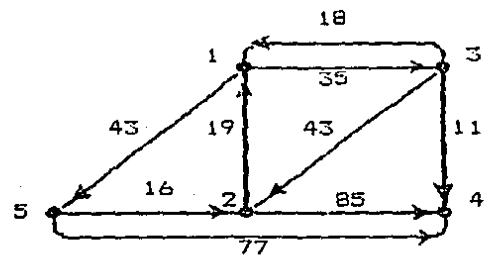


Figura 1.3 Gráfica con pesos en sus arcos

1.3 Representación de una gráfica en la computadora

Uno de los aspectos más importantes en la manipulación de una gráfica es su representación en computadora. La más simple y quizás la más popular representación en la computadora de una gráfica es la matriz pesada. La matriz pesada o la matriz nodos-nodos de una gráfica, es una matriz $W = [w_{ij}]$ de orden nxn, en donde w_{ij} es el peso del arco (i,j) en G. Si no existe arco del nodo i al nodo j en G entonces el elemento correspondiente w_{ij} se hace ∞ (en la práctica un número muy grande). Los elementos de la diagonal principal también son ∞. Es fácil observar que la matriz pesada de una gráfica no dirigida es siempre simétrica. La matriz pesada de la gráfica de la figura 1.3 es:

$$W = \begin{bmatrix} \infty & \infty & 35 & \infty & 43 \\ 19 & \infty & \infty & 85 & \infty \\ 18 & 43 & \infty & 11 & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & 16 & \infty & 77 & \infty \end{bmatrix}$$

Es claro que una matriz pesada requiere de n^2 palabras de memoria de la computadora para guardar la gráfica. Una gráfica no dirigida usualmente sólo requiere la mitad de almacenamiento ya que la matriz es simétrica. Si la gráfica tiene mucho menos de n*(n-1) arcos, el número máximo de arcos posibles, entonces es mucho más eficiente (desde el punto de vista de ahorro de memoria) guardar la lista de los m arcos en tres vectores, uno con los nodos iniciales, otro con los nodos finales y el tercero con los pesos. Esta representación puede implementarse por medio de tres arreglos: $B=(b_1, b_2,\ldots,b_m)$, $D=(d_1, d_2,\ldots,d_m)$ y $Z=(z_1, z_2,\ldots, z_m)$. El arco i va del nodo b_i al nodo d_i con un peso z_i. Por ejemplo, la gráfica de la figura 1.3 puede representarse como:

B = (1,1,2,2,3,3,3,5,5)
D = (3,5,1,4,1,2,4,2,4)
Z = (35,43,19,85,18,43,11,16,77)

Note que la lista de arcos está en orden ascendente con respecto a su nodo inicial. Pero pueden listarse en cualquier otra forma diferente al orden anterior. Este método de representación requiere de 3m palabras de memoria de computadora, en contraste

con n^2 palabras de memoria que se requieren para la matriz pesada.

En algunos algoritmos el procedimiento para guardar una red es más eficiente si todos los arcos que salen de un nodo son agrupados juntos. En este tipo de representación sólamente se necesita guardar cada arco por medio de su nodo terminal (y su respectivo peso) ya que el nodo inicial implícitamente está definido. Esta representación está implementada convenientemente por medio de n listas ligadas, una para cada nodo i. Todos los arcos que salen del i-ésimo nodo están ligados juntos. Así, cada arco requiere tres campos: uno para el nodo destino del arco, otro para el peso del arco y otro más para apuntar al siguiente arco de la lista (que sale del mismo nodo i). Cada lista ligada tiene una cabeza que contiene el punto a que apunta el primer arco que sale del nodo correspondiente. Estas cabezas se guardan secuencialmente en un arreglo, ya que en un procedimiento típico se empezará aleatoriamiento en el nodo k (y se accesará todos sus sucesores inmediatos.).La figura 1.4 muestra una lista ligada adyacente de la gráfica de la figura 1.3. Esta representación requiere 3m + n palabras de almacenamiento. El almacenamiento requerido es ligeramente mayor que el almacenamiento para una lista da arcos, pero la liga de estos arcos proporcionan mayor flexibilidad y facilidad cuando se desea agregar o quitar arcos de la representación.

La lista ligada hacia adelante es una variación de la lista ligada adyacente. Si en el proceso sobre la gráfica no se requiere agregar ni quitar arcos, entonces se pueden ahorrar los apuntadores para cada uno de los arcos en la representación anterior y simplemente guardar los arcos secuencialmente. (Todos los arcos que salen del nodo k se guardan inmediatamente después de todos los arcos que salen del nodo k-1). Por ejemplo, la lista ligada hacia adelante para la gráfica de la figura 1.3, se muestra en la figura 1.5 usando tres arreglos. Observe que todos los arcos

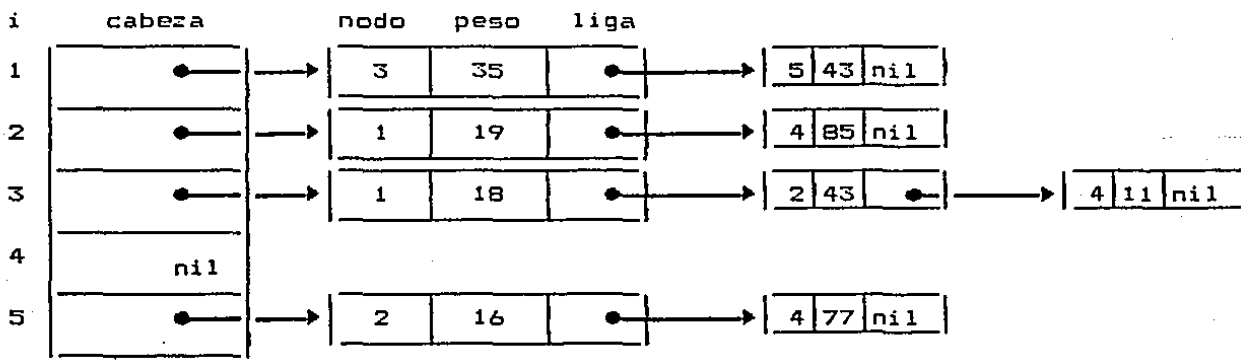


Figura 1.4 Lista ligada adyacente de la fig. 1.2.

que salen del nodo i son: (i, NODO_FIN[APUNTADOR[i]]), (i, NODO_FIN[APUNTADOR[i] +1]),...,(i, NODO_FIN[APUNTADOR[i + 1] - 1]). Note que el caso i = n, se toma cuidadosamente agregando una celda artificial al arreglo APUNTADOR que contiene el valor (m + 1). En forma análoga a la lista ligada hacia adelante, una gráfica puede representarse por una lista ligada hacia atrás, donde todos

los arcos que entran a un nodo k pueden agruparse juntos.

Para una gráfica dirigida de n nodos y m arcos, la representación en una lista ligada hacia adelante requiere, de (n + 1) + 2m palabras de memoria. Con esta representación se han ganado m palabras de memoria, con respecto a la lista ligada, a expensas de alguna flexibilidad. (Ya que todos los arcos se han guardado en forma secuencial, el agregar o quitar arcos no puede realizarse fácilmente.)

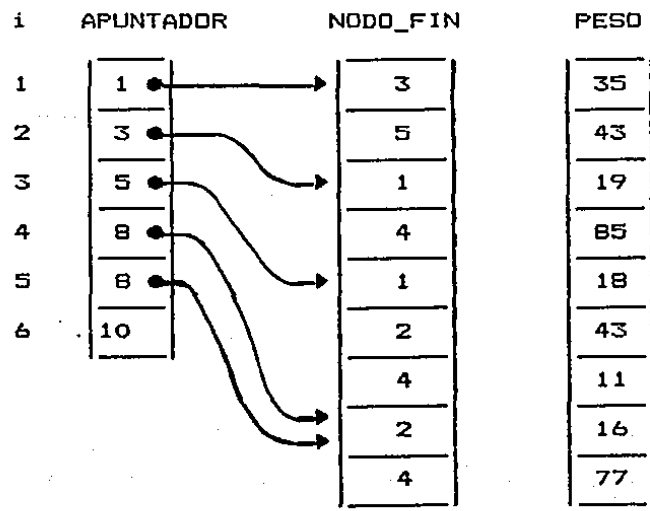


Figura 1.5 Representación hacia adelante de la fig. 1.2.

Estas son las cuatro estructuras de datos más frecuentes para gráficas. Existen otras variaciones pero en este trabajo,

los arcos que entran a un nodo k pueden agruparse juntos.

Para una gráfica dirigida de n nodos y m arcos, la representación en una lista ligada hacia adelante requiere, de (n + 1) + 2m palabras de memoria. Con esta representación se han ganado m palabras de memoria, con respecto a la lista ligada, a expensas de alguna flexibilidad. (Ya que todos los arcos se han guardado en forma secuencial, el agregar o quitar arcos no puede realizarse fácilmente.)

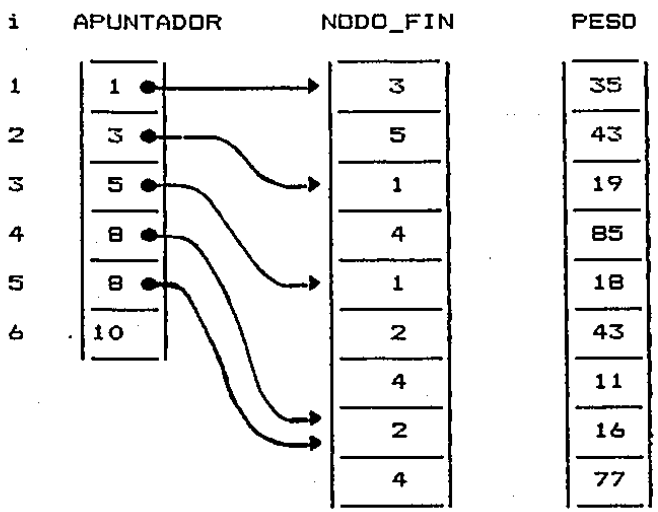


Figura 1.5 Representación hacia adelante de la fig. 1.2.

Estas son las cuatro estructuras de datos más frecuentes para gráficas. Existen otras variaciones pero en este trabajo,

unicamente se emplearán las ya descritas. No existe una representación de una gráfica en la computadora que sea mejor que otra. La elección depende de la naturaleza del proceso que se realiza en la gráfica, tamaño, lenguaje que se use, entre otras características.

1.4 Complejidad computacional.

En este trabajo se presentan tres tipos de problemas en gráficas: el problema de acoplamiento, el problema del cartero chino y el problema de flujo a costo mínimo (usado para resolver el problema del cartero chino en gráficas dirigidas). Los dos primeros problemas son de naturaleza combinatoria y el tercero se puede resolver, resolviendo su dual, que también es de naturaleza combinatoria. El conjunto de soluciones de cada uno de estos es finito. En una gráfica con n nodos y m arcos hay a lo más 2^n posibles acoplamientos, no más de (2m)! rutas del tipo requerido por el cartero chino, y no más de 2^n cortes (El problema de corte mínimo o cortadura mínima, es el problema dual del problema de flujo máximo. Véase R16). Cada uno de estos problemas se puede resolver haciendo un programa que liste todas las soluciones posibles y escoja la mejor de ellas. Se podría pensar que los problemas anteriores no ofrecen el menor reto; ya que para su solución, se requiere la consideración de un número finito de posibilidades y el problema matemáticamente es trivial.

Esta linea de razonamiento encuentra diversas dificultades cuando se tiene la necesidad de encontrar la solución óptima a alguno de estos problemas. Por ejemplo, suponga que m=10 y que la computadora pueda programarse para examinar las soluciones factibles del problema del cartero chino a razón de mil millones de soluciones por segundo, entonces si hay (2m)! soluciones posibles, la computadora terminaría su tarea, en más de 77 años, para m = 11 en alrededor de 35642 años. Este tipo de soluciones desde el punto de vista práctico no es recomendable. El reto de los estudiosos de problemas de carácter combinatorio, es el de desarrollar algoritmos de búsqueda para los que se tenga un número de operaciones elementales, aceptablemente pequeño; así como los de este trabajo, el de desarrollar e implantar programas en la computadora que lleven a una solución aceptable de los problemas antes mencionados. En este contexto, una solución aceptable a un problema, se refiere a que para obtener la solución del problema los requerimientos de almacenamiento y el tiempo de proceso sean "razonables".El problema de la complejidad computacional se refiere a cúando un problema esta bien resuelto o mal resuelto, desde el punto de vista computacional. Específicamente, un algoritmo se considera "bueno", si el número de pasos computacionales elementales que requiere para su ejecución, está acotado por un polinomio en el tamaño del problema.

Una función polinomial crece mucho menos rápido que una función

exponencial, y una función exponencial crece mucho menos rápido que una función factorial.

Suponga que un algoritmo para resolver un problema de gráficas, requiere de $100n^3$ operaciones y otro requiere de 2^n operaciones, en donde n es el número de nodos. El algoritmo exponencial es más eficiente para gráficas con menos de 17 nodos. Para gráficas mayores de 17 nodos, el algoritmo polinomial es más rápido. Un problema con 50 nodos para este algoritmo polinomial, es computacionalmente factible de resolver, mientras que para el algoritmo exponencial es imposible.

En general, si n es el tamaño del problema y el número de pasos computacionales requeridos por un cierto algoritmo es:

$$a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0$$

donde $a_k > 0$, entonces decimos que el algoritmo es "de orden n^k " y se escribe $O(n^k)$. Aquí no interesa la magnitud del coeficiente a_k. Por lo tanto, preferiremos un algoritmo $O(n^k)$ a cualquier otro algoritmo $O(n^{k+1})$; la razón se debe a que para valores grandes de n el algoritmo de orden $O(n^k)$ será más eficiente que cualquier algoritmo $O(n^{k+1})$. Por consiguiente, preferiremos un algoritmo $O(n^3)$ que requiere de $10^{10}n^3$ operaciones sobre un algoritmo $O(n^4)$ que requiere de $10n^4 + 20n^3$ operaciones, solamente porque el algoritmo $O(n^3)$ requiere de menor tiempo de ejecución a partir de

una n grande. En la práctica, es raro que uno se enfrente a una alternativa tan extrema como la anterior, cuando se obtiene un algoritmo de un exponente menor, por lo general, su coeficiente tiene un tamaño aceptable.

Es importante notar, que se ha supuesto que todas las operaciones requieren de una unidad de tiempo, sin importar el tamaño de los operandos. Esta suposición no es válida rigurosamente; sin embargo, para los fines planteados anteriormente, los algoritmos van a seguir teniendo el mismo orden cuando se hagan los cálculos tomando en cuenta las distintas operaciones a que haga referencia.

C A P I T U L O 2
ACOPLAMIENTO MAXIMO

Un acoplamiento en una gráfica es cualquier conjunto de arcos de la gráfica tal que a cada nodo incide a lo más un arco. Entre los problemas de acoplamiento que son importantes para la solución del problema del cartero chino, están: el acoplamiento de máxima cardinalidad y el acoplamiento de peso máximo o pesado. En la literatura sobre el tema, estos problemas se resuelven primeramente para gráficas bipartitas y posteriormente para cualquier tipo de gráfica. La razón de que se resuelva primero en gráficas bipartitas, se debe a lo simple de la solución, ya que el problema de acoplamiento de máxima cardinalidad se puede resolver como un problema de flujo máximo [R16] y el problema de acoplamiento de peso máximo se puede plantear como un problema lineal cuyas restricciones permiten soluciones enteras [R14]. La estructura de una gráfica bipartita permite que haya algoritmos especializados, relativamente simples para la solución de los problemas anteriores [R11 y R16]. Desgraciadamente, para resolver el problema del cartero, se requiere dar solución a un problema de acoplamiento de peso máximo en una gráfica que no es bipartita.

Conviene señalar que desde la década de los 50's, el problema de acoplamiento se ha estudiado desde un punto de vista técnico y complicado y los resultados obtenidos han aparecido en revistas especializadas. La importancia de este capítulo radica en la sencillez de la presentación, la facilidad para implementar el

algoritmo en una computadora y según estudios extensos realizados por Derigs [R3] muestran que este algoritmo es muy rápido y para propósitos prácticos, problamente sea el algoritmo más rápido de todos los algoritmos de acoplamiento conocidos.

El desarrollo del capitulo es como sigue: en la sección 2.1 se describen los problemas de acoplamiento y de acoplamiento máximo y se dan algunas definiciones, también se presentan los resultados básicos que caracterizan los acoplamientos; en la sección 2.2 se detalla la forma que se manejan las floraciones y los criterios que se siguen en el crecimiento de los árboles alternantes; en la sección 2.3 se describe el algoritmo de acoplamiento máximo; en la sección 2.4 se desarrolla un ejemplo del algoritmo de acoplamiento máximo y finalmente en la sección 2.5 se establece la complejidad computacional del algoritmo descrito en la sección 2.3.

2.1 Descripción y resultados básicos

Un conjunto M de arcos en una gráfica no dirigida G=(N,A) se dice que es un **acoplamiento** si no existen dos arcos en M que tengan un nodo en común. Por ejemplo, en la gráfica de la figura 2.1, el conjunto de arcos {(1,5),(2,3),(7,8)} es un acoplamiento, ya que no existen dos arcos que incidan a un mismo nodo. El conjunto {(1,5),(2,6),(3,7)} es otro acoplamiento. Pero el conjunto {(2,3),(3,4)} no es un acoplamiento porque los arcos tienen un nodo común. Un sólo arco en una gráfica obviamente es un acoplamiento.El problema de acoplamiento de **máxima cardinalidad,** consiste en encontrar un acoplamiento en la gráfica con el mayor

número de arcos posible.

En un acoplamiento M, se dice que un arco está **acoplado** si está en M y **no acoplado** si no está en M. Similarmente un nodo x se dice que está **acoplado** o **saturado** si es un nodo terminal de algún arco (x,y) que pertenece al acoplamiento. Los nodos terminales del arco (x,y) en M se dice que son **compañeros**. Un nodo que no está acoplado se llama **expuesto** o **libre**. En la figura 2.1, los nodos 1, 2, 3, 5, 7, 8 son nodos acoplados o saturados, mientras que los nodos 4 y 6 son expuestos o libres (con respecto al acoplamiento M = {(1,5),(2,3),(7,8)}, marcado con lineas más fuertes en la figura 2.1).

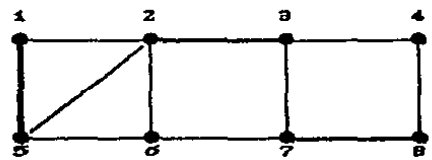


Figura 2.1.

Una **trayectoria alternante** con respecto a un acoplamiento M, es una trayectoria simple en la que se alternan arcos contenidos y no contenidos en M. Por ejemplo, para el acoplamiento de la figura 2.1, la trayectoria (1,2,3,4) es una trayectoria alternante, también las trayectorias (1,5,2,3) y (4,8,7,3,2,6) son trayectorias alternantes; pero (5,6,7,8) no es una trayectoria alternante, ya que los arcos (5,6) y (6,7) no están en el acoplamiento.

Una **trayectoria aumentante** es una trayectoria alternante en la que sus nodos inicial y final están expuestos. Por ejemplo en la figura 2.1, la trayectoria (6,7,8,4) es una trayectoria aumentante; también lo son las trayectorias (6,5,1,2,3,4) y (4,8,7,3,2,6).Si en una trayectoria aumentante se intercambian los arcos que están en el acoplamiento por los arcos que no están en el acoplamiento, entonces se obtiene, un acoplamiento que contiene un arco más que el acoplamiento original. En consecuencia, si un acoplamiento tiene una trayectoria aumentante entonces el acoplamiento no puede ser de máxima cardinalidad y el resultado inverso también es válido. Esta idea se resume en el siguiente teorema debido a Berge [1957].

Teorema 2.1. Un acoplamiento M no es de máxima cardinalidad, si y sólo si el acoplamiento M tiene una trayectoria aumentante.

Prueba: Si existe una trayectoria aumentante, claramente el acoplamiento M no es de máxima cardinalidad. Suponga que M no es un acoplamiento de máxima cardinalidad. Sea M' un acoplamiento de máxima cardinalidad de modo que $|M'| > |M|$. También sea G' la gráfica formada por los nodos de G y los arcos que están en M o en M' pero no en ambos. Note que:

(a) G' tiene más arcos de M' que de M.

(b) a cada nodo de G' incide a lo más un arco de M' y a lo más un arco de M.

Se sigue que cada componente de G' o es un nodo aislado o una

trayectoria alternante. Sin embargo al menos una componente debe tener más arcos de M' que de M (por (a)). Esta componente es una trayectoria aumentante en M. ■

Este teorema es la base de todos los algoritmos para el problema de acoplamiento y sugiere el siguiente método para encontrar un acoplamiento máximo en una gráfica G. Comenzar con un acoplamiento arbitrario M en una gráfica G dada (que puede ser vacío). Encontrar una trayectoria aumentante P con respecto a M. Entonces construir otro acoplamiento M' = (M-P)U(P-M). Claramente la cardinalidad de M' es mayor que la de M. Luego empiece con M', encontrando una trayectoria aumentante con respecto a M' y proceda como antes. Se repite este procedimiento hasta que se encuentra un acoplamiento que no tenga trayectorias aumentantes. Este acoplamiento final es un acoplamiento de cardinalidad máxima.El método descrito anteriormente puede consumir mucho tiempo a no ser que se tenga una manera sistemática y eficiente de encontrar trayectorias aumentantes ya que el número de posibles trayectorias crece exponencialmente conforme crece el tamaño de la gráfica.

Una solución elegante al problema anterior fué proporcionada por Edmonds [1965] y consiste en lo siguiente: para encontrar una trayectoria aumentante con respecto a un acoplamiento M, comience en cualquier nodo expuesto r. Si existe otro nodo expuesto s adyacente a r, agregue el arco (r,s) al acoplamiento actual M. Si no existe ningún nodo expuesto adyacente a r, entonces todos los nodos adyacentes a r están acoplados, se hará crecer un árbol

enraizado en r (vea la gráfica de la figura 2.2). Sea r la raíz situada en un nivel 0 y todos los nodos adyacentes a_1, a_2,...,a_q, en un nivel 1 (Los nodos a_1, a_2,...,a_q, están saturados.). Ahora incluya en este árbol todos los arcos acoplados incidentes en a_1, a_2,...,a_q, y situe sus respectivos nodos terminales b_1, b_2,...,b_q, en el nivel 2. A continuación, todos los nodos adyacentes a los b_i (estos nodos no están contenidos en el árbol) se colocan a un nivel 3. Se continua en cada uno de los b_i como al principio en r. Un árbol construido en la forma anterior se llama **árbol alternante**. Más formalmente, un árbol alternante con respecto a un acoplamiento M, es un árbol enraizado en un nodo expuesto que tiene la propiedad que todas las trayectorias que principian en la raíz son trayectorias alternantes (Existe sólamente una trayectoria entre cada par de nodos en el árbol).

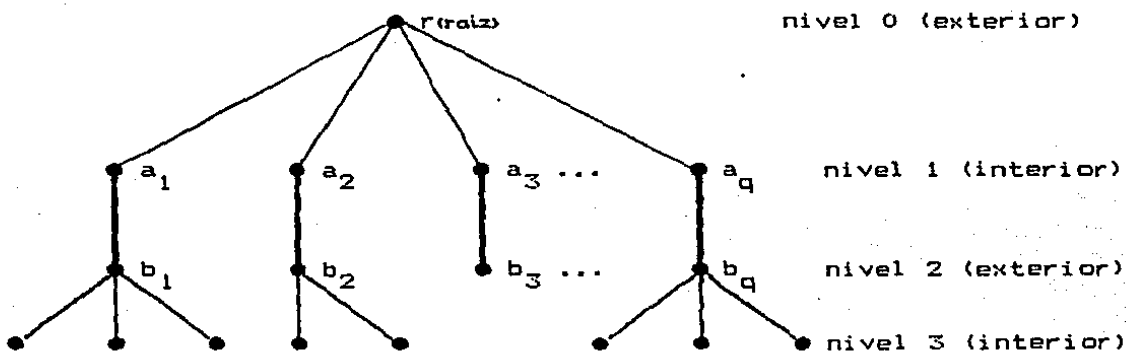


Figura 2.2. Arbol alternante.

En el árbol alternante, se llamarán **nodos interiores** a los nodos situados en los niveles impares y **nodos exteriores** a los nodos

situados en los niveles pares. Conforme el árbol alternante crece, si se encuentra en un nodo interior, entonces se agrega al árbol el correspondiente arco acoplado. Por lo tanto todo nodo interior es de grado 2. Pero en los nodos exteriores x pueden surgir diferentes casos. Si se examina un arco (x,y) incidente en x. Si y es un nodo expuesto y no está contenido en el árbol, se habrá encontrado una trayectoria aumentante. De otra manera, si y es un nodo saturado con nodo compañero z (ninguno de los dos todavía en el árbol), se agragan ambos arcos (x,y) y (y,z) al árbol, haciendo y un nodo interior y z un nodo exterior. Si y está en el árbol como un nodo interior (las dos formas en que esto puede pasar se muestran en la figura 2.3(a)), el ciclo formado agregando el arco

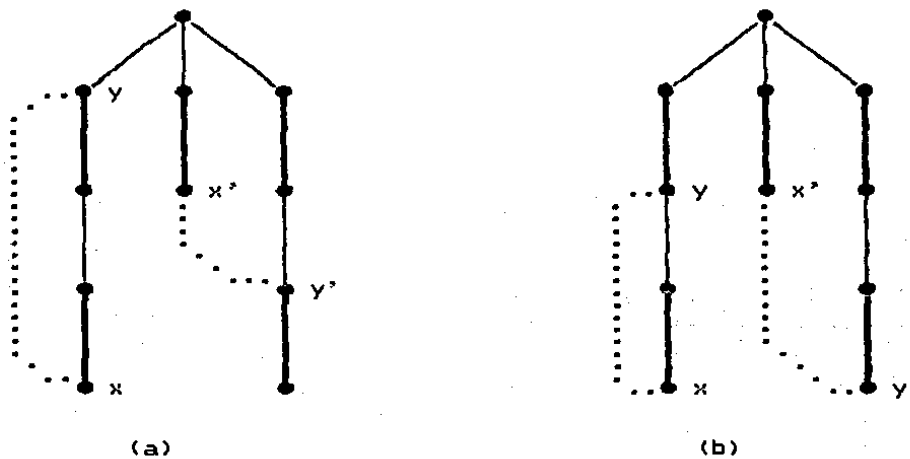


Figura 2.3 Ciclos formados en árboles alternantes:
(a) Ciclos Pares (b) Ciclos Impares.

(x,y) al árbol es un ciclo de longitud par (contiene un número par

de arcos). Puede mostrarse que el arco (x,y) puede ignorarse ya que no crea una trayectoria aumentante adicional. Finalmente, si y está en el árbol como nodo exterior, se habrá formado un ciclo de longitud impar (vea figura 2.3(b)). (Esto no puede ocurrir en las gráficas bipartitas.) A este ciclo alternante de longitud impar se le llama **floración**. La presencia de floraciones complican el algoritmo de acoplamiento para gráficas no bipartitas.

2.2 Manejo de floraciones y árbol alternante

En el algoritmo original de Edmonds, cuando se detecta una floración se contrae a un único nodo (llamado pseudonodo), y se guarda para su futura expansión. El pseudonodo que reemplaza a la floración se registra en el árbol como un nodo exterior y el árbol sigue su crecimiento como ya se explicó antes. Aunque la contracción de un nodo conceptualmente es simple, la implementación en la computadora se vuelve complicada. Edmonds mostró que usando el teorema de Berge, el concepto de árbol alternante y la contracción y expansión de floraciones, un acoplamiento máximo puede obtenerse en un tiempo $O(n^4)$.

Posteriormente, Gabow [1972] refina el algoritmo de Edmonds evitando explícitamente las contracciones y expansiones de floraciones y obteniendo un algoritmo que converge en un tiempo $O(n^3)$. Lawler [1976] también obtiene una version modificada del algoritmo de Edmonds que trabaja en un tiempo $O(n^3)$, en donde las contracciones y expansiones de floraciones no se requieren explícitamente, debido a una eficiente técnica de etiquetación

para recorrerlas.

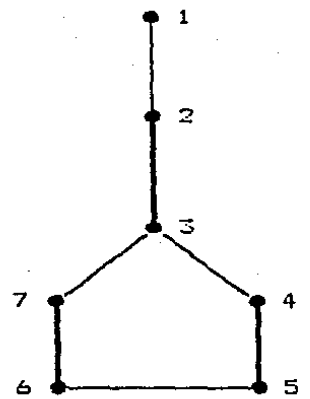


Figura 2.4 Floración.

En este capítulo se empleará una técnica mucho más simple de etiquetación desarrollada por Pape y Conradt [1980]. Ellos desarrollan cada ciclo de longitud impar en dos trayectorias alternantes, que se basa en el hecho que un nodo j debe estar en una distancia par (un nodo exterior) a lo largo de la trayectoria desde la raíz y a una distancia impar (un nodo interior) a lo largo de otra trayectoria desde la raíz si y sólo si j está en un ciclo impar. Por ejemplo en la figura 2.4, el nodo 6 a lo largo de la trayectoria alternante (1,2,3,4,5,6,7) es un nodo interior, mientras que el mismo nodo 6 a lo largo de la trayectoria alternante (1,2,3,7,6,5) es un nodo exterior. Por lo tanto los cinco nodos en el ciclo impar de la figura 2.4 pueden tener etiquetas interior y exterior.

El árbol alternante debe crecer a lo ancho (breadth-first) comenzando en un nodo expuesto r arbitrario. Como ya se mencionó al principio, el crecimiento comienza en un nodo exterior. (En todos los nodos interiores agregamos los correspondientes arcos acoplados.) Para llevar a cabo la búsqueda a lo ancho definimos una **lista Q** que contiene todos los nodos exteriores en el árbol que no han sido explorados. Se toma un nodo x del inicio de la lista, se elimina de la lista y se explora, examinando todos los nodos adyacentes a x (vea figura 2.2).

Para registrar los nodos del árbol que están como raíz o como nodos interiores, se guardan en un arreglo booleano denominado **noarbol** de tamaño n (n es el número de nodos en la gráfica). Al inicio hacemos cada una de sus componentes verdadera excepto la raíz r. Esto es:

```
noarbol(r) ← falso;
para todo j ∈ N, j ≠ r haz noarbol ← verdadero
```

Cada vez que un nuevo nodo j en el árbol se hace interior se registra como noarbol(j) ← falso. Recuerde que siempre que entra al árbol un nodo interior, su nodo adyacente acoplado se hace automáticamente nodo exterior.

Otro arreglo, llamado **padre**, de tamaño n se utiliza para obtener la trayectoria hacia atrás hasta la raíz cuando se descubre una trayectoria aumentante y el acoplamiento se agranda. Para un nodo exterior w, padre(w) = u si hay una trayectoria (u,j,w) en el

árbol alternante de u a w, y (j,w) es un arco acoplado.

2.3 Algoritmo de acoplamiento máximo

Comience con un acoplamiento inicial obtenido de la forma descrita anteriormente y contenido en el arreglo acopla; también se tendrá en la variable expo el número de nodos expuestos. Se continua en la búsqueda de un acoplamiento máximo sólamente si expo ≥ 2. En este caso se toma el primer nodo expuesto r encontrado y se comienza ha hacer crecer el árbol alternante desde r por el método de búsqueda a lo ancho.

Cuando se trate de expander desde un nodo exterior x (tomado del inicio de la lista Q) y se mire su nodo adyacente y, entonces existen varios casos. Si noarbol(y)=falso, ignoramos el arco (x,y) (y procedemos con el siguiente nodo adyacente a x). Esto se debe a que si noarbol(y)=falso implica que y está como nodo interior en el árbol y el arco (x,y) está en el acoplamiento actual o forma un ciclo par con el árbol. En ambos casos se puede ignorar (x,y). Así, se necesita considerar sólamente aquellos nodos adyacentes a x cuyo noarbol tenga valores verdaderos.

Ahora, para el caso noarbol(y)=verdadero, el nodo y puede ser acoplado o puede ser expuesto. Si es expuesto (es decir, si acopla(y)=0), se ha encontrado una trayectoria aumentante. Y se procede ha hacer el acoplamiento con cardinalidad mayor, trazando la trayectoria desde este nodo a la raíz e intercambiar los arcos que están en el acoplamiento con arcos que no lo están. Enseguida

se abandona el árbol alternante actual pero se continua con el ciclo principal haciendo crecer un árbol alternante desde el siguiente nodo expuesto.

Si y no es un nodo expuesto, primero se debe estar seguro que y no sea un ancestro de x en el árbol; de otra manera, puede encontrarse un trayectoria aumentante falsa. Esto se logra checando desde la trayectoria de x a la raíz usando el arreglo abuelo. En el caso que y sea ancestro de x, no se hace nada. Si y no es ancestro de x, y y está acoplado con el nodo z, se deben incorporar al árbol (es decir, insertar en la lista al nodo exterior z, y hacer abuelo(z) ← x; noarbol(y) ← falso).

Se termina el ciclo de construcción del árbol cuando se encuentra una trayectoria aumentante (es decir, la variable booleana **encontrar** = verdadera) o cuando no hay más nodos exteriores que expander (es decir, cuando la lista está vacía).

El algoritmo de acoplamiento finaliza (es decir, el acoplamiento obtenido es máximo) cuando el número de nodos expuestos es menor que 2 o cuando todo nodo expuesto existente ha pasado a formar parte de la raíz de algún árbol alternante sin haber obtenido una trayectoria aumentante.

El algoritmo de acoplamiento máximo puede describirse más formalmente y de manera compacta como sigue:

Algoritmo 2.1: Acoplamiento máximo

```
comienza
   empieza con un acoplamiento inicial;
   para todo r ∈ N haz
      si (acopla(r) = 0) y (expo ≥ 2) entonces
      comienza {* crece árbol alternante enraizado en r *}
         para toda v ∈ N haz noarbol(v) ← verdadero;
         noarbol(r) ← falso;
         inicializa Q conteniendo unicamente a r;
         encontrar ← falso; {* tray. aumentante no encontrada *}
         repite {* hasta que Q=ϕ o encontrar = verdadero *}
            borra nodo x de Q; {* x encabeza la lista en Q *}
            mientras no encontrar haz
            comienza
               para todo y adyacente a x haz
                  si noarbol(y)=verdadero entonces
                  comienza
                     agrandar acoplamiento;
                     expo ← expo-2;
                     encontrar ← verdadero
                  fin
                  sino  si acopla(y) ≠ x entonces
                  comienza
                     z ← acopla(y);
                     si (x ≠ r) or (z no es ancestro de x) entonces
                     comienza {* agrega arcos (x,y) y (y,z) al árbol
                        noarbol(y) ← falso;
                        abuelo(z) ← x;
                        insertar z en la cola de Q
                     fin
                  fin
            fin {* mientras *}
         hasta encontrar o (Q = ϕ)
   fin {* para *}
fin
```

Aunque se puede iniciar el algoritmo con cualquier acoplamiento inicial, incluyendo el vacío, es más eficiente comenzar con un acoplamiento tan grande como sea posible. Un acoplamiento inicial, "razonablemente grande", puede obtenerse rápidamente considerando cada nodo expuesto j en la gráfica y acoplándolo con el primer nodo expuesto adyacente a este nodo.

Sea la variable expo denote el número de nodos expuestos en la gráfica y el arreglo acopla de tamaño n denote el acoplamiento en la gráfica. Esto es, si (i,j) es un arco en el acoplamiento, entonces acopla(i)=j y acopla(j)=i. Para un nodo expuesto k, acopla(k)=0. El siguiente algoritmo da este acoplamiento inicial.

Algoritmo 2.2: Acoplamiento inicial

```
comienza
   para toda v ∈ N haz acopla(v) ← 0;
   expo ← n; (* numero de nodos expuestos *)
   para toda u ∈ N haz
   comienza
      si acopla(u) = 0 entonces
      comienza
         v ← un nodo expuesto adyacente a u;
         acopla(u) ← v;
         acopla(v) ← u;
         expo ← expo-2
      fin
   fin
fin
```

El acoplamiento se agranda cuando se ha encontrado un nodo expuesto y adyacente a el nodo actual x. El algoritmo para agrandar el acoplamiento a través de la trayectoria que une a y con r intercambiando los arcos acoplados en no acoplados y viceversa, es el siguiente:

Algoritmo 2.3: Agrandamiento del acoplamiento

```
acopla(y) ← x;
repite
   siguiente ← acopla(x) {* siguiente es el antiguo compañero de x *}
   acopla(x) ← y {* y es el nuevo campañero de x *}
   si siguiente ≠ 0 entonces {* si x no es la raiz *}
   comienza
      x ← abuelo(x);
      acopla(siguiente) ← x;
      y ← siguiente
   fin
hasta siguiente = 0
```

Finalmente el algoritmo para determinar si el nodo z (que está acoplado con y y es nieto del nodo actual x) es un ancestro de x en el árbol alternante construido es:

Algoritmo 2.4: Chequeo para la formación de ciclos impares

```
si x ≠ r entonces {* la raiz no tiene ancestro *}
comienza
   u ← abuelo(x);
   mientras (u ≠ r) y (u ≠ y) haz
      u ← abuelo(u); {* recorriendo árbol hacia atrás *}
   si u = r entonces z no es un ancestro de x
fin
```

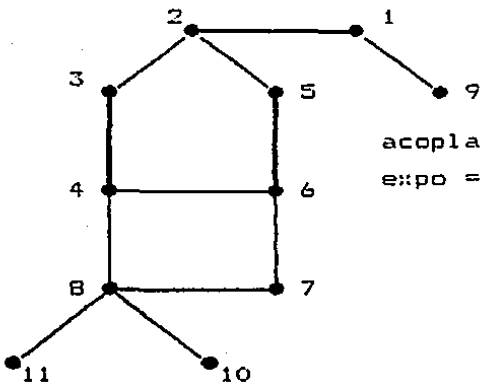
Observe que los arcos (x,y) y (y,z) se incorporan al árbol en el algoritmo 2.1 solamente cuando z no es un ancestro de x (es decir, cuando no se ha formado un ciclo impar).

2.4 Ejemplo

Considere la gráfica G con 11 nodos y 12 arcos dada en la figura 2.5. El acoplamiento inicial producido por el algoritmo 2.2 se

muestra con lineas fuertes.

Empezaremos la aplicación del algoritmo de acoplamiento máximo con el primer nodo expuesto encontrado que es el 9 y por lo tanto se



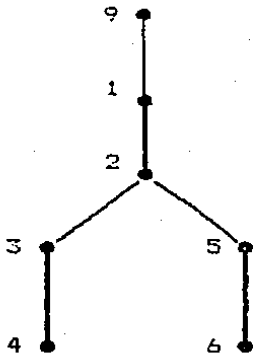
	1	2	3	4	5	6	7	8	9	10	11
acopla	2	1	4	3	6	5	8	7	0	0	0

expo = 2

Figura 2.5. Acoplamiento inicial en la gráfica

comienza el árbol alternante con el nodo 9 como raíz, conforme el algoritmo 2.1. Los arcos (9,1) y (1,2) se agregan al árbol. El siguiente crecimiento comienza en el nodo 2. Después de completar el ciclo para (para todos los nodos adyacentes a 2), el árbol alternante, los dos arreglos y Q se muestran en la figura 2.6.

Después el nodo 4 se quita de la cabeza de Q y sus nodos adyacentes (3,6,8) se examinan en este orden. El nodo 3 se ignora porque noarbol(3)=falso. Para el nodo 6, ya que acopla(6) = 5 ≠ raíz, un chequo para la ancestralidad de 4, por el algoritmo 2.4,



	1	2	3	4	5	6	7	8	9	10	11
noarbol	F	V	F	V	F	V	V	V	F	V	V
abuelo		9		2		2					

inicio			fin
9,	2,	4,	6

Q

Figura 2.6. Arbol alternante

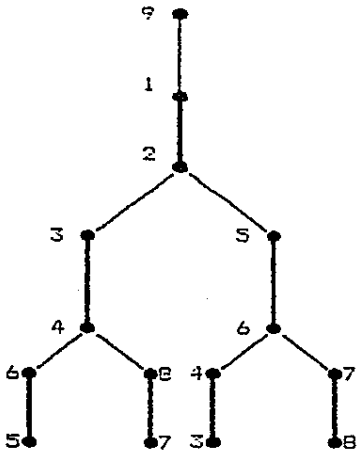
revela que 5 no es un ancestro de 4 en el árbol construido hasta aquí. Por lo tanto, los arcos (4,6) y (6,5) se agregan al árbol; de manera similar, se agregan los arcos (4,8) y (8,7). En seguida, se explora el nodo 6 que está a la cabeza de Q, y el árbol crece a través de los arcos (6,4) y (6,7). La situación en este momento se encuentra sintetizada en la Figura 2.7.

Ahora se explora el nodo 5 y unicamente se necesita considerar el arco (5,2). Puesto que acopla(2) = 1, y 5 raíz, se checa si 1 es un ancestro de 5, con el algoritmo 2.4, como sigue:

```
u ← abuelo(5) = 4
puesto que u        r y u        2 hacemos
   u ← abuelo(u) = abuelo(4) = 2.
Ahora u = 2 = v.
Como u        r, 2 es un ancestro de 5.
Por lo tanto, no se hace nada.
```

Enseguida se toma el nodo 7 y se verifica que para ambos nodos adyacentes 6 y 8, noarbol(6) = noarbol(8) = falso. Así, no se hace

nada en el nodo 7. El nodo siguiente en Q es el 3. Los nodos



	1	2	3	4	5	6	7	8	9	10	11
noarbol	F	V	F	F	F	F	F	F	F	V	V
abuelo		9	6	2	4	2	4	6			

inicio fin

9, 2, 4, 6, 5, 7, 3, 8

Q

Figura 2.7. Arbol Alternante

adyacentes a 3 son el 2 y 4. Como noarbol(2) = verdadero, se examina el arco (3,2). Ya que acopla(2) = 1 raíz, se debe checar si 1 es un ancestro de 3 y en efecto este es el caso. Por lo tanto no se hace nada en 3.

Finalmente se explora el nodo 8, el único nodo actualmente en Q. De los tres nodos adyacentes a 8, se ve que noarbol(4) = noarbol(7) = falso. Pero noarbol(10) = verdadero y acopla(10) = 0. Por lo tanto, se ha encontrado una trayectoria alternante y se debe aumentar el acoplamiento actual por medio del algoritmo 2.3.

Después de aumentar la cardinalidad del acoplamiento en uno, se hace expo ← expo - 2 = 3 - 2 = 1, y encontrar ← verdadera Como

encontrar = verdadera, se sale del ciclo repite...hasta en el algoritmo 2.1, abandonando el árbol alternante actual (y el conjunto Q así como los arreglos noarbol, abuelo y acopla). Se debe continuar en el ciclo exterior, comenzando desde un nodo expuesto, pero como expo = 1, el proceso para.

2.5 Análisis de la complejidad

El acoplamiento inicial obtenido por el algoritmo 2.2, ejecuta dos ciclos para exactamente n veces. En el segundo ciclo, en cada iteración, todos los nodos adyacentes se examinan. Por lo tanto, en el algoritmo 2.2 cada arco de la gráfica se examina dos veces y obtiene un acoplamiento inicial en O(m).

Para el algoritmo 2.1, el ciclo externo puede ser ejecutado a lo más n veces, ya que un árbol alternante está enraizado a lo más una vez en cada nodo. (Comúnmente este ciclo deberá ejecutarse unas cuantas veces, ya que generalmente habrá unos cuantos nodos expuestos con respecto al acoplamiento inicial.) Para cada valor de la raíz r, el ciclo **repite...hasta** se ejecutará a lo más n-1 veces, ya que un nodo puede insertarse en la cola a lo más una vez, y cada iteración de este ciclo elimina un nodo de la cola.

Ahora para cada x eliminado de la cola, el ciclo **mientras** se ejecuta una vez para cada vecino de x. Por lo tanto la condición noarbol(y) = verdadero se prueba, en el peor de los casos, nm veces durante la ejecución del algoritmo. Note sin embargo que esta condición puede ser verdadera a lo más una vez por nodo y para cada valor de la raíz r, puesto que solamente un nodo y puede ser encontrado que no este en el árbol alternante, y se agrega al

árbol [haciendo noarbol(y) ← falso]. Así, en cada iteración para hacer crecer el árbol alternante, la condición noarbol(y) = verdadero debe satisfacerse no más de n veces. Unicamente después de que esta condición se satisface se agranda el acoplamiento (algoritmo 2.3) o se agregan al árbol los arcos (x,y), (y,z) (después de checar si no son ancestros con el algoritmo 2.4). Estas dos operaciones requieren a lo más de O(n) de tiempo. Por lo tanto, para obtener un acoplamiento de máxima cardinalidad con este algoritmo se requiere de $O(n^3)$ de tiempo en el peor de los casos.

C A P I T U L O 3

ACOPLAMIENTO PESADO

En este capítulo se generalizan las ideas del capítulo anterior, al caso en que cada arco de la gráfica tiene un peso o costo y se denomina el problema de acoplamiento pesado; dicho problema consiste en determinar un acoplamiento tal que la suma de los pesos de los arcos que lo forman sea máxima. El problema de acoplamiento pesado se plantea en términos de un problema de programación lineal y para proceder a obtener el algoritmo de solución correspondiente, es necesario extender el manejo de las floraciones y del árbol alternante, para aplicar el algoritmo de acoplamiento máximo a una subgráfica, en donde las soluciones propuestas cumplen con las condiciones de holgura complementaria. El algoritmo de solución se basa en un método primal-dual, donde no aparecen explícitamente todas las restricciones del modelo lineal, sino que se presentan o eliminan al mismo tiempo que las floraciones aparecen o se eliminan.

El desarrollo de este capítulo es como sigue: en la sección 3.1 se extienden las definiciones de trayectoria aumentante y los resultados básicos del capitulo anterior; en la sección 3.2 se modifica para el caso de acoplamiento pesado, el problema del crecimiento de árboles alternantes ya tratados en el capítulo 2; en la sección 3.3 se plantea y se desarrolla el algoritmo de solución; en la sección 3.4 se da un ejemplo del empleo del algoritmo; y en la sección 3.5 se discute la complejidad del

algoritmo propuesto.

3.1 Resultados básicos

El algoritmo de acoplamiento pesado es una generalización del algoritmo descrito en el capítulo 2. Una trayectoria aumentante pesada con respecto al acoplamiento M, es una trayectoria alternante en donde la suma de los pesos de los arcos que no están en M es mayor que la suma de los pesos de los arcos que están en M. Además, si el primero o el último arco de la trayectoria no está en M, entonces el arco tiene un nodo expuesto. Claramente, si se intercambian los arcos que no están en M por los que están en M, en la trayectoria alternante, se obtendrá un acoplamiento nuevo de mayor peso. A diferencia de las trayectorias aumentantes definidas en el capítulo 2, una trayectoria aumentante pesada en M puede tener arcos de M al inicio o al final. Esto induce las siguientes definiciones. Una trayectoria aumentante pesada en M que contiene:

(a) más arcos fuera de M que en M se llama trayectoria aumentante fuerte.

(b) el mismo número de arcos fuera de M que en M se llama trayectoria aumentante neutral.

(c) más arcos en M que fuera de M se llama trayectoria aumentante débil.

El teorema 2.1 tiene la siguiente extensión:

Teorema 3.1. Existe una trayectoria aumentante pesada en M sí y sólo sí M no es un acoplamiento de peso máximo.

Prueba: Claramente si existe una trayectoria aumentante pesada P en G=(N,A) entonces se puede intercambiar los papeles de los arcos en P y obtener un acoplamiento de mayor peso. Así M no puede ser un acoplamiento de peso máximo.

En forma inversa, suponga que M no es un acoplamiento de peso máximo. Se demostrará que G contiene una trayectoria aumentante pesada. Sea M' un acoplamiento de peso máximo, de modo que w(M') > w(M). Sea G' la gráfica formada por todos los nodos de G y los arcos que están en M o M' pero no en ambos. En cada nodo de G' a los más incide un arco de M y uno de M'. Así, cada componente de G' es una trayectoria (quizás un ciclo de longitud par) de arcos que se alternan en M y M'. Como w(M') > w(M) entonces debe haber por lo menos una componente de G' en la que la suma de los pesos de los arcos para arcos en M' que exceda la suma de los pesos de los arcos en M. Esta componente es una trayectoria aumentante pesada en M. ■

3.2 Búsqueda de trayectorias aumentantes

El algoritmo que se llamará algoritmo de búsqueda de una trayectoria aumentante, TRAUM(G), es equivalente a la parte de crecimiento del árbol alternante del algoritmo 2.2, sólamente que en este caso si se guardará memoria de las floraciones que se vayan obteniendo. (Vea algoritmo 3.1). T se inicializa externamente en v, y se etiqueta exterior. Hay de hecho, tres posibles salidas del algoritmo y están etiquetadas por M, B y H. Unicamente la salida M indica que se ha encontrado una trayectoria aumentante.

En otras palabras, que en alguna rama de T se ha encontrado un nodo expuesto. Enseguida se describirá cada salida.

Note que TRAUM(G) construye un árbol a no ser que y se encuentre etiquetado exterior, en cuyo caso, se ha encontrado un ciclo de longitud impar y esto causa un brinco a B. Si y es interior, se detecta un ciclo de longitud par. En este caso (x,y) no se agrega a T y el procedimiento busca extender el árbol desde algún otro nodo exterior.

Algoritmo 3.1. Búsquda de trayectorias aumentantes TRAUM(G).

1. Escoja un nodo exterior x en T y algún arco (x,y) no explorado previamente. Considere que (x,y) ha sido explorado. Si no existe tal arco vaya a H.
2. Si y es expuesto y no etiquetado, agrege (x,y) a T. Vaya a M.
3. Si y es exterior agrege (x,y) a T. Vaya a B.
4. Si y es interior vaya a 1.
5. Sea (y,z) el arco en A con punto final y. Agrege (x,y) y (y,z) a T. Etiquete y interior y z exterior. Vaya a 1.

Como ya se mencionó en el capítulo 2, la presencia de ciclos impares introduce ambiguedades en la búsqueda de trayectorias alternantes. Esto sucede puesto que cualquier nodo j, en este tipo de ciclos, puede etiquetarse tanto exterior como interior dependiendo de la dirección que se tomó alrededor del ciclo para llegar a j. Si j se etiqueta interior, entonces T no puede extenderse a partir de j y no se pueden detectar posibles trayectorias aumentantes. De hecho, el algoritmo de acoplamiento

de peso máximo que usa TRAUM(G) deja todas las opciones abiertas recordando cualquier ciclo C de longitud impar. Una gráfica nueva se construye al contraer C para formar un sólo nodo etiquetado exterior. El algoritmo continua con otra llamada al procedimiento TRAUM(G). Todas las etiquetas exteriores e interiores se llevan adelante, excepto las que están en C. Si subsecuentemente en una rama de T se encuentra que hay un nodo expuesto, entonces la trayectoria aumentante encontrada, puede pasar a través de uno o más de estos pseudonodos. De hecho, note que un ciclo de longitud impar puede por sí mismo contener pseudonodos y asi sucesivamente. Cada pseudonodo puede contener en su interior otros pseudonodos. Se regresará a esto cuando se describa la salida a la etiqueta A.

Considere la salida a H. En esta situación T no puede extenderse. Cada trayectoria alternante trazada a partir de la raíz de T se interrumpe en algún nodo exterior. El único nodo expuesto es la raíz de T. A este tipo de árbol se le llama **arbol húngaro**. Algunos de los nodos etiquetados exterior pueden ser pseudonodos, pero cada nodo etiquetado interior es un nodo ordinario. Es importante notar que arcos que conectan nodos que están en T con nodos que no están en T unicamente pueden unirse a nodos interiores de T. De otro modo algún nodo exterior deberá estar conectado a un nodo expuesto o T puede extenderse a partir del nodo. Se puede ver que ningún nodo en el árbol húngaro , puede formar parte de una trayectoria aumentante, ya que T sólamente contiene un nodo expuesto, si algún nodo de T está en una trayectoria aumentante, entonces esta trayectoria debe entrar a T por un arco que no está

contenido en A, que llega a un nodo interior. De ahí en adelante, la trayectoria visita alternadamente nodos exteriores e interiores, entrando a los primeros por arcos en A y a los últimos por arcos que no están en A. Tal trayectoria nunca puede alcanzar la raíz de T ni dejar a T. Por lo tanto saliendo a H, el algoritmo puede remover T de la gráfica y continuar con la búsqueda de la trayectoria aumentante actual. Si eventualmente A se aumenta, entonces T debe restablecerse a G antes de buscar la siguiente trayectoria aumentante.

Finalmente considere la salida a M. T contiene una trayectoria aumentante de la raíz de T a algún otro nodo expuesto. Sin embargo está puede pasar a través de una o más floraciones contraídas. Estas se expanden y un lado de cada ciclo de longitud impar (el lado de longitud par) puede interpolarse dentro de la trayectoria. Se continua en la misma forma hasta que no existan floraciones en la trayectoria aumentante. De hecho cada expansión puede exponer otros pseudonodos que fueron creados anteriormente al que se está expandiendo. Sin embargo, eventualmente no permanecerán floraciones y un aumento de A es posible.

3.3 Algoritmo de solución

Antes de describir el algoritmo de acoplamiento de peso máximo, se planteará el problema en términos de programación lineal. En lo que sigue, sea $x(u,v)=1$ si el arco (u,v) está en M y $x(u,v)=0$ si no está; $w(u,v)$ denote el peso de arco (u,v). Así, el problema de acoplamiento pesado queda:

$$\text{Maximizar} \sum_{(u,v)} w(u,v)\, x(u,v)$$

$$\text{sujeto a:} \sum_{v} x(u,v) \leq 1 \text{ para toda } u \in N$$

$$\sum_{(u,v) \in R_k} x(u,v) \leq r_k \quad \text{para} \quad 1 \leq k \leq z$$

$$x(u,v) \geq 0 \text{ para toda } (u,v) \in A$$

Como cada arco tiene un peso w(u,v) entonces, la función objetivo es simplemente el peso del acoplamiento. El primer conjunto de restricciones establece que no más de un arco en M es incidente a cualquier nodo u. En el segundo conjunto de restricciones R_k denota la subgráfica inducida por cualquier conjunto de $(2r_k+1)$ nodos. Se denotará por z el número de estas subgráficas. Claramente, no hay más de r_k arcos de R_k en M. Como se demostrará, esta formulación particular proveé un conjunto de condiciones de holgura complementaria que pueden satisfacerse asignando valores 0 o 1 a cada x(u,v).

El problema dual se puede expresar como sigue. Las variables duales y_v y z_k están asociadas con las restricciones primales para el nodo v y la subgráfica R_k, respectivamente.

$$\text{Minimizar} \quad \sum_{v} y_v + \sum_{k} r_k z_k$$

$$\text{sujeto a:} \quad y_u + y_v + \sum_{k:\,(u,v) \in R_k} z_k \geq w(u,v) \quad \text{para toda } (u,v) \in A$$

$$y_v \geq 0 \qquad \text{para toda } v \in N$$

$$z_k \geq 0 \qquad \text{para } 1 \leq k \leq z$$

Note que dentro de la restricción para (u,v), la suma es sobre toda k tal que R_k contiene (u,v).

Las condiciones de holgura complementaria para el par primal-dual son las siguientes:

$$x(u,v) > 0 \rightarrow y_u + y_v + \sum_{k:\,(u,v) \in R_k} z_k = w(u,v) \quad \text{para toda } (u,v) \in A$$

$$y_u > 0 \rightarrow \sum_{v} x(u,v) = 1 \quad \text{para toda } u \in N$$

$$z_k > 0 \rightarrow \sum_{(u,v) \in R_k} x(u,v) = r_k \quad \text{para toda } 1 \leq k \leq z$$

Se referirá a estos tres conjuntos de condiciones como condiciones de holgura para el arco, nodo y cardinalidad impar respectivamente.

El algoritmo que se describirá comienza con el acoplamiento nulo (esto es, $x(u,v)=0$ para todo $(u,v) \in A$) y con variables duales:

$$y_s = W = \frac{1}{2} \max_{(u,v)} (w(u,v)), \text{ para toda } s \in N$$

$$z_k = 0, \text{ para } 1 \leq k \leq z$$

Con esta solución inicial, se observa que las restricciones y las condiciones de no negatividad se cumplen para ambos problemas y de hecho, se siguen satisfaciendo a lo largo de la ejecución del algoritmo. Con la excepción de las condiciones de holgura en los nodos, todas las otras condiciones de holgura se satisfacen. Sin embargo, a medida que se ejecuta el algoritmo, estas condiciones se van satisfaciendo una a una. Cuando todas las condiciones de holgura en los nodos se cumplen, el algoritmo termina con un acoplamiento de peso máximo, ya que la solución que da es factible en el primal y en el dual y cumple con las condiciones de holgura complementaria. Note que si algún nodo v no satisface las condiciones de holgura en el nodo, entonces $y_v > 0$ y v es un nodo expuesto (en todas partes $\sum_v x(u,v)$ es cero o uno).

El algoritmo esencialmente consiste en encontrar en cada iteración una trayectoria aumentante (usando el procedimiento TRAUM, descrito descrito en la sección 3.2) en la subgráfica $G' \leq G$ que consiste de los arcos $(u,v) \in A^*$ donde:

$$A^* = \{ (u,v) \mid y_u + y_v + \sum_{k:(u,v) \in R_k} z_k = w(u,v) \}$$

Si se encuentra una trayectoria aumentante, entonces se extiende entre dos nodos expuestos r y s para los que se cumple:

$$y_r = y_s = W > 0$$

Si se intercambian los papeles de los arcos a lo largo de esta trayectoria aumentante fuerte, entonces se hace que r y s satisfagan las condiciones de holgura en sus nodos. Note que se toma convenientemente el término de trayectoria aumentante fuerte, aún cuando no es claro que el peso del acoplamiento sea incrementado. Lo importante es que este "aumento" causa que dos nodos más satisfagan las condiciones de holgura en nodos. Como cada arco (u,v) en la trayectoria pertenece a la subgráfica G', las condiciones de holgura en los arcos también permanecen satisfechas. Suponga que en lugar de encontrar una trayectoria aumentante fuerte en G', la búsqueda finaliza con un árbol húngaro. (un árbol alternante que no tiene trayectorias aumentantes) En este caso se hacen cambios en las variables duales. Estos cambios pueden permitir que uno o varios arcos sean agregados a A^*, un pseudonodo se expanderá o la variable dual de algún nodo exterior se convertirá en cero. En el último caso si el nodo en cuestión es la raíz del árbol de búsqueda, entonces este nodo ahora satisface las condiciones de holgura; si no es raíz, entonces la trayectoria de la raíz al nodo es una trayectoria aumentante neutral y aumentando a lo largo de la trayectoria, la raíz satisface sus condiciones de holgura en el nodo. Note que este aumento expone al nodo v para el cual $y_v = 0$; sin embargo v satisface la condición de holgura en el nodo simplemente porque $y_v = 0$. Si alguno de los primeros dos casos ocurre (esto es, se agregan arcos a A^* o se expande un pseudonodo) entonces la

búsqueda para una trayectoria aumentante puede continuar desde la misma raíz. Eventualmente, esta búsqueda resultará en que la raíz satisfaga las condiciones de holgura.

Algoritmo 3.2: Cambios en las variables duales, CVD

---

1. para todo nodo u más externo etiquetado exterior y todo nodo u contenido en una floración más externa cuyo pseudonodo está etiquetado exterior haga

   $y_u \leftarrow y_u - t$

2. para todo nodo u más externo etiquetado interior y todo nodo u contenido en una floración más externa cuyo pseudonodo está etiquetado interior haga

   $y_u \leftarrow y_u + t$

3. para toda floración más externa R_k cuyo pseudonodo está etiquetado exterior haga

   $z_k \leftarrow z_k + 2t$

4. para toda floración más externa R_k cuyo pseudonodo está etiquetado exterior haga

   $z_k \leftarrow z_k - 2t$

Cambios en las variables duales envuelven una cantidad t como se describe en el procedimiento de cambios de variables duales, CVD, del algoritmo 3.2 En las lineas 1 y 2 los cambios se hacen a y_u para nodos u que no estén contenidos en una floración (nodos más exteriores) o a nodos u contenidos en una floración más externa (pero no contenidos en floraciones anidadas más profundamente).

Algoritmo 3.3: Procedimiento para evaluar t, EV

1. $t_1 \leftarrow \frac{1}{2} \min_{R_k} \{z_k\}$

   donde R_k esta en la floración más externa cuyo pseudonodo esta etiquetado interior.

2. $t_2 \leftarrow \min_{u} \{y_v\}$

   donde u es cualquier nodo más externo etiquetado exterior o cualquier nodo contenido en una floración más externa cuyo pseudonodo esta etiquetado exterior.

3. $t_3 \leftarrow \min_{(u,v)} \{ y_u + y_v - w(u,v) \}$

   donde u es cualquier nodo más externo etiquetado exterior o cualquier floración más externa cuyo pseudonodo está etiquetado exterior y v es un nodo no etiquetado o está contenido en una floración más externa cuyo pseudonodo no esta etiquetado.

4. $t_4 \leftarrow \frac{1}{2} \min_{(u,v)} \{ y_u + y_v - w(u,v) \}$

   donde u y v están en nodos más externos etiquetados exterior o nodos contenidos en diferentes floraciones más externas cuyos pseudonodo esta etiquetado exterior.

5. $t \leftarrow \min\{ t_1, t_2, t_3, t_4 \}$

Se define t como el máximo tal que las variables duales continuen dando una solución factible al problema dual. Las restricciones duales y las condiciones de no negatividad continuan siendo válidas si t se obtiene mediante la evaluación de t en el procedimiento, EV, del algoritmo 3.3. Note que en la proposición 4 de este diagrama, u y v no están contenidos en la misma floración

más externa. Si así fuera, entonces cualquier cambio a las variables duales no afectaría la restricción:

$$y_u + y_v + \sum z_k \geq w(u,v)$$

puesto que el cambio de t en cada y_u y y_v esta compensado por el cambio de 2t en z_k para la floración más externa que contenga a u y v.

A continuación se considera el efecto de los cambios en las variables duales:

(a) Si $t = t_1$ entonces alguna variable dual z_k se hace cero. Se expande el pseudonodo asociado a su ciclo original de longitud impar. El pseudonodo fue etiquetado interior y debe haber tenido el punto final en algún arco de M. Este arco por lo tanto, acopla algún arco en el ciclo de longitud impar asociado. Los otros $2r_k$ nodos del ciclo pueden entonces acoplarse agregándose arcos del ciclo a M. También cuando el pseudonodo se expande, se puede retener las etiquetas existentes en los nodos que definen a T y (si el pseudonodo tiene grado 2 en T) se puede agregar a T la única trayectoria alrededor de un lado de la floración que deja a T conectada y alternante. En este caso etiquetamos los nodos de la trayectoria exterior e interior de la manera apropiada.

(b) Si $t = t_2$, entonces alguna variable dual y_u se convierte en cero. La trayectoria en el árbol de búsqueda de la raiz a u (si u no es la raiz) es una trayectoria alternante con el mismo número

de arcos en M y fuera de M. Si se intercambian los papeles de los arcos a lo largo de esta trayectoria entonces la raíz se convierte en acoplada (satisfaciendo las condiciones de holgura en los nodos) y puesto que $y_u=0$, u también continua satisfaciendo la condición de holgura en el nodo. Si la variable y para la raíz es cero, entonces cumple con la condicion de holgura en su nodo. En ambos casos se puede continuar construyendo un nuevo árbol de búsqueda desde otro nodo expuesto v para el cual $y_v > 0$. Si no existe este nodo, el algoritmo para.

(c) Si $t = t_3$, entonces el arco asociado puede agregarse a A^* y la búsqueda para una trayectoria aumentante puede extenderse.

(d) Si $t = t_4$, entonces el arco asociado puede agregarse a A^*. Cuando continua la búsqueda de una trayectoria aumentante, esta resultará en el descubrimiento de un ciclo impar.

Con la explicación anterior, se puede presentar el procedimiento del árbol húngaro dado en el algoritmo 3.4. Cuando el procedimiento TRAUM sale a H, la etiqueta H es una llamada a ARHUN. La salida de ARHUN regresa a TRAUM, etiquetado por M o a la proposición etiquetada C. Esto precede a TRAUM y escoje la raíz de un árbol alternante nuevo. Note que ARHUN incluye los procedimientos EV y CVD en las lineas 1 y 2. La linea 3 expande pseudonodos como se describe en (a) arriba. La proposición condicional de las lineas 4-6 trata con trayectorias aumentantes (neutrales) como se describe en (b). La linea 7 agrega arcos

apropiados a A^k de acuerdo a lo preescrito en (c) y (d). Si $t = t_2$ entonces la raíz de T se hace que satisfaga la condición de holgura en el nodo de modo que una nueva raíz deberá escogerse para un nuevo árbol T. Esto se logra a través de las lineas 8-10. En los otros casos, $t=t_1$, t_3 o t_4, la búsqueda en el árbol actual puede continuarse en la llamada a TRAUM en la linea 11.

Algoritmo 3.4: El árbol húngaro, ARHUN.

---

1. EV
2. CVD
3. **Si** $t=t_1$ **entonces** expande cada pseudonodo más externo etiquetado interior y que tenga su variable z igual a cero.
4. **Si** $t=t_2$ y (la variable y de la raíz de T $\neq \phi$) **entonces comienza**
5. Identifique la trayectoria alternante P de la raíz a algún nodo cuya variable y sea cero.
6. Intercambie los papeles de los arcos a lo largo de P
   fin
7. **Si** $t=t_3$ o $t=t_4$ **entonces** aumente A^*
8. **Si** $t=t_2$ **entonces comienza**
9. remueva todas las etiquetas interiores y exteriores
10. **vaya a** C
    fin
11. **Si** $t=t_1$ o $t=t_3$ o $t=t_4$ **entonces vaya a** M.

Ahora se está en posición de presentar el algoritmo de acoplamiento de peso máximo que está esbozado en el algoritmo 3.5. La linea 1 inicializa M en el acoplamiento vacío. En la linea 2 las variables duales y_v se inicializan, mientras que la variable z

de cada floración se inicializa en la linea 5 cuando se descubre la floración. El paso general del algoritmo se inicializa en la linea 3 con la identificación de un nodo que no satisfaga la condición de holgura en nodo. Dentro de la linea 4 el procedimiento de búsqueda de una trayectoria aumentante crea un árbol alternante enraizado en este nodo y únicamente usa los arcos de A^* que define G^*. Como se describió en un principio, la salida del procedimiento TRAUM a B si se encuentra una floración, a H si

Algoritmo 3.5: Acoplamiento de peso máximo

1. $M \leftarrow \phi$
2. Para toda v en N haz
   $$y_v \leftarrow \frac{1}{2} \max \{ w(v_i,v_j) \mid (v_i,v_j) \in A \}$$

C: 3. Escoja un nodo v tal que v es expuesto y $y_v > 0$. Si no existe tal nodo vaya a L. Etiquete v exterior.

M: 4. TRAUM(G^*)

B: 5. Identifique la floración y sáltela. Etiquete el pseudonodo resultante exterior y asigne cero a su variable z.
vaya a M.

H: 6. ARHUN

A: 7. Identifique la trayectoria aumentante (fuerte). Aumente M intercambiando los papeles de los arcos. Quite todas las etiquetas interiores y exteriores.
vaya a C.

L: 8. Expanda todos los pseudonodos restantes en la gráfica final haciéndolo en el orden inverso al que fueron encontrados, induciendo un acoplamiento máximo en cada floración expandida.

se descubre un árbol húngaro y a A si se descubre una trayectoria aumentante (fuerte). Las proposiciones de las lineas 5, 6 y 7

resultan en un regreso a la linea 3 si la condición de holgura complementaria del nodo raíz en el árbol se satisface o en un regreso a la linea 4 cuando el árbol puede seguir desarrollándose. Eventualmente todos los nodos satisfacerán sus condiciones de holgura complementaria y el algoritmo termina en la linea 9.

Como se indicó, el crecimiento de un árbol alternante desde alguna raíz v eventualmente resultará en que las condiciones de holgura complementaria en el nodo se satisfagan. Como se ha visto, el algoritmo mantiene factibilidad en la solución primal y en la solución dual. Para la verificación completa del algoritmo, por lo tanto únicamente se debe mostrar que al terminar, las condiciones de holgura complementaria en los arcos, y en los subconjuntos de cardinalidad impar se cumplen. Note que si un arco (u,v) esta en M y no está en un pseudonodo entonces (u,v) esté en A^*. También si (u,v) esta contenido dentro de algún pseudonodo entonces el valor de $(y_u+y_v+\sum z_m)$ no cambia por cambios a las variables duales. Así, las condiciones de holgura en los arcos se satisfacen. Considere las variables z. Cualquier z_k puede hacerse positiva solamente si está contenida dentro de algún pseudonodo. Siempre que un pseudonodo sea expandido, un acoplamiento máximo se induce en los arcos del ciclo impar, de modo que al terminar el algoritmo (cuando ya no existen pseudonodos) las condiciones de holgura en los subconjuntos de cardinalidad impar se satisfacen. Es fácil ver que el algoritmo de acoplamiento de peso máximo es un algoritmo de tiempo polinomial.

3.4 Ejemlo

Con el fin de mostrar como trabaja el algoritmo 3.5, considere la gráfica de la figura 3.1 en donde los números sobre los arcos son los pesos asignados

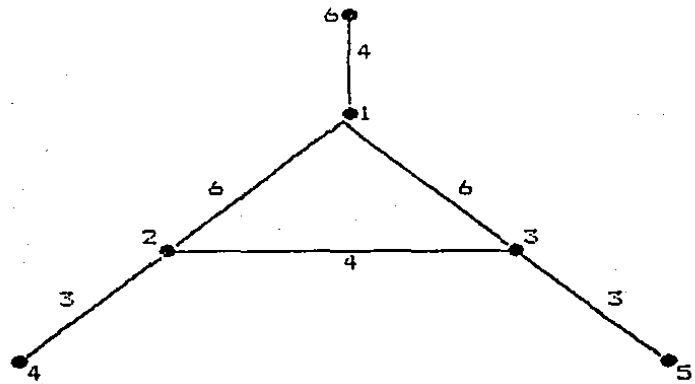


Figura 3.1

En la figura 3.2 se han puesto los cambios en la variables duales y en la variable z_B. Los nodos 1, 3, 4, 5 y 6, se toman respectivamente como raices de los nuevos árboles alternantes en cada iteración del algoritmo 3.5.

Se inicia con un acoplamiento $M = \phi$ y el conjunto,

$$A^* = \{ (u,v) \mid y_u + y_v + \sum_{k:(u,v)\in R_k} z_k = w(u,v) \} = \{(1,2),(1,3)\}$$

	y_1	y_2	y_3	y_4	y_5	y_6	z_B
	3	3	3	3	3	3	
(1)							
(3)	4	2	2	3	3	3	0
	2	0	0	3	3	3	4
(4)							
(5)	4	2	2	1	1	3	0
	3	3	3	0	0	3	
(6)	1	3	3	0	0	3	

Figura 3.2

Se escoje el nodo 1 (1 se etiqueta exterior)

TRAUM(G'): (1,2) se explora. 2 es un nodo expuesto y no etiquetado, de modo que
T ← 1——2
y hay una salida a A

A: (1,2) es una trayectoria aumentante fuerte, de modo que (1,2) se agrega a M y se remueven las etiquetas de G.
Salida a C.

Se escoje el nodo 3 (3 se etiqueta exterior)

TRAUM(G'): (3,1) se explora:
T ← 3——1——2
No se pueden explorar arcos adicionales desde 3 o 2.
Salida a H.

ARHUN: EV: $t_1=\infty$, $t_2=3$ (nodos 2 y 3)
$t_3=3$ (arcos (2,4) y (3,5)), $t_4=1$ (arco (2,3))
Así, $t = t_4 = 1$

CVD: $y_1 \leftarrow 3+1=4$, $y_2 \leftarrow 3-1=2$, $y_3 \leftarrow 3-1=2$
se agrega el arco (2,3) a A^*. Salida a M.

TRAUM(G'): (2,3) se explora; como 3 está etiquetado exterior se encuentra una floración consistente de los arcos (1,2),(2,3),(1,3). Salida a B.

B: Se contrae la floración y el pseudonodo se denota por B y se etiqueta exterior. El arco (1,2) se remueve de M.
$z_B \leftarrow 0$. Salida a M.

TRAUM(G'): No existen arcos explorables desde B. Salida a H

ARHUN:EV: $t_1=\infty$, $t_2=2$ (nodos 2 y 3)
$t_3=2$ (arcos (2,4) y (3,5)), $t_4=\infty$
Así, $t = t_2 = t_3 = 2$

CVD: $y_1 \leftarrow 4-2=2$, $y_2 \leftarrow 2-2=0$, $y_3 \leftarrow 2-2=0$
$z_B \leftarrow 0 + 2(2) = 4$
se agregan (2,4) (3,5) a A^*
Todas las etiquetas se remueven de G
Salida a C.

Se escoje el nodo 4 (4 se etiqueta exterior)

TRAUM(G'): (4,2) se explora. 2 es expuesto y no etiquetado de modo que
T ← 4——B
Salida a A.

A: (4,2) es una trayectoria aumentante, de modo que (4,2) se agrega a M. La etiqueta exterior se remueve de 4
Salida a C.

Se escoje el nodo 5 (5 se etiqueta exterior)

TRAUM(G'): (5,3) se explora de modo que
T ← 5——B——4
Ningún arco se puede explorar desde 5 o 4
Salida a H.

ARHUN: EV $t_1=2$, (nodo B) $t_2=3$ (nodos 4 y 5)
$t_3=\infty$, $t_4=\infty$; $t = t_1 = 2$

CVD: $y_1 \leftarrow 2+2=4$, $y_2 \leftarrow 0+2=2$, $y_3 \leftarrow 0+2=2$
$y_4 \leftarrow 3-2=1$, $y_5 \leftarrow 3-2=1$, $z_B \leftarrow 4 - 2(2) = 0$
Se expande el nodo B. El lado de longitud par de la floración se interpola dentro de T etiquetando apropiadamente sus nodos. (1,3) se agrega a M:
T ← 5——3——1——2——4
Salida a M.

TRAUM(G'): No existe ningún arco explorable desde 5, 1 o 4
Salida a H.

ARHUN:EV: $t_1=\infty$, $t_2=1$ (nodos 4 y 5)

t_3=3 (arco(1,6)) , t_4=∞; $t = t_2 = 1$

CVD: $y_1 \leftarrow 4-1=3$, $y_2 \leftarrow 2+1=3$, $y_3 \leftarrow 2+1=3$
$y_4 \leftarrow 1-1=0$, $y_5 \leftarrow 1-1=0$
Todas las etiquetas se remueven.
Salida a C.

Se escoje el nodo 6 (6 se etiqueta exterior)

TRAUM(G'):T ← 6
Ningún arco se puede explorar desde 6
Salida a H.

ARHUN: EV t_1=∞, t_2=∞, t_3=2, t_4=∞; $t = t_3 = 2$

CVD: $y_1 \leftarrow 3-2=1$
Se agrega arco (1,6) a A^*
Salida a M

TRAUM(G'): T ← 6——1——3——5
5 es un nodo libre y no etiquetado. Salida a A

A: La trayectoria aumentante es (6,1,3,5). Así (1,3) se quita de M mientras que (6,1) y (3,5) se agregan a M.
Se quitan todas las etiquetas.
Salida a C.

No existe ningún nodo expuesto, salida a L. No existen ya pseudonodos de modo que el algoritmo termina con:

$$M = \{(1,6),(2,4),(3,5)\}$$

C A P I T U L O 4

EL PROBLEMA DEL CARTERO CHINO I

El problema de determinación de rutas óptimas puede ser atacado usando diversos enfoques. Uno de estos es en términos de programación entera, sin embargo, los correspondientes métodos de solución resultan inadecuados para casos prácticos. Un enfoque que ha demostrado ser útil es el de teoría de gráficas, dado que sus métodos de solución resultan sencillos y fáciles de implantar en la computadora. En este enfoque conviene analizar el problema en cuestión en términos de gráficas no-dirigidas o dirigidas, dependiendo de la restricción que se tenga en el sentido de recorrer una calle o avenida (en una dirección o en ambas). En este capítulo se resuelve el problema para el caso de arcos no dirigidos (es decir, arcos que se pueden transitar en ambos sentidos).

El desarrollo del capítulo es como sigue: en la sección 4.1 se describe el modelo; en la sección 4.2 se enuncian algunos conceptos y resultados básicos para el estudio del problema para arcos no dirigidos; en la sección 4.3 se propone el algoritmo de solución al problema; en la sección 4.4 se dan algunos ejemplos ilustrativos del algoritmo de solución; finalmente en la sección 4.5 se da un ejemplo de aplicación a un caso de recolección de residuos sólidos.

4.1 Descripción del modelo

El problema que se tiene es escoger una ruta que pase a través de cada calle o camino al menos una vez y se regrese al punto de partida, de tal forma que la distancia total recorrida sea la mínima posible. A este problema se le conoce como el Problema del Cartero Chino.

Este tipo de problema, se puede transformar de manera natural en un problema de optimización en una gráfica; donde las calles o caminos pueden transformarse en arcos, sus uniones y cruces en nodos y la distancia viajada por el cartero al ir recorriendo cada calle en la longitud asociada a este arco. De modo que para la gráfica asociada, el problema es identificar el circuito que atraviesa cada arco al menos una vez y cuya distancia total sea mínima. La longitud total del circuito se evalua sumando las longitudes de los arcos multiplicados por el número de veces que cada arco se usa.

4.2 Conceptos y resultados básicos

Las gráficas más simples que se pueden estudiar para este tipo de problema, son aquellas en las que hay una ruta que atraviesa cada arco exactamente una vez. La longitud total de una ruta de este tipo, es la suma de todas las longitudes de los arcos en la gráfica y es evidente que no existe otra ruta de longitud más corta. Este caso especial no se ve afectado por la longitud de los arcos, ya que podemos alterar las longitudes de los arcos y la ruta seguirá siendo óptima; para este caso especial no es

necesario el uso de las distancias de cada arco. La ruta que satisface estas restricciones se conoce como ruta de Euler o Euleriana. Para este tipo de gráficas, el problema de encontrar una ruta, se reduce a encontrar una ruta de Euler. Un problema más difícil es encontrar una ruta del cartero chino en gráficas donde no existan rutas de Euler o alguno o todos los arcos son dirigidos, este último caso será tratado en el capítulo siguiente. En particular la figura 4.1 muestra una gráfica que tiene varias rutas de Euler, mientras que la figura 4.2 no tiene ninguna.

Por otra parte se debe hacer notar que empezar en cualquier punto de la gráfica no afecta la ruta en sí misma. Suponga que se ha encontrado una ruta que comienza y finaliza en el nodo s y se requiere una ruta que comience y finalice en otro nodo t. Entonces la ruta que comienza en s deberá en algún momento pasar por el nodo t. La primera visita a t puede dividir la ruta en dos trayectorias P1 y P2. Una ruta que se origine en t puede ejecutarse usando la trayectoria P2 y en seguida P1.

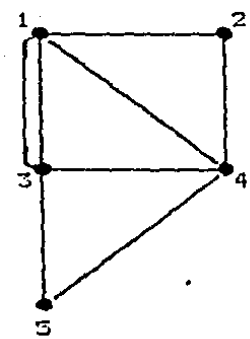


Figura 4.1 Gráfica con varias rutas de Euler.
Por ejemplo: 1-2-4-1-3-5-4-3-1.

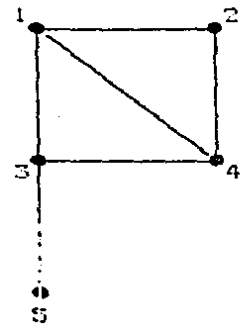


Figura 4.2. Gráfica que no tiene rutas de Euler.

4.3 Algoritmo de solución

El primer paso para determinar una ruta en una gráfica no dirigida es especificar si la gráfica es o no par; una gráfica par es aquella en la que el número de arcos que inciden a todo nodo es par. Si alguno de los nodos tiene un número de arcos incidentes impar, entonces se dice que la gráfica no es par. En una gráfica par se puede encontrar una ruta por la que se transite cada arco solamente una vez. Aquí se describe un algoritmo que determina alguna de estas rutas. En otro tipo de gráfica (gráfica no par) determinaremos una ruta pero algunos arcos deberán circularse más de una vez creándose una gráfica extendida de la original con algunos arcos duplicados, de modo tal que, la gráfica extendida resulta par y la distancia adicional recorrida sea mínima.

La primera parte, es encontrar si la gráfica es o no par; esto simplemente se realiza, contando los arcos que inciden en cada

nodo. Si existe un nodo con un número impar de arcos, entonces no existe ruta de Euler. Los nodos con un número impar de arcos incidentes ocurren por pares ya que cada arco en la gráfica contribuye con dos unidades a la suma de los grados de todos los nodos (una por cada uno de sus nodos terminales), así, la suma de todos los arcos incidentes a todos los nodos es par pero esta suma contiene sumandos pares e impares (ya sea que el nodo sea par o impar respectivamente). Es por ello que en estos sumandos debe haber un número par de sumandos impares para que la suma total sea par.

En una gráfica par, encontrar una ruta de Euler es sencillo. Para tal propósito los arcos se dividen en dos conjuntos, aquellos que han sido usados (los ya transitados en la ruta) y el resto. Una ruta se construye transfiriendo arcos del último conjunto al primero. Inicialmente, todos los arcos están en el segundo. Empezando con el origen de la ruta deseada, cualquier arco no usado incidente a este nodo se selecciona. Este arco se convierte en usado; el proceso se repite, encontrando un arco que no ha sido usado en el otro nodo terminal que une el arco usado y el proceso continua hasta que el origen se alcanza. Si en esta etapa, todos los arcos ya se han usado, entonces la ruta está completa. De otra forma, una o más partes extras deberán agregarse en la ruta; estas partes se encuentran seleccionando un nodo en la ruta que posee un arco incidente no usado. Este nodo se utiliza como el punto inicial para una ruta de arcos no usados y esta "mini ruta" se inserta en la ruta en el punto donde la ruta original visita el

nodo seleccionado. Este proceso continua hasta que todos los arcos han sido usados. Formalmente esto puede escribirse como un algoritmo (ver algoritmo 4.1).

ALGORITMO 4.1. CARTERO CHINO EN UNA GRAFICA NO-DIRIGIDA PAR.

PROPOSITO. Determinar la ruta de longitud mínima en una gráfica no-dirigida donde todos los nodos tienen grado par.

D E S C R I P C I O N

Paso 0: Sea s el origen de la ruta. Etiquete todos los arcos como "no usados". Sea t = s (t representa el último nodo visitado). Sean U y V dos conjuntos vacíos de arcos, que representan la ruta parcialmente completa y la sucesiva "mini-ruta" respectivamente.

Paso 1. Encuentre cualquier arco entre t y q (otro nodo) que no ha sido usado. Uselo y agréguelo a U. Haga t = q.

Paso 2. Si t es igual a s, haga el paso 3; de otra manera regrese al paso 1.

Paso 3. Inserte U en V, en el punto de V donde s se toca por primera vez; U se convierte en vacío. Encuentre un nodo t visitado en V, pero que tiene arcos incidentes no usados. Si no existe este nodo, entonces pare, la ruta está completa; de otra forma, sea s = t y regrese al paso 1.

Cuando se tiene una gráfica que no es par entonces algunos arcos deben transitarse más de una vez. El objetivo es seleccionar aquellos arcos que hagan mínima la distancia total recorrida. La longitud de la ruta es la longitud de los arcos que se repiten, más la longitud total (costo fijo) de todos los arcos en la gráfica. Para encontrar el mejor de estos conjuntos, ponga primero atención en los nodos de orden impar. Puesto que cada visita a un

nodo, requiere el uso de dos arcos, es evidente que al menos uno de los arcos que terminan en un nodo impar deberá usarse dos veces. De modo que la primera parte del método es encontrar todos los nodos de orden impar y entonces encontrar la distancia más corta entre ellos.

Para completar una ruta Euleriana se deberá incluir las trayectorias entre los nodos impares. Estos nodos deberán ser aparejados, esto es, deberá especificarse que pares de nodos impares se conectan por medio de una trayectoria, de manera tal que la distancia total del acoplamiento sea mínima. Si existen únicamente dos nodos impares se tiene un único (y posible) acoplamiento. Sin embargo, si hay cuatro nodos impares, digamos a,b,c,d, existen tres posibles acoplamientos {(a,b),(c,d); (a,c),(b,d); (a,d),(b,c)}; si hay seis nodos impares entonces hay quince posibles acoplamientos; en general, si hay 2m nodos impares hay (2m)!/m!*2m posibles acoplamientos. Una vez que se encuentra el acoplamineto óptimo las trayectorias correspondientes se agregan a la gráfica original, convirtiéndola en una gráfica par y se encuentra, en esta gráfica extendida, una ruta de Euler.

El algoritmo para solucionar el problema del cartero chino en cualquier gráfica no dirigida se describe en el algoritmo 4.2.

ALGORITMO 4.2. CARTERO CHINO EN UNA RED NO-DIRIGIDA.

PROPOSITO. Determinar la ruta de longitud mínima en una gráfica no-dirigida cualesquiera.

D E S C R I P C I O N

Paso 0. Determine si el orden de cada nodo i de la gráfica G = (N,A) es par o impar. Sea $S=\{i_1, i_2, \ldots, i_{2p}\}$ el conjunto de todos los nodos impares. Si S es vacío, vaya al paso 3.

Paso 1. Usando la matriz D de longitudes de los arcos calcule la matriz de 2p*2p de distancias más cortas entre miembros de S, usando una rutina de trayectoria más corta entre dos nodos.

Paso 2. Encuentre las parejas de miembros de S que tienen mínima longitud total, por medio del método de enumeración exhautiva o un algoritmo de acoplamiento. Usando este acoplamiento, encuentre las trayectorias que corresponden a estas distancias más cortas y agregue los arcos de esta trayectoria al conjunto A, formando el conjunto A^*.

Paso 3. Encuentre una ruta de Euler en (N,A^*) que es una gráfica no dirigida par.

4.4 Ejemplos ilustrativos

Ejemplo 4.1

Considere la gráfica que se muestra en la figura 4.3 donde

$$N = \{1,2,3,4,5\};$$

$$A = \{(1,2),(1,3),(2,3),(2,4),(2,5),(3,4),(3,5)\}$$

con matriz de longitudes o costos entre cada par de nodos:

$$D = \begin{bmatrix} 0 & 4 & 2 & \infty & \infty \\ 4 & 0 & 5 & 3 & 6 \\ 2 & 5 & 0 & 3 & 7 \\ \infty & 3 & 3 & 0 & \infty \\ \infty & 6 & 7 & \infty & 0 \end{bmatrix}$$

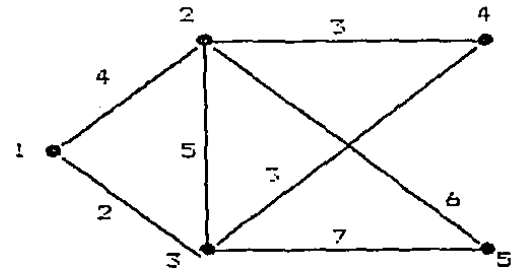


Figura 4.3

Apliquemos el algoritmo dado en la sección (1.2):

Paso 0. nodo 1 tiene orden 2

nodo 2 tiene orden 4

nodo 3 tiene orden 4

nodo 4 tiene orden 2

nodo 5 tiene orden 2

$S = \phi$

$(N,A^*) = (N,A)$

S es vacío, entonces vaya al paso 3.

Paso 3. Use el algoritmo de Euler en (N,A^*).

Paso E0 Sean $S = 1$, $t = s$. Haga $U = V = \phi$. Todos los arcos en A^* no usados.

Paso E1 Seleccione el arco (1,2) y úselo. $U = \{(1,2)\}$; $t = 2$.

Paso E2 Repita paso 1.

Paso E1 Seleccione arco (2,5) y úselo. $U = \{(1,2),(2,5)\}$; $t=5$.

Paso E2 Repita paso 1.

Paso E1 Seleccione arco (5,3) y úselo. U = {(1,2),(2,5),(5,3)}; t = 3.

Paso E2 Repita paso 1.

Paso E1 Seleccione arco (3,1) y úselo. U = {(1,2),(2,5),(5,3),(3,1)}; t = 1.

Paso E2 Como t = s, vaya al paso 3

Paso E3 V = {(1,2),(2,5),(5,3),(3,1)}; U = ϕ. Los nodos 2 y 3 ya han sido visitados en V, pero tienen arcos incidentes no usados. Sea t = s = 2 y vaya al paso 1.

Paso E1 Seleccione arco (2,4) y úselo. U = {(2,4)}. t = 4.

Paso E2 Repita paso 1.

Paso E1 Seleccione arco (4,3) y úselo. U = {(2,4),(4,3)}; t = 3

Paso E2 Repita paso 1.

Paso E1 Seleccione arco (3,2) y úselo. U = {(2,4),(4,2),(3,2)}; t = 2.

Paso E2 Como t = s, vaya al paso 3.

Paso E3 V = {(1,2),(2,4),(4,3),(3,2),(2,5),(5,3),(3,1)}; U=ϕ. Ya no hay nodos con arcos incidentes, no usados; de modo que V representa la ruta del cartero en esta gráfica.

En este ejemplo todos los nodos tienen orden par, de modo que no hay necesidad de usar la matriz de longitudes D.

Ejemplo 4.2

Considere la gráfica que se muestra en la figura 4.4 donde N = {1,2,3,4,5,6,7}; A = {(1,2),(1,3),(1,4),(1,5),(2,5),

(2,7),(3,4),(3,6),(4,5),(5,7),(6,7)}. Con matriz de longitudes:

$$\begin{bmatrix} 0 & 6 & 10 & 9 & 10 & \infty & \infty \\ 6 & 0 & \infty & \infty & 8 & \infty & 16 \\ 10 & \infty & 0 & 12 & \infty & 7 & \infty \\ 9 & \infty & 12 & 0 & 5 & 5 & \infty \\ 10 & 8 & \infty & 5 & 0 & \infty & 7 \\ \infty & \infty & 7 & 5 & \infty & 0 & 13 \\ \infty & 16 & \infty & \infty & 7 & 13 & 0 \end{bmatrix}$$

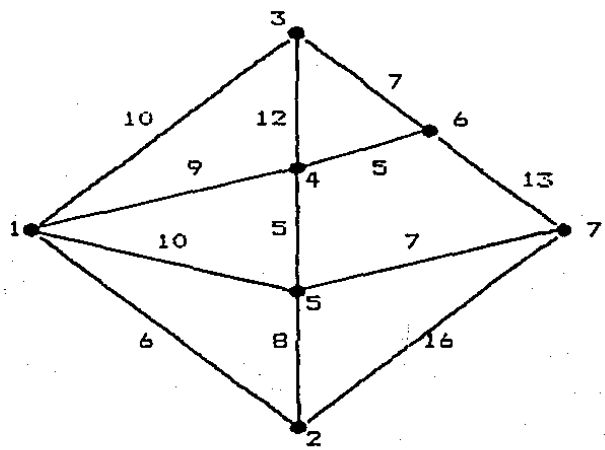


Figura 4.4

Paso 0 Como los nodos 2,3,6 y 7 tienen orden impar, entonces: S = { 2,3,6,7 }; $A^* = A$.

Paso 1 La matriz de distancias mas cortas entre estos nodos es:

$$D^{*}= \begin{bmatrix} 0 & 16 & 18 & 15 \\ 16 & 0 & 7 & 20 \\ 18 & 7 & 0 & 13 \\ 15 & 20 & 13 & 0 \end{bmatrix}$$

Paso 2. Los tres posibles aparejamientos son:

(a) (2,3) y (6,7); longitud = 18 + 13 = 29

(b) (2,6) y (3,7); longitud = 18 + 20 = 38

(c) (2,7) y (3,6); longitud = 15 + 7 = 22

El mejor aparejamiento es (2,7) y (3,6); cuyas trayectorias correspondientes son: (2,5), (5,7) y (3,6), de modo que $A^{*} = A + \{ (2,5),(5,7),(3,6) \}$.

Paso 3. Encuentre una ruta de Euler en (N,A^{*}). Haciendo los pasos como en el ejemplo anterior, encontramos la ruta:

V = {(1,2),(2,5),(5,7),(7,2),(2,5),(5,7),(7,6),(6,3), (3,4),(4,6),(6,3),(3,1),(1,4),(4,5),(5,1)}.

Con una longitud total de 130 unidades.

4.5 Ejemplo de aplicación

Para ilustrar como pasar de la geometría del área bajo estudio a una representación de la misma en la terminología nodos arcos de una gráfica, a continuación se presenta un ejemplo de diseño de rutas de recolección de residuos sólidos municipales. Para este ejemplo se incluye su representación esquemática y algunas observaciones sobre la mejor forma de pasarlos a la terminología nodos arcos de una gráfica no dirigida.

Considere que el área bajo estudio es la que se representa en forma esquemática en la figura 4.5. Suponga que cada una de las calles puede ser transitada en ambos sentidos. La representación espacial del área bajo estudio, puede modificarse a una representación en forma de nodos y arcos asignando a cada segmento de calle un arco no dirigido y en los puntos donde concurren dos o más calles un nodo. Esta transformación se representa en la gráfica de la figura 4.6, donde puede observarse que muchos de los nodos que se encuentran en las fronteras del área bajo estudio tienen orden impar (por ejemplo los nodos 2, 8, 14, 39, 40, etc.) para el caso de la gráfica de la figura 4.6 tenemos 22 nodos de orden impar. Si se alimenta al programa CHINO_I con estos datos de acuerdo a su entrada de datos, descrita en el apéndice, se obtiene la solución que se anexa a continuación.

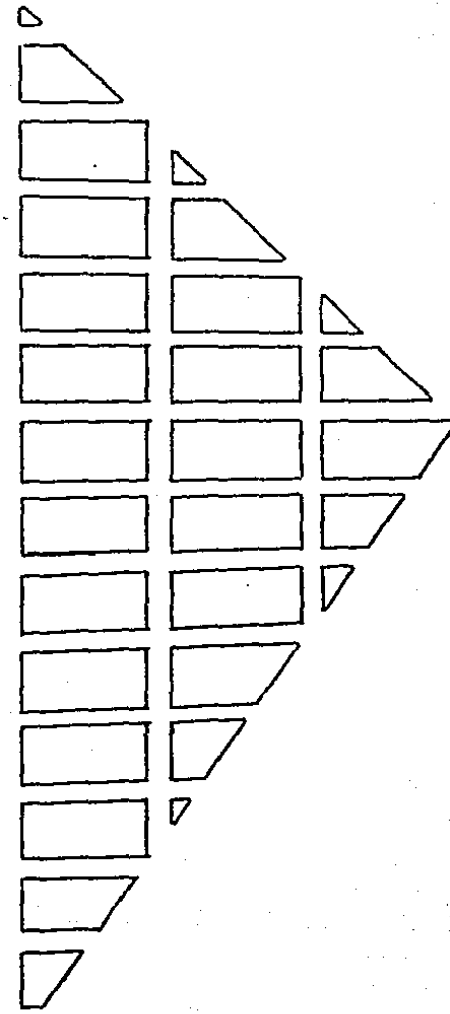


Figura 4.5. Representación esquemática del área bajo estudio del ejemplo 4.5

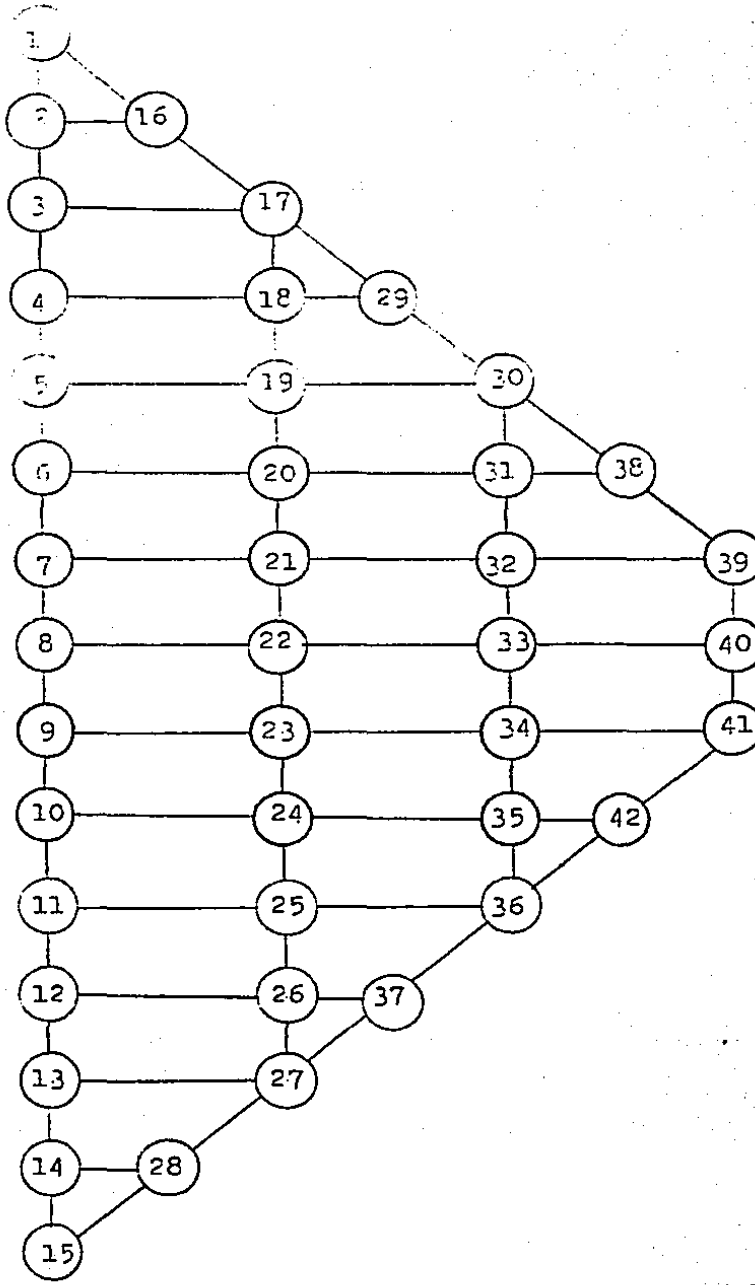


Figura 4.6. Representación de la geometría del ejemplo 4.5 en términos de una gráfica no dirigida

Solución del ejemplo de aplicación 4.5 (vea gráfica de la figura 4.6)

por el programa CHINC_I

Una ruta del cartero chino en la red no dirigida es :

Nodo inicial		Nodo final		Arco usado	
del nodo	1	al nodo	2	por el arco	1
del nodo	2	al nodo	3	por el arco	3
del nodo	3	al nodo	2	por el arco	4
del nodo	2	al nodo	16	por el arco	5
del nodo	16	al nodo	17	por el arco	37
del nodo	17	al nodo	18	por el arco	38
del nodo	18	al nodo	29	por el arco	41
del nodo	29	al nodo	18	por el arco	41
del nodo	18	al nodo	19	por el arco	40
del nodo	19	al nodo	20	por el arco	42
del nodo	20	al nodo	21	por el arco	44
del nodo	21	al nodo	22	por el arco	46
del nodo	22	al nodo	23	por el arco	48
del nodo	23	al nodo	24	por el arco	50
del nodo	24	al nodo	25	por el arco	52
del nodo	25	al nodo	26	por el arco	54
del nodo	26	al nodo	27	por el arco	56
del nodo	27	al nodo	28	por el arco	58
del nodo	28	al nodo	14	por el arco	34
del nodo	14	al nodo	28	por el arco	35
del nodo	28	al nodo	15	por el arco	36
del nodo	15	al nodo	14	por el arco	33
del nodo	14	al nodo	13	por el arco	31
del nodo	13	al nodo	27	por el arco	32
del nodo	27	al nodo	37	por el arco	59
del nodo	37	al nodo	36	por el arco	73
del nodo	36	al nodo	25	por el arco	55
del nodo	25	al nodo	11	por el arco	27
del nodo	11	al nodo	12	por el arco	26
del nodo	12	al nodo	13	por el arco	28
del nodo	13	al nodo	12	por el arco	29
del nodo	12	al nodo	26	por el arco	30
del nodo	26	al nodo	37	por el arco	57
del nodo	37	al nodo	36	por el arco	73
del nodo	36	al nodo	42	por el arco	74
del nodo	42	al nodo	41	por el arco	80
del nodo	41	al nodo	34	por el arco	70
del nodo	34	al nodo	23	por el arco	51
del nodo	23	al nodo	9	por el arco	22
del nodo	9	al nodo	8	por el arco	18
del nodo	8	al nodo	9	por el arco	19

Nodo inicial		Nodo final		Arco usado	
del nodo	9	al nodo	10	por el arco	21
del nodo	10	al nodo	11	por el arco	23
del nodo	11	al nodo	10	por el arco	24
del nodo	10	al nodo	24	por el arco	25
del nodo	24	al nodo	35	por el arco	53
del nodo	35	al nodo	36	por el arco	71
del nodo	36	al nodo	35	por el arco	71
del nodo	35	al nodo	42	por el arco	72
del nodo	42	al nodo	35	por el arco	72
del nodo	35	al nodo	34	por el arco	69
del nodo	34	al nodo	33	por el arco	67
del nodo	33	al nodo	40	por el arco	68
del nodo	40	al nodo	41	por el arco	78
del nodo	41	al nodo	40	por el arco	79
del nodo	40	al nodo	39	por el arco	77
del nodo	39	al nodo	32	por el arco	66
del nodo	32	al nodo	21	por el arco	47
del nodo	21	al nodo	7	por el arco	17
del nodo	7	al nodo	6	por el arco	13
del nodo	6	al nodo	7	por el arco	14
del nodo	7	al nodo	8	por el arco	16
del nodo	8	al nodo	22	por el arco	20
del nodo	22	al nodo	33	por el arco	49
del nodo	33	al nodo	32	por el arco	65
del nodo	32	al nodo	31	por el arco	63
del nodo	31	al nodo	38	por el arco	64
del nodo	38	al nodo	39	por el arco	75
del nodo	39	al nodo	38	por el arco	76
del nodo	38	al nodo	30	por el arco	62
del nodo	30	al nodo	19	por el arco	43
del nodo	19	al nodo	5	por el arco	12
del nodo	5	al nodo	4	por el arco	8
del nodo	4	al nodo	5	por el arco	9
del nodo	5	al nodo	6	por el arco	11
del nodo	6	al nodo	20	por el arco	15
del nodo	20	al nodo	31	por el arco	45
del nodo	31	al nodo	30	por el arco	61
del nodo	30	al nodo	29	por el arco	60
del nodo	29	al nodo	17	por el arco	39
del nodo	17	al nodo	3	por el arco	7
del nodo	3	al nodo	4	por el arco	6
del nodo	4	al nodo	18	por el arco	10
del nodo	18	al nodo	17	por el arco	38
del nodo	17	al nodo	16	por el arco	37
del nodo	16	al nodo	1	por el arco	2

Con una longitud total de 633 unidades

C A P I T U L O 5

EL PROBLEMA DEL CARTERO CHINO II

En este capítulo se considerará el problema del cartero chino en una gráfica dirigida y con pesos en sus arcos. Si la gráfica tiene ciertas características entonces se puede extender el concepto de ruta de Euler; si no, en algunos casos se podrá extender agregando arcos en paralelo de tal suerte que se obtenga, como en el capítulo 4, una gráfica en la que existan rutas de Euler. Existen gráficas dirigidas en las que no tiene solución el problema del cartero chino como se discute en la sección 5.1.

El desarrollo del capítulo es como sigue: en la sección 5.1 se describe el modelo; en la sección 5.2 se enuncian algunos conceptos y resultados básicos para el estudio del problema para arcos no dirigidos; en la sección 5.3 se propone el algoritmo de solución al problema; en la sección 5.4 se da un ejemplo ilustrativo del algoritmo de solución; finalmente en la sección 5.4 se da un ejemplo de aplicación a un caso de recolección de residuos sólidos suponiendo que las calles se recorren en un solo sentido previamente establecido.

5.1 Descripción.

En contraste con la situación descrita en el capitulo anterior, en el caso de arcos dirigidos puede que no exista una ruta, puesto que es posible, que un nodo o varios nodos estén ligados al resto de los nodos de la gráfica en una dirección pero no haya forma de

regresar a ellos, como se muestra en la gráfica de la figura 5.1 en donde los nodos 1 y 2 tienen arcos que comunican con los nodos 3, 4, 5 y 6. Pero una vez estando en los nodos 3, 4, 5 o 6 es imposible regresar a 1 o 2.

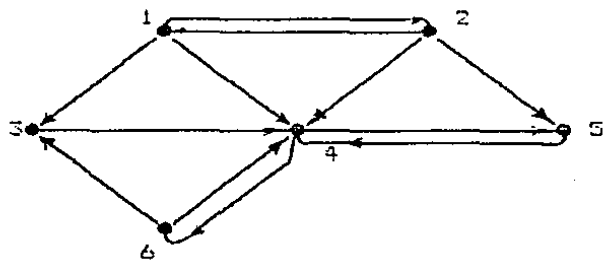


Figura.5.1 No hay ruta del cartero chino puesto que es imposible regresar a los nodos 1 y 2.

5.2 Conceptos y resultados básicos

En una gráfica con arcos dirigidos pueden ocurrir dos situaciones: si el número de arcos que entran a un nodo, es igual al número de arcos que salen del nodo, para todo nodo de la gráfica, entonces existe una ruta de Euler; de otra manera no hay ruta de Euler. En la primera de estas situaciones se dice que la gráfica es **simétrica.**

En una gráfica dirigida simétrica, se puede encontrar una ruta del cartero chino por el método descrito para una ruta de Euler en una gráfica no dirigida, unicamente hay que tener cuidado en escoger el arco que se va a usar, entre aquellos que salen del nodo en el

que actualmente se está.

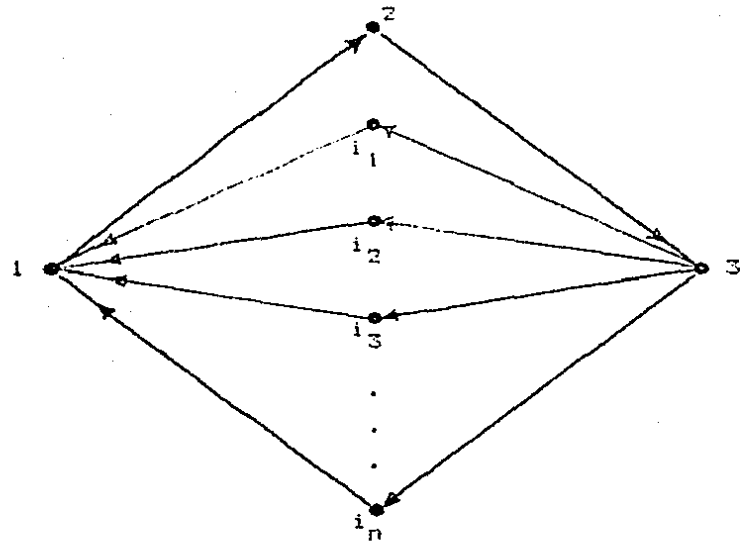


Figura 5.2

Cuando la gráfica no es simétrica, se tendrá la necesidad de repetir algunos arcos para completar la ruta. En contraste al problema de repetir arcos en una gráfica no dirigida, donde el máximo número de veces que un arco se transita es dos, en una gráfica dirigida un arco puede atravesarse cualquier número de veces. Por ejemplo la figura 5.2 muestra una gráfica en la que los arcos (1,2) y (2,3) se usan n veces, y n puede incrementarse sin límite agregando arcos y nodos en la forma que se indica. Para encontrar una ruta del cartero chino, primero es necesario calcular cuántas veces se debe usar cada arco. Una vez que esto se conoce se puede encontrar una ruta de Euler en la gráfica simétrica que resulta de extender la gráfica original con el

número apropiado de copias de cada arco.

5.3 Algoritmo de solución.

Supongamos que la gráfica (N,A) no es simétrica y para cada nodo i contenido en A denotemos por E(i) el conjunto de arcos que entran al nodo i; es decir E(i) = {(j,i) | (j,i) en A } y por S(i) el conjunto de arcos que salen del nodo i, es decir S(i) = {(i,j) | (i,j) en A }. Si E(i) tiene menos elementos que S(i), entonces, alguno de los miembros de E(i) tendrá que repetirse; de otra manera, miembros en S(i) no se podrán usar. Ocurre lo contrario cuando S(i) tiene menos elementos que E(i). Suponga que asociamos a cada arco (j,k) un número X(j,k) + 1 que registra el número de veces que el arco debe ser usado. Claramente X(j,k) ≥ 0. El número de veces que se visita el nodo i está dado por el número de veces que el cartero deja el nodo i o por el número de veces que el cartero entra al nodo i. Estos dos números deberán ser iguales; así que: el número de veces que se entra al nodo i es igual al número de veces que se sale del nodo i o bien;

$$|E(i)| + \sum_{j} X(j,i) = |S(i)| + \sum_{j} X(i,j) \qquad (5.1)$$

Pero también se debe escoger X(i,j) de tal manera que minimice la suma $\sum C(i,j)*X(i,j)$ para (i,j) en A, donde C(i,j) es la longitud del arco (i,j). Este problema se puede ver como un problema de flujo a costo mínimo en donde X representa el flujo y C el costo. Arreglando la ecuación (5.1) nos da la ecuación de balance de

flujo en el nodo i como:

$$\sum_{j} (X(i,j) - X(j,i)) = |E(i)| - |S(i)| = Y(i)$$

Para que el flujo se conserve en el nodo i, en el problema de flujo a costo mínimo, el nodo i debe considerarse como un nodo fuente (Y(i) > 0) o como un nodo sumidero (Y(i) < 0). Para convertir este problema a un problema de flujo factible a costo mínimo, aquellos nodos i, que actuen como nodos fuente deben ligarse a un nodo artificial por arcos cuyo flujo es Y(i) y un conjunto de arcos similar usados para ligar los nodos sumidero. Una vez que el problema de flujo a costo mínimo se resuelve para X(i,j), este número de copias del arco (i,j) deberá agregarse a la gráfica, y se puede encontrar entonces una ruta de Euler en la gráfica simétrica aumentada que resulta. (Ver algoritmo 5.1).

ALGORITMO 5.1. CARTERO CHINO EN UNA GRAFICA DIRIGIDA.

PROPOSITO Determinar la ruta de longitud mínima en una gráfica dirigida cualquiera.

D E S C R I P C I O N

Paso 0. Para cada nodo i de la gráfica G = (N,A), calcule Y(i) = número de arcos que salen de i - el número de arcos que entran a i. Si Y(i) = 0 para todo i en N, vaya al paso 2.

Paso 1. Para cada arco (i,j) en A, haga la cota inferior l(i,j) = 0, la cota superior u(i,j) = ∞ y el costo C(i,j) igual a la longitud del arco. Agregue un nodo r y los siguientes arcos: Para todo nodo i con Y(i)>0, haga un arco (r,i) con l(r,i) = u(r,i) = Y(i) y C(r,i) = 0. Para todo nodo i con Y(i)<0, haga un arco (i,r) con l(i,r) = u(i,r) = |Y(i)| y C(i,r) = 0. Resuelva el problema de flujo a costo mínimo en esta gráfica asociada, con los costos y restricciones dados arriba, con flujos resultantes X(i,j) en el arco (i,j). Agregue X(i,j) copias del arco (i,j) a la gráfica aumentada (N,A^*).

Paso 2. Encuentre una ruta de Euler en la gráfica dirigida $(N;A^*)$. Este paso es idéntico al algoritmo para una ruta de Euler en una gráfica par no dirigida, excepto en el paso 1 de este algoritmo, el siguiente arco (t,q) debe ser un arco dirigido que vaya de t a q.

5.4 Ejemplo ilustrativo

Considere la gráfica dirigida de la figura 5.3 donde:

N = {1,2,3,4,5,6}

A = {(1,2),(2,3),(2,6),(3,6),(4,3),(5,3),(5,4),(6,1),(6,4),(6,5)}

y observe que la matriz de longitudes o costos es:

$$D = \begin{bmatrix} 0 & 10 & \infty & \infty & \infty & \infty \\ \infty & 0 & 14 & \infty & \infty & 12 \\ \infty & \infty & 0 & \infty & \infty & 11 \\ \infty & \infty & 5 & 0 & \infty & \infty \\ \infty & \infty & 6 & 15 & 0 & \infty \\ 8 & \infty & \infty & 13 & 10 & 0 \end{bmatrix}$$

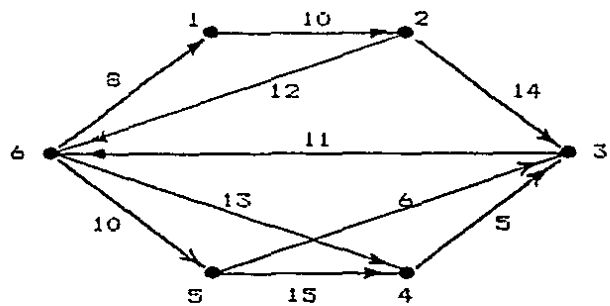


Figura 5.3

Paso 0. $Y(1)=0$, $Y(2)=-1$, $Y(3)=2$, $Y(4)=1$, $Y(5)=-1$, $Y(6)=-1$.
Haga $(N,A^*) = (N,A)$

Paso 1. Resuelva el problema de flujo a costo mínimo para la gráfica que se muestra en la figura 5.4. La solución es: $X(i,j)=0$ excepto para; $X(1,2)=1$; $X(3,6)=3$; $X(4,3)=1$; $X(6,1)=1$; $X(6,5)=1$. A^* se convierte en:
$A + \{(1,2),(3,6),(3,6),(3,6),(4,5),(6,5),(6,1)\}$

Paso 2. Encuentre una ruta de Euler en la gráfica (N,A^*) mostrada en la figura 5.5. Una ruta en esta gráfica es por ejemplo:

(1,2), (2,3), (3,6), (6,5), (5,3), (3,6), (6,5), (5,4), (4,3),
(3,6), (6,4), (4,3), (3,6), (6,1), (1,2), (2,6), (6,1).

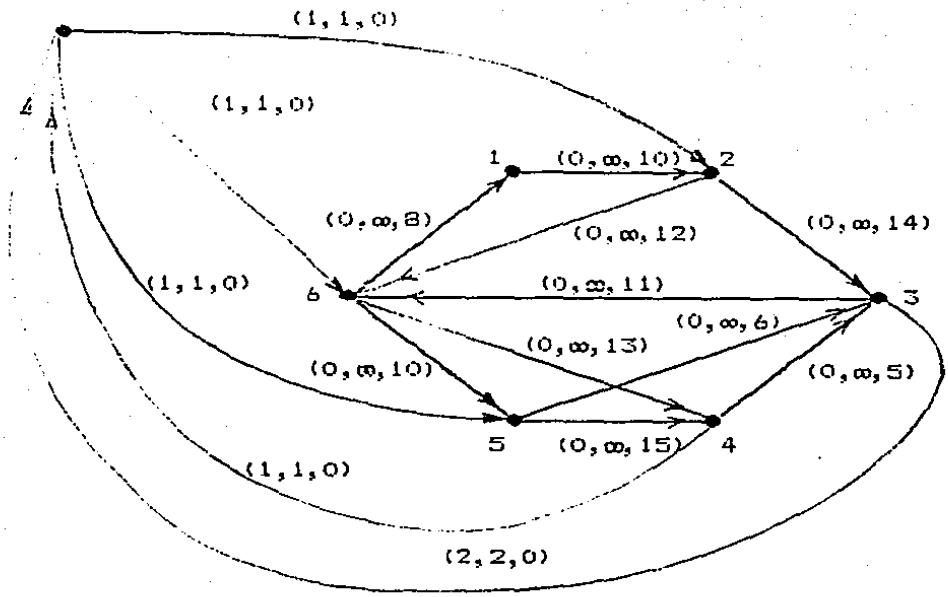


Figura 5.4. Problema de flujo a costo mínimo para la fig. 5.3

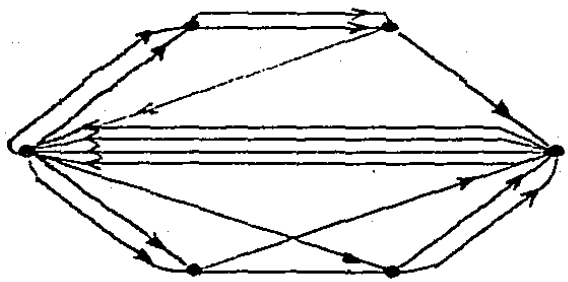


Figura 5.5

5.5 Ejemplo de aplicación

Para ilustrar como pasar de la geometría del área bajo estudio (en donde existan direcciones permisibles al recorrer el área), a una representación de la misma en la terminología nodos y arcos de una gráfica dirigida, a continuación se presenta un ejemplo de diseño de rutas de recolección de residuos sólidos municipales. Para este ejemplo se incluye su representación esquemática y algunas observaciones.

Considere que en el área bajo estudio del ejemplo de aplicación de la sección 4.5, las calles tienen los sentidos dados en la figura 5.6. Así que, cada una de las calles únicamente se puede transitar en el sentido que se indica. La representación espacial del área bajo estudio, puede modificarse a una representación en forma de nodos-arcos en una gráfica dirigida, asignando a cada segmento de calle un arco dirigido en el sentido que se indica, y en los puntos donde concurren dos o más calles un nodo. Esta transformación se representa en la gráfica de la figura 5.7, donde puede observarse que muchos de los nodos no tienen el mismo número de arcos que entran y salen, por ejemplo 2, 11, 16, 42; esto es, la gráfica dirigida no es simétrica y necesariamente una ruta del cartero chino tendrá que repetir algunos arcos más de una vez.

Si se alimenta al programa CHINO_II con estos datos de acuerdo a su entrada de datos, descrita en el apéndice, se obtiene la solución que se anexa a continuación.

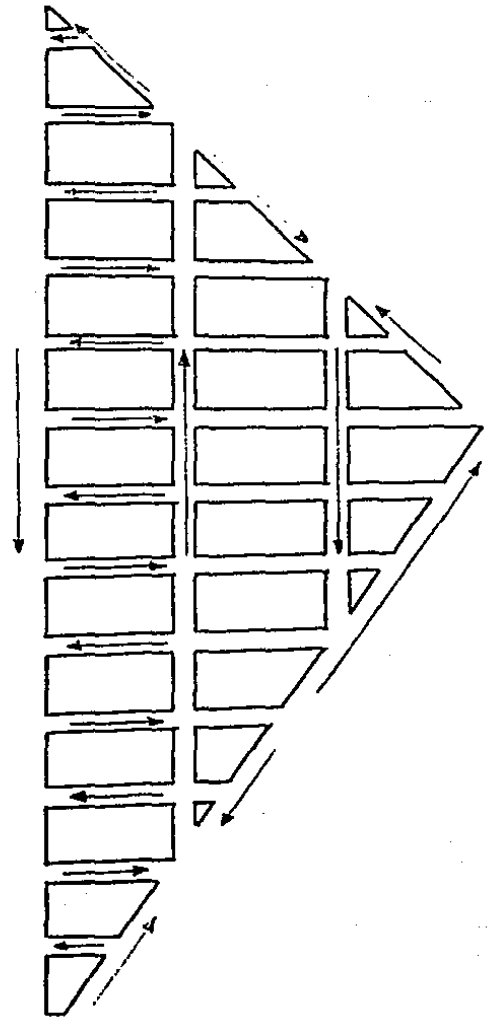


Figura 5.6. Representación esquemática del área bajo estudio del ejemplo 5.5

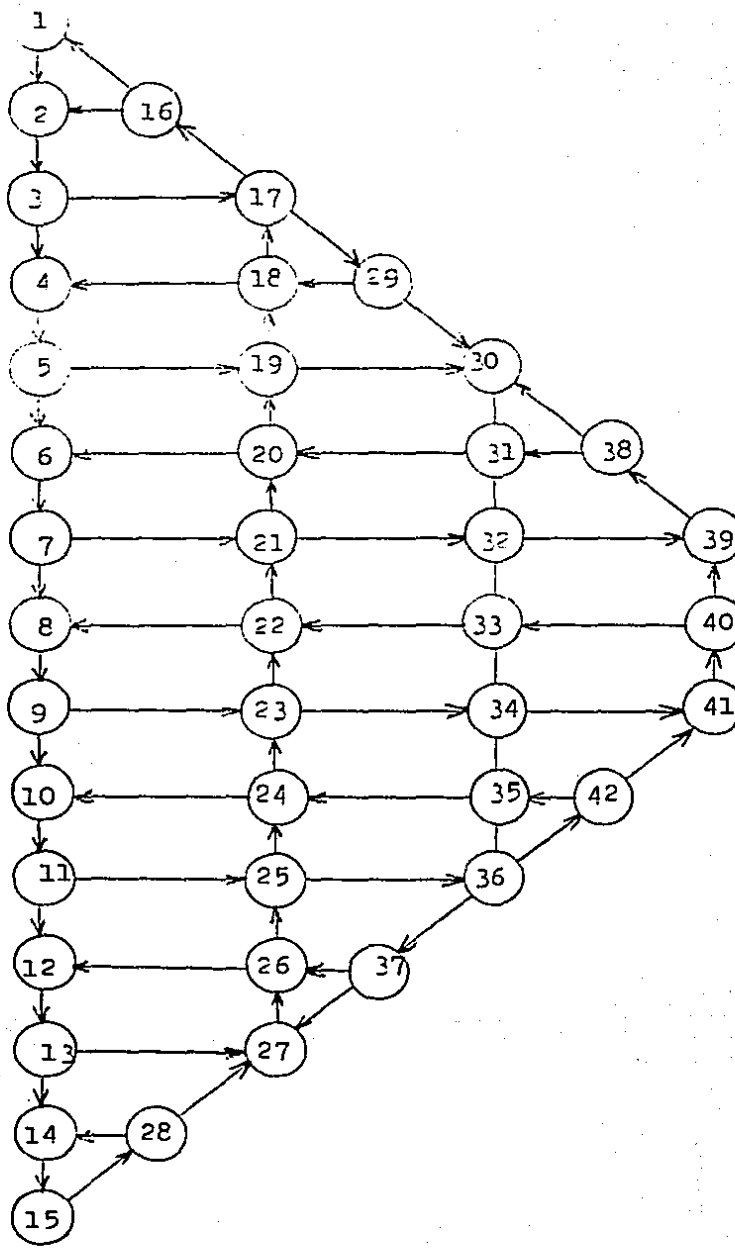


Figura 5.7. Representación de la geometría del ejemplo 5.5 en términos de una gráfica dirigida

Solución del ejemplo de aplicación 5.5 (vea gráfica de la figura 5.7) por el programa CHINC_II

Una ruta del cartero chino en la red dirigida es:

Nodo inicial		Nodo final		Arco usado	
del nodo	1	al nodo	2	por el arco	1
del nodo	2	al nodo	3	por el arco	3
del nodo	3	al nodo	4	por el arco	5
del nodo	4	al nodo	5	por el arco	7
del nodo	5	al nodo	6	por el arco	9
del nodo	6	al nodo	7	por el arco	11
del nodo	7	al nodo	8	por el arco	13
del nodo	8	al nodo	9	por el arco	15
del nodo	9	al nodo	10	por el arco	17
del nodo	10	al nodo	11	por el arco	19
del nodo	11	al nodo	12	por el arco	21
del nodo	12	al nodo	13	por el arco	23
del nodo	13	al nodo	14	por el arco	25
del nodo	14	al nodo	15	por el arco	27
del nodo	15	al nodo	28	por el arco	29
del nodo	28	al nodo	14	por el arco	28
del nodo	14	al nodo	15	por el arco	27
del nodo	15	al nodo	28	por el arco	29
del nodo	28	al nodo	27	por el arco	51
del nodo	27	al nodo	26	por el arco	49
del nodo	26	al nodo	12	por el arco	24
del nodo	12	al nodo	13	por el arco	23
del nodo	13	al nodo	27	por el arco	26
del nodo	27	al nodo	26	por el arco	49
del nodo	26	al nodo	25	por el arco	47
del nodo	25	al nodo	36	por el arco	48
del nodo	36	al nodo	37	por el arco	66
del nodo	37	al nodo	26	por el arco	50
del nodo	26	al nodo	25	por el arco	47
del nodo	25	al nodo	36	por el arco	48
del nodo	36	al nodo	37	por el arco	66
del nodo	37	al nodo	27	por el arco	52
del nodo	27	al nodo	26	por el arco	49
del nodo	26	al nodo	25	por el arco	47
del nodo	25	al nodo	36	por el arco	48
del nodo	36	al nodo	42	por el arco	67
del nodo	42	al nodo	41	por el arco	71
del nodo	41	al nodo	40	por el arco	70

Nodo inicial		Nodo final		Arco usado	
del nodo	40	al nodo	33	por el arco	61
del nodo	33	al nodo	22	por el arco	42
del nodo	22	al nodo	8	por el arco	16
del nodo	8	al nodo	9	por el arco	15
del nodo	9	al nodo	23	por el arco	18
del nodo	23	al nodo	22	por el arco	41
del nodo	22	al nodo	21	por el arco	39
del nodo	21	al nodo	32	por el arco	40
del nodo	32	al nodo	33	por el arco	58
del nodo	33	al nodo	34	por el arco	60
del nodo	34	al nodo	35	por el arco	62
del nodo	35	al nodo	36	por el arco	64
del nodo	36	al nodo	42	por el arco	67
del nodo	42	al nodo	35	por el arco	65
del nodo	35	al nodo	24	por el arco	46
del nodo	24	al nodo	10	por el arco	20
del nodo	10	al nodo	11	por el arco	19
del nodo	11	al nodo	25	por el arco	22
del nodo	25	al nodo	24	por el arco	45
del nodo	24	al nodo	23	por el arco	43
del nodo	23	al nodo	34	por el arco	44
del nodo	34	al nodo	41	por el arco	63
del nodo	41	al nodo	40	por el arco	70
del nodo	40	al nodo	39	por el arco	69
del nodo	39	al nodo	38	por el arco	68
del nodo	38	al nodo	30	por el arco	55
del nodo	30	al nodo	31	por el arco	54
del nodo	31	al nodo	32	por el arco	56
del nodo	32	al nodo	39	por el arco	59
del nodo	39	al nodo	38	por el arco	68
del nodo	38	al nodo	31	por el arco	57
del nodo	31	al nodo	20	por el arco	38
del nodo	20	al nodo	6	por el arco	12
del nodo	6	al nodo	7	por el arco	11
del nodo	7	al nodo	21	por el arco	14
del nodo	21	al nodo	20	por el arco	37
del nodo	20	al nodo	19	por el arco	35
del nodo	19	al nodo	30	por el arco	36
del nodo	30	al nodo	31	por el arco	54
del nodo	31	al nodo	20	por el arco	38
del nodo	20	al nodo	19	por el arco	35
del nodo	19	al nodo	18	por el arco	33
del nodo	18	al nodo	4	por el arco	8
del nodo	4	al nodo	5	por el arco	7
del nodo	5	al nodo	19	por el arco	10
del nodo	19	al nodo	18	por el arco	33
del nodo	18	al nodo	17	por el arco	31
del nodo	17	al nodo	16	por el arco	30

Nodo inicial		Nodo final		Arco usado	
del nodo	16	al nodo	2	por el arco	4
del nodo	2	al nodo	3	por el arco	3
del nodo	3	al nodo	17	por el arco	6
del nodo	17	al nodo	29	por el arco	32
del nodo	29	al nodo	18	por el arco	34
del nodo	18	al nodo	17	por el arco	31
del nodo	17	al nodo	29	por el arco	32
del nodo	29	al nodo	30	por el arco	53
del nodo	30	al nodo	31	por el arco	54
del nodo	31	al nodo	20	por el arco	38
del nodo	20	al nodo	19	por el arco	35
del nodo	19	al nodo	18	por el arco	33
del nodo	18	al nodo	17	por el arco	31
del nodo	17	al nodo	16	por el arco	30
del nodo	16	al nodo	1	por el arco	2

Con una longitud total de 752 unidades

C O N C L U S I O N E S

En este trabajo se presentaron dos problemas clásicos de la teoría de gráficas y la investigación de operaciones como son: el problema del cartero chino y el problema de acoplamiento. Los algoritmos descritos para la solución de estos problemas son de lo más eficientes, si no es que los más eficientes que actualmente existen, lo que hace a este trabajo de gran importancia en las aplicaciones prácticas a problemas reales cuya modelación pueda ser planteada en términos de los modelos ya descritos.

Los algoritmos que se desarrollaron en el texto, están programados en PASCAL o en FORTRAN y se encuentran implantados para computadoras IBM-PC o compatibles. Una breve descripción de la entrada de datos y resultados se bosqueja en el apéndice.

REFERENCIAS

1. Balinski, M. L., "*Labelling to Obtain a Maximun Matching, in Combinatorial Mathematics and its Applications*", University of North Carolina, pag. 585-602, 1967.

2. Berge, C., "*Two Theorems in Graph Theory*", Proc. Nat. Acad. Sci. USA 43(1957), 842-844.

3. Burkard, R. E., y Derigs, U., "*Assignment and Matching Problems: Methods with FORTRAN-Programs*", Lecture Notes in Economics and Mathematical Systems 184, Springer-Verlag, Nueva York, 1980.

4. Edmonds, J. y Johnson, E. L., "*Matching, Euler Tours and the Chinese Postman*", Mathematical Programming, 5, pag. 88-124, 1973.

5. Edmonds, J. y Johnson, E.L, "*Matching: A Well Solved Class of Integer Linear Programs, Combinatorial Structures and Their Applications*", Gordon and Breach, New York, pag 89-92, 1970.

6. Edmonds, J., "*Path, Trees and Flowers*", Canad. J. Math., 17(1965), 449-467.

7. Ford, R.L. y Fulkerson, D. R., "*Solving the Transportation Problem*", reporte RM-1736, The Rand Corporation, Sta. Monica California, 1956.

8. Fulkerson, D. R., "*An Out of Kilter Method for minimal cost flow problem*", Journal Soc. Ind. Appl. Math. 9, I, 1968.

9. Gabow, H., "*An Efficient Implementation of Edmonds' Algorithm for Maximun Matchings on Graphs*", J. ACM 23(1975), 221-234.

10. Gondran, M. y Minoux, M., "*Graphs and Algorithms*", Wiley-Interscience Series in Discrete Mathematics, 1984.

11. Lawler, E. L., "*Combinatorial Optimization: Networks and Matroids*", Holt, Rinehart and Winston, 1775.

12. Minieka, E., "*Optimization Algorithms for Networks and Graphs*", Marcel Dekker, Inc., 1978.

13. Moreno, E., "*Problemas de Aparcamiento*", Tesis de Maestría en Investigación de Operaciones, DEPFI-UNAM, México, D.F., 1987.

14. Papadimitriou, C. y Steiglitz, K., "*Combinatorial Optimization: Algorithms and Complexity*", Prentice-Hall, 1982.

15. Pape, U., Conradt, D., "*Maximales Matching in Graphen, in H. Späth (ed.)*", Ausgewählte Operations Research Software in FORTRAN, Oldenburg, Munich, 1980, pp. 103-114.

16. Rockafellar, R. T., "*Network Flows and Monotropic Programming*", Wiley-Interscience, 1984.

17. Syslo, M., Deo, N. y Kowalik, J., "*Discrete Optimization Algorithms with PASCAL Programs*", Prentice-Hall, Inc., 1983.

A P E N D I C E

Los algoritmos tratados a través de los capítulos 2 a 5 del documento, se encuentran implantados para computadoras IBM-PC o compatibles en un paquete. El programa se ejecuta al dar de alta el sistema operativo con el disketto que lo contiene o si ya se está dentro del sistema MS-DOS, para invocarlo teclee CHINO.

Todos los programas permiten manejar una gráfica de hasta 100 nodos y 100 arcos si su problema es de más de estas dimensiones debe hacer modificaciones sobre los programas fuente. Después de autoejecutarse el programa o invocarlo, aparecerá el siguiente menú:

"MENU PRINCIPAL"
Programa para resolver el problema del cartero chino en redes dirigidas y no dirigidas.
Se cuenta con las siguientes opciones:

1) Creación o corrección de un archivo de datos.

2) Ejecución del problema de acoplamiento máximo.

3) Ejecución del problema de acoplamiento pesado.

4) Ejecución del problema del cartero chino en una gráfica no dirigida.

5) Ejecución del problema del cartero chino en una gráfica dirigida.

6) Desplegar un archivo de datos o de resultados.

7) Salida del menú.

Tecleé el número de la opción que desea

En este momento debe escoger alguna de la opciones tecleando un

número del 1 al 7 y en seguida RETURN.

Aquí aparecerá el letrero: "Favor de esperar un momento." e indicará que se está cargando el programa adecuado de acuerdo con la opción escogida.

Cada una de las opciones del menú se describirán a continuación:

1) Creación o corrección de un archivo de datos.
Esta opción permite crear o corregir un archivo que se empleará en las opciones 2 a 5. El archivo creado aquí, puede usarse en cada una de las opciones 2 a 5 según sea el problema que se quiera resolver. Cuando se teclea esta opción aparecerá el siguiente menú:

CREACION O CORRECCION DE UN ARCHIVO DE DATOS

Para crear o modificar un archivo de datos, recuerde que su problema debe estar en la representacion nodos arcos como se explica en el apendice. Si el problema es no dirigido, el orden en que se dan los nodos a los que incide un arco es indiferente. Pero si el problema es dirigido, entonces el orden debera ser el adecuado.
El numero maximo de arcos permitido es de 100.
Si algun arco de un archivo ya creado quiere borrarse, unicamente cambie el valor de su nodo inicial por cero.
En seguida se piden los siguientes datos:

1) Nombre del archivo de datos que se va a crear o corregir.
2) Titulo o identificacion de la corrida.
3) Numero de nodos en el problema.
4) Numero de arcos en el problema.
5) Nodo en donde el cartero inicia su ruta.
6) Para cada arco, su nodo inicial, su nodo final, y su peso o costo.

Pulse cualquier tecla para continuar

Como se dijo anteriormente, esta entrada de datos es común para las opciones 2 a 5 del menú principal. La gráfica debe especificarse como se explica en la sección 1.3 del documento como lista de los tres arreglos B, D Y Z; (Vea sección 1.3) También los nodos deberán numerarse en orden consecutivo comenzando en 1, hasta el valor máximo, que debe coincidir con el número de nodos del problema. Quizá algunos datos no sean necesarios para alguna de las opciones; como por ejemplo, en el problema de acoplamiento máximo no se requiere peso o costo sobre los arcos, en este caso ese dato no se usará, pero deberá teclearse.

2) Ejecución del problema de acoplamiento máximo.

Esta opción obtiene el acoplamiento máximo en la gráfica como se describe en el capítulo 2. El programa ACOMA implementa el algoritmo descrito en el capítulo 2 para encontrar un acoplamiento de máxima cardinalidad en una gráfica no dirigida que no contenga nodos aislados. El programa sigue lo esbozado en el algoritmo 2.1.

El procedimiento ACOINICIAL obtiene un acoplamiento inicial como se describe en el algoritmo 2.2. Enseguida, para todo nodo expuesto RAIZ se construye un árbol alternante. Si se encuentra una trayectoria aumentante el acoplamiento actual se agranda en la forma que se explica en el algoritmo 2.3. Si no es el caso, el nodo bajo consideración (NBHR) y su acoplado (MATWI) se incluyen al árbol alternante, después de checar su ancestridad (es decir, que no se forma un ciclo impar por el algoritmo 2.4). La variable booleana AGARBOL se hace TRUE. En este caso MATWI se agrega a Q.

La red suministrada a este programa se da en forma de lista ligada hacia adelante (vea sección 1.3) y tiene N nodos y M arcos. El arreglo APUNTADOR, apunta al inicio en el arreglo NODOFIN, que contiene la lista de nodos adyacentes. Como la gráfica es no

dirigida, cada arco se representa dos veces, una por cada uno de sus nodos terminales.

3) Ejecución del problema de acoplamiento pesado.

El programa para resolver el problema de acoplamiento pesado descrito en el capítulo 3 está escrito en FORTRAN y consiste de un programa principal para operaciones de entrada salida y las subrutinas SMP, SCAN1, SCAN2.

En el programa principal se especifican los arreglos y vectores en la proposición DIMENSION. El tamaño puede ser alterado dependiendo del problema particular.

ENTRADA: La entrada de datos en el número par N de nodos y dentro del programa se maneja la parte superior de la matriz de costos C que es simétrica y entera. Sus elementos se leen por columna en la forma siguiente.

$$c_{12}, c_{13}, c_{23}, c_{14}, c_{24}, c_{34}, \ldots, c_{N-2,N}, c_{N-1,N}.$$

En SMP se guardan en el vector CC(I) de longitud

$$K = 0.5*N*(N-1)$$

Y se requieren, un número suficientemente grande

$$SUP > \sum_{i=1}^{N} \left(\sum_{\substack{j=1 \\ j \neq i}}^{N} c_{ij} \right)$$ y la precisión de la máquina.

Para identificar el costo C(I,J) en el vector CC(K) se usa el apuntador P(I), I = 2,...,N con

P(2) = 0

P(I) = P(I-1) + I - 2, I = 3,...,N.

Ahora sea I > J, I,J = 1,2,...,N y

IND = P(I) + J

Entonces se encuentra el elemento C(I,J) como CC(IND).

SALIDA: El programa principal imprime a la salida, el nombre del

problema, el acoplamiento óptimo M se guarda en el vector NMATCH(I) y se imprime a la salida en la forma

I -- NMATCH(I) , I= 1,2,...,N

que significa que el arco que conecta I y NMATCH(I) es un arco acoplado.

Y el valor $\sum_{e_{ij} \in M} c_{ij}$ del acoplamiento óptimo se da.

El programa requiere de una capacidad de almacenamiento de $(1/2)N^2 + 13N$.

4) Ejecución del problema del cartero chino en una gráfica no dirigida.

El programa para resolver el problema del cartero chino en gráficas no dirigidas, descrito en el capitulo 4, está escrito en FORTRAN y consiste de un programa principal para operaciones de entrada salida y las subrutinas CPP, PRUEF1, PRUEF2, SCHR, ERW, KASU Y TOUR. En CPP se determina el conjunto de arcos duplicados y en TOUR se construye la ruta Euleriana.

En el programa principal se especifican los arreglos y vectores en la proposición DIMENSION. El tamaño puede ser alterado dependiendo del problema particular.

ENTRADA: Aparte del número de nodos N y del número de arcos M, CPP requiere, en forma interna, la lista ligada hacia adelante (vea sección 1.3) y el costo de los arcos asociados. Estas listas están contenidas en dos vectores NB y KOST de longitud 2M. Para la identificación de los nodos adyacentes se usa el vector INDEX de longitud N+1. Para cada nodo I la lista de nodos adyacentes se guarda en las posiciones

NB(INDEX(I)),...,NB(INDEX(I+1)-1)

Las listas NB, KOST y INDEX se construyen en el programa

principal. Para este propósito el programa principal requiere para cada nodo I el número GRAD(I) de arcos adyacentes, es decir, el grado del nodo I.

Para cada arco en la gráfica se leen sus nodos adyacentes y peso.

Se requiere la precisión de la máquina EPS y un entero suficientemente grande TCP = $\sum_{e_{ij} \in A} c_{ij}$

También se define un nodo específico donde el cartero comienza su ruta.

SALIDA: El programa principal imprime en la salida el nombre del problema y la lista de arcos con sus pesos asociados.

El nodo donde el cartero comienza su ruta, la lista de arcos duplicados y los nodos que caracterizan la ruta optima del cartero. Así, para cada nodo v la lista asociada

$$L_v(1), \ldots, L_v(k)$$

se especifica.

Finalmente el peso de los arcos duplicados y el peso de la ruta del cartero se dan.

El programa requiere de una capacidad de almacenamiento de 2M + 12N. El tiempo de corrida depende en gran parte de la estructura de la gráfica.

5) Ejecución del problema del cartero chino en una gráfica dirigida.

El programa para resolver el problema del cartero chino en gráficas dirigidas, descrito en el capítulo 5, está escrito en BASIC. En este programa es importante recordar, como se describe en el capítulo 5, que la gráfica es dirigida, esto es, si existe el arco (i,j), unicamente se puede transitar por él, del nodo i al nodo j.

6) Desplegar un archivo de datos o de resultados.

Esta opción permite desplegar en la pantalla un archivo de datos o de resultados creado por el programa. Unicamente se debe proporcionar el nombre del archivo y la extensión del mismo y aparecerá en pantalla.

7) Salida del menú.

Esta opción permite abandonar el sistema CHINO y salir al sistema operativo.