APLICACIONES DE LA TEORÍA DE GRAFOS: BÚSQUEDA DE CAMINOS EN UNA RED Y ANÁLISIS DE SU CONECTIVIDAD

(APPLICATIONS OF THE GRAPH THEORY: SEARCH FOR PATHS IN A NETWORK AND ANALYSIS OF THEIR CONNECTIVITY)

Alfonso Recuero, Dr. Ingeniero de Caminos
Instituto Eduardo Torroja/CSIC
ESPAÑA

Fecha de recepción: 14 - X - 94
403-16

RESUMEN

Se presentan tres algoritmos para la búsqueda de caminos orientados en un digrafo, basados en la generación de un árbol en el que se hace una búsqueda exhaustiva, en amplitud en el primer algoritmo, y en profundidad en el segundo y en el tercero. El primero permite encontrar todos los caminos óptimos entre dos vértices; el segundo permite resolver este mismo problema así como el de hallar los caminos hamiltonianos con origen en un vértice, o los ciclos de cualquier orden, en tanto que el tercero permite encontrar todos los caminos o circuitos eulerianos. Se describen, asimismo, dos algoritmos que hacen uso del mismo tipo de técnicas para el análisis de la conectividad de un grafo. El primero permite separar un grafo no conexo en sus partes conexas, y el segundo permite la detección de puentes en grafos conexos.

SUMMARY

This article presents three algorhitms for the search of oriented paths in a digraph, based on the generation of a tree in which an exhaustive search is performed —breadth— first in the first one and depth-first in the second and third ones. The first one allows us to find the optimum paths between two vertices, the second permits the solution of the same problem and also finds the Hamiltonian paths beginning in one vertex or the cycles of any length, while the third one allows us to find all the paths or the Eulerian circuits. The article also describes two algorhitms that use the same type of techniques for the analysis of the connectivity of a graph. The first one permits the division of a non-connective graph into its connective parts while the second one permits the detection of bridges in connective graphs.

1. Introducción

Considérese una red formada por un conjunto de N vértices, identificados por los números de 1 a N, conectados entre sí, bien por arcos orientados, cada uno de ellos con un valor de recorrido (digrafo), o por arcos bidireccionales (grafo). En un digrafo, dos vértices pueden estar conectados mediante uno o dos arcos, esto es, en uno o ambos sentidos, pudiendo ser distintos sus valores de recorrido, admitiendo la existencia de bucles (arcos que unen a un vértice consigo mismo), en tanto que en un grafo sólo puede haber un arco entre dos vértices, los cuales han de ser distintos.

Llamaremos orden al número de vértices, y tamaño al número de arcos.

Dos vértices conectados por un arco, o dos arcos con un extremo común, diremos que son adyacentes.

El grafo asociado a un digrafo es el resultante de transformar todos los arcos de éste en bidireccionales, suprimiendo los arcos duplicados y los que conectan un vértice consigo mismo.

Vamos a describir cuatro problemas clásicos de búsqueda de caminos orientados en digrafos, teniendo en cuenta el sentido de recorrido de los arcos, y dos sobre la conectividad en grafos, presentando para todos ellos algoritmos basados en la construcción de un árbol, que proporcionan soluciones eficaces a los mismos.

Los grafos constituyen una herramienta de gran utilidad en la resolución de numerosos problemas en una gran variedad de áreas. Por citar sólo algunas de ellas mencionaremos: flujos en redes, árboles genealógicos, distribución de espacios arquitectónicos, emparejamientos, programación de actividades, análisis de estructura, resultados de torneos, coloreado de mapas, coloreado de análisis lingüísticos, diversos juegos, circuitos electrónicos, etc. Si bien su uso se remonta a muchos siglos atrás, su desarrollo actual puede considerarse iniciado con los trabajos de Euler. Es un área de las matemáticas en las que no existe una nomenclatura generalmente aceptada, tal vez por que son muchos los autores que han realizado aportaciones de interés, sin que haya una figura dominante.

De hecho, hay todavía numerosos problemas sin resolver, que se van ampliando a medida que se profundiza en su estudio. Vale aquí el principio: "Todo problema, por complicado que sea, al estudiarlo detenidamente se complica más." Algunos nombres de matemáticos que han hecho aportaciones importantes, tanto teóricas como en forma de algoritmos son: Hamilton, Kirchoff, König, Tutte, Kuratowski, Menge, Verge, Girdle, Dijkstra, Ramsey, Fulkerson, Harari, etc. Hay numerosos ar-tículos y libros sobre temas de grafos. Se referencian algunas obras que permitirán al lector interesado profundizar tanto en los aspectos teóricos como en las aplicaciones concretas [Knuth, Gould, Abellanas, Elmaghrabi, Dolan, Recuero *et al.*].

Este trabajo forma parte de los estudios previos del proyecto de investigación "Desarrollo, mejora e integración de sistemas CAD/CAE en Construcción", que se lleva a cabo en el Instituto Eduardo Torroja, y está financiado por la DGICYT, con el código PB93/0111.

1.1. Caminos y circuitos en un digrafo

En el primer problema estudiado, se trata de encontrar todos los posibles caminos que conecten dos puntos, de modo que la suma de los valores de recorrido de sus arcos sea mínima. Como condición adicional, es posible prohibir que algunos puntos estén incluidos en dichos caminos. Puede tratarse, por ejemplo, de una red de comunicaciones, en cuyo caso los valores de recorrido pueden ser tiempos. Las soluciones representan los caminos alternativos para desplazarse de un punto a otro, y los vértices prohibidos pueden ser puntos a evitar por haberse producido algún problema en los mismos.

Un caso práctico de aplicación es la consulta del camino óptimo entre dos estaciones de una red de metro. Para describir la red, cada estación da lugar a un vértice correspondiente al vestíbulo y otro por cada una de las líneas que pasan por ella. El vértice del vestíbulo se une con el de cada línea con un arco de ida con tiempo igual al tiempo de desplazamiento más el tiempo medio de espera de la línea, y otro de vuelta, con el tiempo de desplazamiento. Entre cada dos vértices de línea de una estación hay dos arcos con unos tiempos de recorrido iguales al tiempo de desplazamiento más el tiempo medio de espera de la línea destino.

Entre cada dos estaciones consecutivas de una línea hay dos arcos con igual valor de recorrido en ambos sentidos, igual al tiempo de recorrido entre ellas. La descripción de la red puede variar según las horas y los días, cambiando básicamente los tiempos medios de espera, que dependen de la frecuencia de paso de los trenes. En caso de existir problemas en alguna línea, bastará considerar las estaciones afectadas como vértices prohibidos.

El segundo problema estudiado es el de determinar los caminos, óptimos o todos, que recorran todos los vértices empezando por un vértice prefijado. Este problema es también conocido como el del viajante [Bellmore y Nemhauser]. A tales caminos se los denomina hamiltonianos, en honor del matemático irlandés William R. Hamilton que comercializó un puzzle basado en las aristas de un dodecaedro, el cual debía ser recorrido pasando por todos los vértices, a los que asignó nombres de ciudades, terminando en el inicial. En la Figura 1 se muestra una perspectiva de las aristas del dodecaedro junto con una representación plana del mismo, en la que se ha marcado un camino hamiltoniano. De nuevo, puede imponerse como condición adicional la

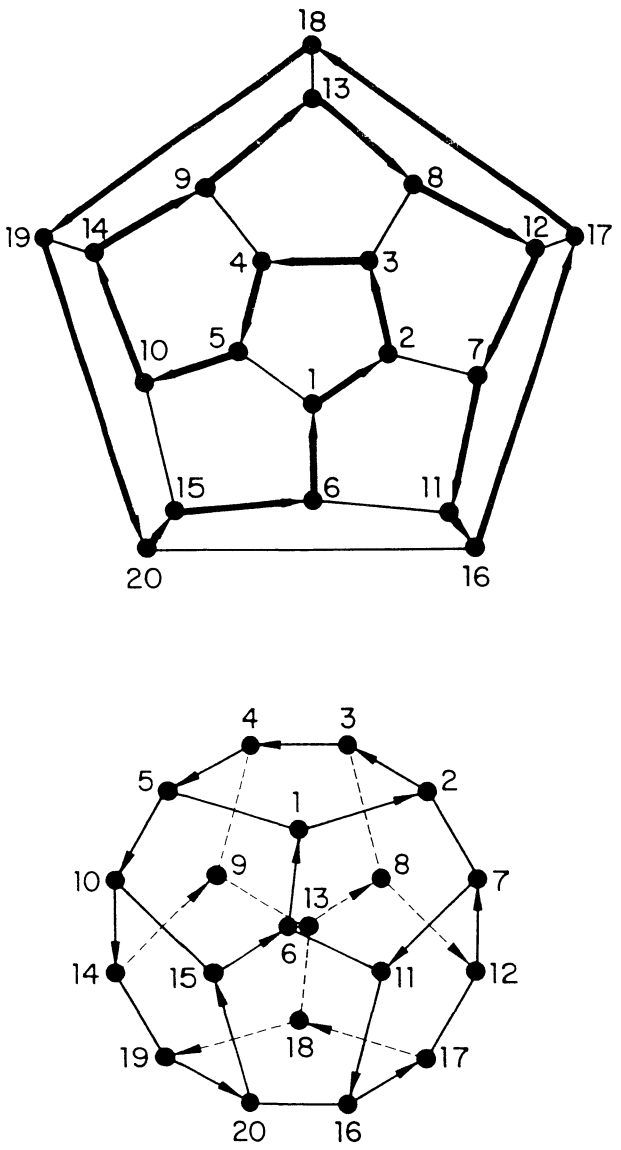

Fig. 1.

prohibición de paso por algunos vértices determinados. Si desde el último vértice de un camino hamiltoniano existe un arco que le una al primero tendríamos un ciclo hamiltoniano. Evidentemente, para estudiar los ciclos hamiltonianos en la red, pueden estudiarse los caminos hamiltonianos con origen en cualquier vértice.

Otro ejemplo de circuitos hamiltonianos es el problema del caballo de ajedrez. Dado un tablero de M • N escaques, encontrar los recorridos de un caballo que pasen por todos los escaques sin pasar dos veces por el mismo. Las soluciones son itinerarios hamiltonianos. Como se verá, es necesario generar todos los posibles itinerarios sin repetir el paso por un vértice, lo que con un tablero ordinario, de 8 x 8, conduce a tiempos prohibitivos. Sin embargo, si se trata de obtener sólo algunas soluciones, pueden emplearse técnicas, heurísticas de búsqueda que aceleren la aparición de soluciones, tal como acceder preferentemente a los vértices de menor grado, dentro de las posibilidades de movimiento desde cada vértice [Morán].

El tercer problema es el de enumerar todos los ciclos de orden M, esto es, los conjuntos de M vértices distintos unidos cíclicamente por arcos. En el caso en que M coincidiese con N, estaríamos de nuevo ante los ciclos hamiltonianos.

El cuarto problema es el de encontrar todos los caminos o circuitos que recorran una sola vez todos los arcos de la red (caminos o circuitos eulerianos). En este problema no tiene sentido la consideración de vértices prohibidos ni la del valor de recorrido de los arcos, ya que todos han de estar incluidos. Se denominan así en honor a Leonhard Euler, iniciador de la moderna teoría de grafos, que demostró la imposibilidad de recorrer los siete puentes que unían las orillas del río Praguel y dos islas (Figura 2), en la ciudad prusiana de Königsberg, en un paseo dominical, sin pasar dos veces por el mismo. Llamaremos grado de un vértice al número de arcos que concurren en él. Si todos los arcos fuesen bidireccionales, el problema sólo tiene solución cuando el número de vértices de grado impar es 0 ó 2 (condición necesaria y suficiente). En el primer caso, los caminos serán cerrados (circuitos), y en el segundo tendrán sus extremos en dichos vértices. Esta condición seguirá siendo necesaria cuando algunos arcos tengan uno de sus sentidos de recorrido prohibido, pero no será suficiente. En este último casos debe considerarse el número de arcos de entrada y de salida a cada vértice (grados de entrada y salida); si

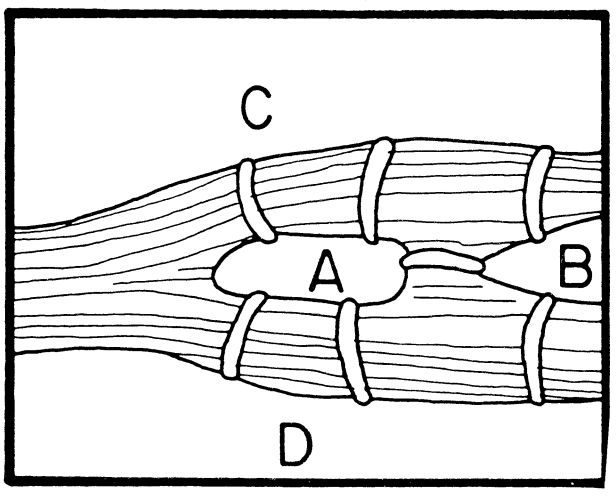

Fig. 2.

ambos grados coinciden, se dice que el vértice está equilibrado. Si todos los vértices están equilibrados y el grafo es conexo, será posible encontrar circuitos eulerianos.

Considérese el problema de las fichas del dominó de orden N. Llamaremos así al conjunto de N* (N + 1)/2 fichas cada una de las cuales tiene una pareja de los números desde el 0 al N-1. Se trata de averiguar de cuántas formas pueden colocarse todas las fichas, de modo que las mitades en contacto de dos fichas contiguas tengan el mismo número. Si establecemos un grafo completo de orden N, en el que cada vértice está unido a todos los demás, y además cada vértice unido a sí mismo mediante un bucle, cada uno de los arcos representará una de las fichas, representando los bucles las fichas dobles. Las soluciones del problema serán los circuitos o caminos eulerianos en dicho grafo. De la condición necesaria y suficiente se deduce que ningún dominó de orden par, excepto el de orden 2, tiene solución.

1.2. Conectividad de un grafo

El primer problema de conectividad que se va a considerar es el de determinar si un grafo es conexo, y de no serlo, separar sus partes conexas. Un grafo se dice conexo cuando, considerado cualquier par de vértices, puede encontrarse algún camino que los una. Un caso práctico de aplicación de este problema surge en la renumeración automática de estructuras para reducir el ancho de banda de su matriz de rigidez. En general, los algoritmos descritos en la literatura especializada están concebidos para actuar sobre una estructura única, pero en el proyecto automatizado pueden surgir conjuntos de estructuras que el sistema considera como una sola. En tales casos, el algoritmo de reducción debe separar las distintas estructuras independientes, antes de proceder con cada una de ellas por separado. Este problema se describe en [Recuero y Gutiérrez].

El segundo problema es el de la detección de puentes en un grafo conexo. Diremos que un arco es un puente, o istmo, cuando su supresión transformaría un grafo conexo en uno inconexo. Se trata de arcos críticos, pues en una red de distribución o de comunicaciones, un problema en un puente dejaría partes de la red aisladas. Si todos los arcos fuesen puentes, el grafo sería un árbol.

Los problemas de conectividad en digrafos pueden reducirse a los del grafo asociado, para el estudio de la simple conectividad, o hacer uso de las soluciones a los problemas del apartado anterior, en los casos de conectividad fuerte (que entre cualquier pareja de vértices exista un camino orientado), o del enraizamiento (que exista un vértice desde el que se pueda llegar a todos los demás mediante un camino orientado).

2. Almacenamiento de la descripción de la red

La forma más compacta de almacenar la descripción de la red, y la más eficiente de manejarla a los efectos de los algoritmos que se exponen, es utilizar los vectores unidimensionales siguientes:

— T. Este vector entero, con tantos elementos como arcos, contiene los vértices finales de éstos. Se denomina "lista de adyacencias" y contiene toda la información de la matriz de adyacencias de la que se han suprimido los elementos nulos.

— DIST. Entero o real, según sean los valores asignados a los recorridos, con igual número de componentes que T, que contiene los valores de recorrido de los arcos. Sólo es necesario cuando se busquen caminos óptimos.

— ARCO. Entero, con igual número de componentes que T, que contiene el número de orden del arco. Sólo es necesario utilizarlo en la búsqueda de caminos eulerianos y en la detección de puentes, problemas en los que han de considerarse todos los arcos. Puede considerárselo también como lista de adyacencias, dando en tal caso un direccionamiento indirecto de los vértices adyacentes.

— POST. Entero de N + 1 componentes, que indica que todos los arcos con origen en el vértice I son aquellos cuyos extremos ocupan las posiciones POST(I) a POST(I + 1) - 1 del vector T; sus valores de recorrido ocupan las mismas posiciones del vector DIST, y sus números de orden ocupan las mismas posiciones del vector ARCO.

Esta información es poco variable, pues sólo cambia si se modifican el número de vértices o los arcos, por lo que puede generarse una sola vez, almacenarla en un archivo de disco y leerla antes de aplicar alguno de los algoritmos a un caso concreto. Permite, además, la programación de los algoritmos utilizando sólo matrices unidimensionales, sin punteros, lo que hace posible su implementación con lenguajes de programación simples, tal como QuickBASIC.

3. Algoritmo 1

El primer algoritmo consiste en generar un árbol en cuyo primer nivel se sitúe el vértice inicial y en los sucesivos los vértices conectados con él mediante un número creciente de arcos. A medida que se genera el árbol, se realiza una comprobación exhaustiva en amplitud, desechando los nudos que incumplen las comprobaciones, prosiguiendo la generación en tanto que los nudos puedan formar parte de los caminos solución. Las soluciones serán recorridos en profundidad del árbol.

A cada nudo del árbol se le asocia el número del vértice a que corresponde (etiqueta del nudo), la suma de los valores de recorrido de los arcos que le unen al primer nivel (valor del nudo), así como el nudo del nivel anterior del que procede (a efectos de facilitar el recorrido inverso del árbol). Además, para cada vértice accedido, se conserva su mínima distancia al vértice inicial.

3.1. Camino óptimo entre dos vértices

En la búsqueda de los caminos óptimos entre dos puntos, INI y FIN, las comprobaciones que se efectúan antes de generar un nuevo nudo del árbol son:

a) El valor del nudo origen coincide con la mínima distancia, hasta el momento, de su vértice etiqueta hasta INI. Sirve para evitar el progreso del árbol cuando, en el mismo nivel, se ha alcanzado el mismo vértice por un camino más corto.

b) El nudo destino no corresponde a un vértice prohibido. Sirve para evitar este tipo de vértices.

c) El nudo destino no es el vértice FIN. En tal caso se habría alcanzado una posible solución.

d) El valor del nudo destino es menor que la mejor distancia conseguida, hasta el momento, desde INI a FIN. En caso contrario no conduciría a una solución mejor que la ya alcanzada.

e) El valor del nudo destino no supera la menor distancia conseguida, hasta el momento, desde INI al mismo vértice. Sirve para evitar la formación de ciclos.

La comprobación a) es previa a considerar los arcos con origen en la etiqueta (antes de generar cada subárbol), en tanto que las restantes han de hacerse para cada arco con origen en la misma. Las comprobaciones han de hacerse en el orden indicado, si bien las dos últimas pueden hacerse simultáneamente.

El incumplimiento de cualquiera de ellas implica no generar el o los nudos destino.

El incumplimiento de la condición c) implica un proceso adicional: si se accede al vértice FIN, y su distancia a INI es menor que la mejor ya conseguida, se iniciará una lista de soluciones correspondiente al nuevo valor; si fuese igual, se añadiría a la lista actual; si fuese mayor, se despreciaría.

La generación del árbol se continúa hasta que no se pueda generar un nuevo nivel. La lista de soluciones existente en ese momento será la solución del problema.

Con estas condiciones, el máximo número posible de niveles es N-1-NVP (NVP es el número de vértices prohibidos), pues no puede aparecer dos veces un mismo vértice en un recorrido en profundidad del árbol. Sin embargo, un vértice puede repetirse en el mismo o en distintos niveles, si bien cada nueva aparición implica que su distancia a INI iguala o mejora la mejor anterior. En caso de aparecer más de una vez en el mismo nivel, sólo podrá progresar el árbol cuando sus valores coincidan con la mínima distancia desde INI a la etiqueta, evitando así la formación de soluciones espúreas.

3.2. Almacenamiento de la información del árbol

La forma más compacta de almacenar la información manejada por el algoritmo es utilizar los siguientes vectores unidimensionales:

— A, que contiene las etiquetas de los nudos del árbol, colocados en cola a medida que se genera el árbol.

— DINI, que contiene los valores de los nudos del árbol, y se genera paralelamente a A.

— B, que contiene la posición en A del nudo del que procede cada nudo, y se genera también en paralelo con A.

— C, que contiene las mejores distancias conseguidas, en cada momento, desde INI hasta cada vértice. Puede utilizarse también para marcar los vértices prohibidos.

A y B son vectores enteros, en tanto que DINI y C han de ser del tipo necesario para contener valores de recorrido acumulados. En la lista actual de soluciones basta conservar la distancia que la define, el número de soluciones de la lista y, para cada una de ellas, la posición en A del nudo anterior a FIN. Con este último dato y la información contenida en B, la reconstrucción de los caminos solución es inmediata, recorriendo el camino en sentido inverso al de generación.

3.3. Ejemplo 1

El primer ejemplo es una malla regular de 9 vértices y 24 arcos, cuyo esquema se muestra en la Figura 3, y en la que se atribuye un peso 1 a cada arco. Si no hay vértices prohibidos, existen 6 caminos óptimos desde 1 hasta 9, con un valor 4, a saber: 1-2-3-6-9, 1-2-5-6-9, 1-2-5-8-9, 1-4-5-6-9, 1-4-5-8-9 y 1-4-7-8-9.

En este caso, el árbol se desarrolla en el algoritmo 1 hasta el nivel 4, pues todas las soluciones se producen en el nivel 5. El vértice 5 aparece dos veces en el nivel 3, y los vértices 6 y 8 lo hacen tres veces en el nivel 4, pero en todos los casos los nudos con igual etiqueta tienen el mismo valor. Si el vértice 5 estuviese prohibido, las soluciones se reducirían a 2, a saber: 1-2-3-6-9 y 1-4-7-8-9.

Si no hay vértices prohibidos, pueden encontrarse 8 caminos hamiltonianos todos ellos con igual valor, con origen en el vértice 1, a saber: 1-2-3-6-5-4-7-8-9, 1-2-3-6-9-8-5-4-7, 1-2-3-6-9-8-7-4-5, 1-2-5-4-7-8-9-6-3, 1-4-5-2-3-6-9-8-7, 1-4-7-8-5-2-3-6-9, 1-4-7-8-9-6-3-2-5 y 1-4-7-8-9-6-5-2-3.

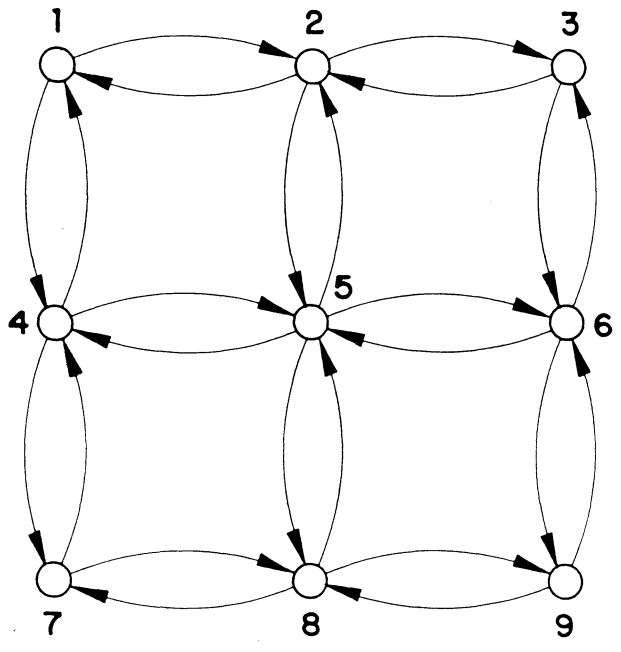

Fig. 3.

Puesto que todos los caminos hamiltonianos terminan en vértices que no tienen conexión con el inicial, no existe en la red ningún circuito hamiltoniano.

Si el vértice 5 estuviese prohibido, estos caminos se reducen a 2, a saber: 1-2-3-6-9-8-7-4 y 1-4-7-8-9-6-3-2. En este caso, ambos caminos dan lugar a circuitos hamiltonianos. Si el vértice 2 estuviese prohibido, no existiría ningún camino hamiltoniano en toda la red.

Si se cambiase el arco 2-5, asignándole un valor de recorrido de 2, en la misma red anterior, al pedir los caminos entre 1 y 9, sin vértices prohibidos, se obtendrían 4 soluciones. En este caso, también aparece el vértice 5 como etiqueta de dos nudos del nivel 2, pero el primero que aparece tiene un valor más alto, por lo que el árbol no se desarrolla a partir de él cuando se utiliza el algoritmo 1. En este caso, si no hay vértices prohibidos, de los 8 caminos hamiltonianos con origen en 1, 3 son óptimos, a saber: 1-2-3-6-5-4-7-8-9, 1-2-3-6-9-8-5-4-7 y 1-2-3-6-9-8-7-4-5.

4. Algoritmo 2

El segundo algoritmo se basa, asimismo, en la generación del árbol con raíz en INI, como en el caso anterior, pero en este caso se realiza una búsqueda en profundidad. El método utiliza tres procedimientos: el primero, de avance al siguiente nivel del árbol; el segundo, de retroceso al nivel anterior; y el tercero, de comprobación de la posible solución. Sólo es necesario conservar la rama del árbol que se está examinando, las etiquetas y los valores de sus nudos, junto con sendos vectores que indican, para cada vértice, si éste está contenido en la rama y el número de orden de la última arista con origen en el vértice que se ha analizado.

El procedimiento de avance comprueba para el nudo del último nivel alcanzado, si se ha llegado a una posible solución, pasando, en tal caso al procedimiento de comprobación de la misma. En caso contrario, trata de progresar al siguiente nivel utilizando el arco siguiente al último utilizado, con origen en el vértice del nudo. Si puede progresar, se anotará el nuevo vértice como utilizado, se lo

incluirá en el siguiente nivel y se volverá a aplicar el procedimiento de avance, y en caso contrario se pasará al procedimiento de retroceso.

El procedimiento de retroceso comprueba, en primer lugar, si se está en el nivel 1, en cuyo caso se habría concluido la ejecución del algoritmo. En caso contrario, se cancela la lista de arcos utilizados con origen en el vértice del presente nivel, se elimina dicho vértice de la lista de los incluidos, se retrocede un nivel y se pasa al procedimiento de avance.

En el procedimiento de comprobación de la posible solución se verifica si ésta existe y es válida. En caso afirmativo, se inicia o completa la lista de soluciones, según sea el caso.

A continuación, se pasa al procedimiento de retroceso.

Las soluciones alcanzadas han de ser almacenadas íntegramente, pudiendo hacerlo en archivos de disco si fuese necesario.

4.1. Caminos y ciclos hamiltonianos

En la búsqueda de los caminos hamiltonianos con origen en el vértice INI, las comprobaciones a realizar en el procedimiento de avance son:

a) Se ha alcanzado el nivel N-NVP. En tal caso, se pasaría al procedimiento de comprobación de la posible solución.

b) Se ha agotado la lista de arcos con origen en el vértice. En tal caso, se pasaría al procedimiento de retroceso.

Si se superan las comprobaciones a) y b), se considera el siguiente arco de la lista. Se efectúan entonces las siguientes comprobaciones:

c) El vértice destino no está ya incluido. Sirve para evitar la formación de ciclos.

d) El valor del nudo no supera el valor de la mejor solución obtenida. Sólo se aplicará si se buscan los caminos óptimos.

Si se superan las comprobaciones c) y d), se podrá progresar al siguiente nivel. En cualquier caso, se volverá a aplicar el procedimiento de avance.

En el procedimiento de retroceso se comprueba si se está en el nivel 1, en cuyo caso se habría concluido la ejecución del algoritmo. En caso contrario, se cancela la lista de arcos utilizados con origen en el vértice del presente nivel, se elimina dicho vértice de la lista de los incluidos, se retrocede un nivel, y se pasa al procedimiento de avance.

En el procedimiento de comprobación, en el caso de caminos, no precisa verificar la validez, a menos que se busquen caminos óptimos, en cuyo caso se debe comprobar que el valor de la solución alcanzada no supera el mejor valor previamente alcanzado. Si se buscasen los ciclos hamiltonianos, bastaría con comprobar, una vez encontrado un camino hamiltoniano, si los vértices inicial y final son adyacentes. En este caso, las soluciones obtenidas no dependen del vértice inicial.

4.2. Ciclos de orden M

Si se buscasen todos los ciclos de orden M (M>2), se consideraría como vértice inicial el 1, en primer lugar, y las comprobaciones a realizar en el procedimiento de avance son:

a) Se ha alcanzado el nivel M. En tal caso, se pasaría al procedimiento de comprobación de la posible solución.

b) Se ha agotado la lista de arcos con origen en el vértice. En tal caso, se pasaría al procedimiento de retroceso.

Si se pasan las comprobaciones a) y b), se considera el siguiente arco de la lista. Se efectúa entonces la siguiente comprobación:

c) El vértice destino no está ya incluido. Sirve para evitar la formación de ciclos de menor orden. Si se supera esta comprobación, se podrá progresar al siguiente nivel. En cualquier caso, se volverá a aplicar el procedimiento de avance.

En el procedimiento de retroceso, se realizan las siguientes comprobaciones:

a) Si el nivel actual es el 1 y el vértice inicial es anterior al N - M + 1, se califica dicho vértice inicial como prohibido, se adopta como vértice inicial al siguiente vértice, se inicializa la lista de arcos incluidos, y se pasa al procedimiento de avance.

b) Si el nivel actual es el 1 y el vértice inicial es el N - M + 1, el algoritmo se da por concluido.

Si se superan las comprobaciones a) y b), se elimina el vértice del nivel actual de la lista de los ya utilizados, se anula la lista de arcos con origen en dicho vértice ya utilizados, se disminuye en 1 el nivel actual, y se pasa al procedimiento de avance.

En el procedimiento de comprobación de la posible solución, se verifica si existe un arco que une el vértice del nudo de nivel M con el inicial, incorporando la solución a la lista, en caso afirmativo. Se pasará después al procedimiento de retroceso.

Si N coincidiese con M, se obtendrían los ciclos hamiltonianos.

4.3. Camino óptimo entre dos vértices

Las comprobaciones a efectuar en el procedimiento de avance, cuando se buscan los caminos óptimos entre dos vértices INI y FIN, coinciden con las indicadas para los caminos hamiltonianos, salvo que la comprobación consiste en ver si se ha alcanzado el vértice FIN.

Las comprobaciones a realizar en los procedimientos de retroceso y de comprobación de la posible solución coinciden con las descritas para la búsqueda de caminos hamiltonianos. En este algoritmo, la parte de árbol a conservar en memoria es menor que en el algoritmo 1, pero puede haber que desarrollar más extensamente el árbol, sobre todo si el número de arcos del camino solución no es grande. Por otra parte, la descripción de las soluciones obtenidas ha de conservarse íntegra aparte, en tanto que en el algoritmo 1 sólo ha de conservarse un puntero por solución, ya que el resto de la información está en la parte de árbol conservada.

4.4. Almacenamiento de la información

La forma más compacta de almacenar la información manejada por el algoritmo es utilizar los siguientes vectores unidimensionales:

— A, que contiene las etiquetas de los nudos de la parte del árbol que se conserva. Puede contener hasta N-NVP enteros.

— DIST, que contiene los valores de los nudos conservados. Se genera en paralelo a A.

— YA, que indica para cada vértice si está incluido en A. Puede utilizarse para indicar los vértices prohibidos.

— LAST, que indica para cada vértice el último arco con origen en él, y por tanto forma parte del camino actualmente considerado.

Los vectores A, YA y LAST son enteros, en tanto que DINI será del tipo adecuado para contener valores acumulados. De hecho, YA puede estar formado sólo por ceros y unos. La lista de soluciones a conservar debe contener dichas soluciones completas, pues se pierden en el proceso posterior. Pueden conservarse en memoria central o, si fuesen muchas y grandes, pueden almacenarse en un archivo de disco de acceso secuencial.

4.5. Ejemplo 2

El segundo ejemplo es el grafo "P", de Petersen, llamado así en honor del matemático danés que lo incluyó en una de sus obras, aunque ya había sido utilizado con anterioridad, grafo no plano que aparece frecuentemente en teoría de grafos por su gran interés, que consta de 10 vértices y 15 arcos no dirigidos, esto es, que pueden ser recorridos en ambos sentidos. En cada vértice concurren tres arcos, y posee una gran simetría, pues todos sus vértices son equivalentes. La Figura 4 muestra una representación de este grafo, del que se hace un interesante estudio en Chartran y Wilson. En él se pueden encontrar 24 caminos hamiltonianos con origen en

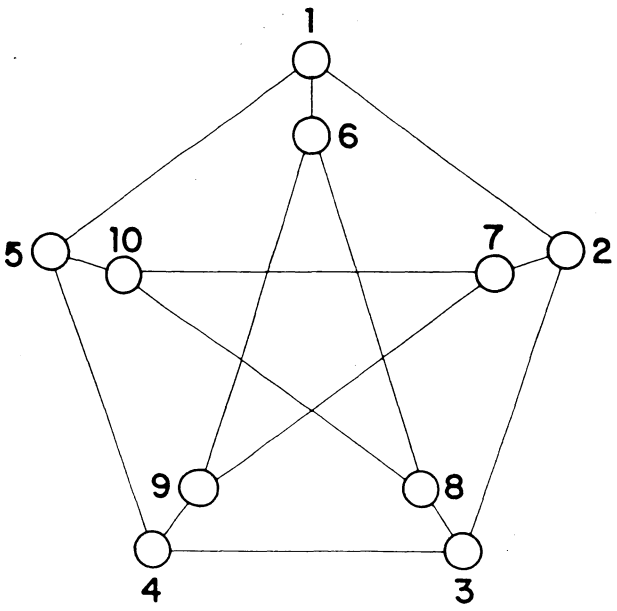

Fig. 4.

cualquier vértice, si no se consideran vértices prohibidos, y 8 caminos si hubiese un vértice prohibido. Se pueden encontrar 24 ciclos de orden 5, 20 de orden 6, 30 de orden 8, y 40 de orden 9, si no se consideran vértices prohibidos. No hay ciclos de orden 3, 4, 7 ni 10. Si cualquier vértice se considera prohibido, entonces el grafo admite 4 circuitos hamiltonianos (40 circuitos de orden 9/10 vértices), esto es, se trata de un grafo hipo hamiltoniano. Así, si se prohibiese el vértice 5, tendríamos los circuitos: 1-2-3-4-9-7-10-8-6-1, 1-2-7-10-8-3-4-9-6-1, 1-6-8-10-7-9-4-3-2-1 y 1-6-9-4-3-8-10-7-2.

En el ejemplo 1, sólo se forman circuitos hamiltonianos cuando se prohibe uno de los vértices de grado par.

5. Algoritmo 3

El tercer algoritmo es muy semejante al segundo, realizando también una búsqueda en profundidad del árbol con origen en el vértice elegido como inicial. Utiliza tres procedimientos: el primero de avance al siguiente nivel del árbol, el segundo de retroceso al nivel anterior, y el tercero de anotación de la solución. Sólo es necesario conservar, para la rama del árbol que se está examinando, las etiquetas de sus nudos, junto con sendos vectores que indican, para cada arco si está contenido en la rama y, para cada nivel, la posición en T del arco que se ha utilizado en dicho nudo para pasar al siguiente nivel.

El procedimiento de avance comprueba, para el nudo del último nivel alcanzado, en primer lugar si se ha llegado a una solución, pasando, en tal caso, al procedimiento de anotación de la misma. En caso contrario, trata de progresar al siguiente nivel utilizando el arco siguiente al último utilizado en ese nivel, con origen en el vértice del nudo. Si puede progresar, se anotará el nuevo arco como utilizado, tanto en la lista de arcos utilizados como en la de los utilizados en cada nivel; se incluirá el vértice destino en el siguiente nivel y se volverá a aplicar el procedimiento de avance, y en caso contrario se pasará al procedimiento de retroceso.

El procedimiento de retroceso comprueba en primer lugar si se está en el nivel 1, en cuyo caso se habría concluido la ejecución del algoritmo. En caso contrario, el arco utilizado para llegar al nivel actual se elimina de la lista de arcos utilizados, se retrocede un nivel, y se pasa al procedimiento de avance.

En el procedimiento de anotación de la solución, se incluye a la misma en la lista de soluciones y se pasa después al procedimiento de retroceso. Las soluciones alcanzadas han de ser almacenadas íntegramente, pudiendo hacerlo en archivos de disco si fuese necesario.

5.1. Caminos y circuitos eulerianos

En las búsqueda de los caminos eulerianos con origen en el vértice INI, se debe comprobar si dicho vértice puede ser origen de los mismos, verificando si cumple la condición necesaria. Las comprobaciones a realizar en el procedimiento de avance son:

a) Se ha alcanzado el nivel W + 1. En tal caso, se pasaría al procedimiento de anotación de la solución.

b) Se ha agotado la lista de arcos utilizados en dicho nivel con origen en el vértice. En tal caso, se pasaría al procedimiento de retroceso.

Si se superan las comprobaciones a) y b), se considera el siguiente arco de la lista. Se efectúa entonces la siguiente comprobación:

c) El arco a utilizar para pasar al siguiente nivel no está ya incluido. Si se supera la comrpobación c), se podrá progresar al siguiente nivel. En cualquier caso, se volverá a aplicar el procedimiento de avance.

En el procedimiento de retroceso se comprueba si se está en el nivel 1, en cuyo caso se habría concluido la ejecución del algoritmo. En caso contrario, se elimina el arco con el que se ha alcanzado el nivel actual de la lista de arcos ya incluidos en la rama en estudio y se retrocede un nivel, y se pasa al procedimiento de avance.

5.2. Almacenamiento de la información

La forma más compacta de almacenar la información manejada por el algoritmo es utilizar los siguientes vectores unidimensionales:

— A, que contiene las etiquetas de los nudos de la parte del árbol que se conserva. Puede contener hasta W + 1 enteros.

— YA, que indica para cada arco si está incluido en la rama en examen.

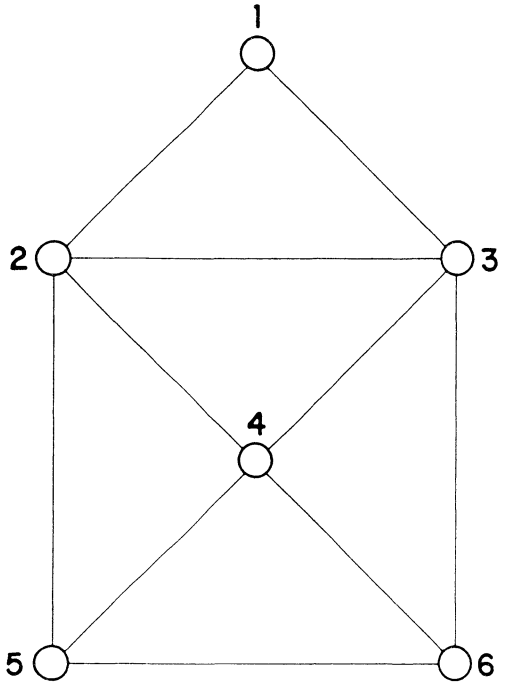

Fig. 5.

— LAST, que indica, para cada nivel, el número de orden en T del arco utilizado para pasar al siguiente nivel.

Los vectores A, YA y LAST son enteros, de hecho, YA puede estar formado sólo por ceros y unos. La lista de soluciones a conservar debe contener dichas soluciones completas, pues se pierden en el proceso posterior. Pueden conservarse en memoria central o, si fuesen muchas y grandes, pueden almacenarse en un archivo de disco de acceso secuencial.

5.3. Ejemplo 3

El tercer ejemplo es el grafo de la figura 5, que consta de 6 vértices y 10 arcos bidireccionales. En los vértices 5 y 6 concurren 3 arcos, mientras que en los restantes vértices concurre un número par de arcos. En consecuencia, los caminos eulerianos tendrán necesariamente sus extremos en los vértices 5 y 6. El número de caminos eulerianos con origen en el vértice 5 es de 120, al igual que los que tienen origen en 6, los mismos que con origen en 5 pero recorridos en sentido inverso. Si el arco 5-6 se hiciese de sentido único, el número de caminos se reducirá a 88 con origen en 5 y a 32 con origen en 6, que suman 120 pues permite los caminos con origen en 5 inversos a los que prohibe con origen en 6, y viceversa.

6. Conectividad de grafos

Vamos ahora a suponer que la red en estudio es un grafo, esto es, que todos sus arcos son bidireccionales. En el caso de digrafos, no son de general aplicación los conceptos que se utilizan. Así, puede ocurrir que sea posible acceder de un vértice a otro, pero no a la inversa, o puede ocurrir que la supresión de un arco produzca esta situación. Pueden también existir fuentes o sumideros, etc. Por consiguiente, cuando se hace referencia a la conectividad en digrafos se entiende que se habla del grafo asociado.

6.1. Separación en partes de grafos inconexos

El algoritmo para separar un grafo inconexo en sus partes o subgrafos conexos es muy simple. Consiste en desarrollar en amplitud el árbol con origen en un nudo, etiquetando los vértices que incluye como pertenecientes al subgrafo en curso, y cortando su desarrollo cuando el vértice destino ya haya sido etiquetado. Cuando no sea posible progresar, se habrá completado un subgrafo conexo, y se procederá a la búsqueda del siguiente. El orden en el que se van utilizando los vértices como origen de un nuevo subárbol es el mismo en el que se van etiquetando. El algoritmo utiliza dos procedimientos: el de búsqueda del primer vértice de un nuevo subgrafo y el de avance por el árbol.

El procedimiento de búsqueda consiste en localizar el primer vértice no etiquetado, al que se le asignará la etiqueta del nuevo subgrafo, se anotará en la lista ordenada de vértices origen de subárbol, se incrementará la cuenta de vértices ya asignados, y se pasará al procedimiento de avance. cuando todos los vértices estén ya etiquetados, el algoritmo habrá concluido.

El procedimiento de avance comprueba, en primer lugar, si el número de vértices ya etiquetados coincide con el número de orden del último vértice de la lista de orígenes de subárbol que ha sido utilizado. En tal caso, se habría completado un subgrafo conexo, y se pasaría al procedimiento de búsqueda. De no ser así, se tomará, de la lista ordenada de vértices etiquetados, el siguiente vértice que será origen de subárbol, revisando todos los arcos con origen en él. Si el vértice destino no estuviese etiquetado, se le asigna la etiqueta, se incrementa la cuenta de vértices etiquetados, y se agrega a la lista ordenada de vértices ya etiquetados. Una vez agotada la lista de arcos, se comprueba si ya se han eti-

quetado todos, en cuyo caso el algoritmo habría concluido. En otro caso, se pasaría al procedimiento de avance.

La información utilizada por este algoritmo puede almacenarse en dos vectores enteros unidimensionales, de N elementos, tales como:

— LANUD, que contiene, para cada vértice, la etiqueta del subgrafo a que pertenece.

— YA, que contiene la lista de los vértices ya etiquetados, en el orden en que lo han sido. A partir de esta matriz, puede construirse el vector de direccionamiento indirecto que utilizan los algoritmos de renumeración de estructuras para reducir el ancho de banda de su matriz de rigidez, que indica el nuevo número asignado a cada nudo. Este vector permite utilizar la numeración inicial en la entrada de datos y salida de resultados, y la optimizada en la formación de la matriz de rigidez y en la resolución del sistema de ecuaciones lineales.

6.2. Identificación de componentes y puentes

Denominaremos componente al conjunto de vértices conectados de modo que puedan establecerse circuitos que incluyan a cualquiera de ellos. Esta condición implica la no existencia de puentes entre dos vértices de una misma componente, puesto que si se suprimiese un arco de un circuito, todos sus vértices deberían seguir estando conectados. En consecuencia, una componente contendrá uno o más de 3 vértices.

El algoritmo que se describe consiste en identificar y etiquetar todos los vértices y arcos de cada una de las componentes, para lo cual se buscarán ciclos de cualquier orden que conecten sus nudos, etiquetando todos los vértices y arcos del ciclo como pertenencientes a la componente. En el mismo proceso, se identifican y etiquetan como puentes los arcos que no pueden formar parte de ciclos, y los que unen vértices de distintas componentes. Para la generación de los ciclos mencionados se hace uso de una variante del algoritmo 2, generando y analizando en profundidad un árbol con origen en un vértice hasta que se forme un ciclo. El algoritmo hace uso de 5 procedimientos:

— De inicio de componentes, en el que se determina el primer vértice de una nueva componente.

— De avance en profundidad en el árbol con origen en un vértice, en búsqueda de un ciclo.

— De retroceso a lo largo del árbol, cuando se alcanza un punto muerto.

— De anotación de un ciclo y búsqueda del posible origen de un nuevo ciclo.

— De examen de los arcos, una vez etiquetados todos los vértices.

El procedimiento de inicio de componente busca, en primer lugar, el primer vértice no etiquetado, que pertenecerá a una nueva componente. Si no lo encontrase, se pasaría al procedimiento de examen. A continuación, etiqueta el vértice como perteneciente a la nueva componente y examina ordenadamente los arcos con origen en el vértice elegido, viendo si el vértice destino pertenece a una componente anterior o si está sin etiquetar. En el primer caso, el arco será un puente, y se etiqueta como tal. En el segundo, se elige como primer arco del primer ciclo a intentar generar, y se pasa al procedimiento de avance, al que se llegará en el segundo nivel del árbol. Si sólo existiese un arco con origen en el vértice elegido, o no se encontrase ningún arco que pudiese dar origen a un circuito, la componente tendría solamente un vértice, por lo que se volvería al procedimiento de inicio de componente.

El procedimiento de avance en profundidad en el árbol comprueba, en primer lugar, si el vértice del nivel actual coincide con el inicial, en cuyo caso se habría formado un ciclo y se pasaría al procedimiento de anotación del mismo.

A continuación, comprueba si se ha agotado la lista de arcos con origen en el vértice del nivel actual, en cuyo caso se pasaría al procedimiento de retroceso. Se prueba entonces el siguiente arco de la lista y se examina el vértice destino. Si el vértice destino fuese el inicial y el nivel actual fuese el segundo, o si fuese distinto del vértice inicial y ya hubiese sido incluido en la rama en estudio, se volvería al procedimiento de avance. Si se hubiesen superado todos los filtros anteriores, se incluirá el vértice destino en el siguiente nivel, se anotarán como incluidos en la rama tanto el vértice como el arco utilizados para llegar a él, se inicializará la lista de arcos con origen en el nuevo vértice, y se pasará al procedimiento de avance.

El procedimiento de retroceso comprueba, en primer lugar, si se está en el nivel 1, en cuyo caso no

se continuará con el actual vértice inicial, pasando al procedimiento de anotación de ciclo, que busca también el origen del siguiente ciclo a intentar generar. A continuación, si el nivel actual fuese el segundo, anota el primer arco como puente, pues no hay ningún circuito que lo incluya. Seguidamente, borra de las listas de incluidos en la rama actual tanto el vértice del nivel actual como el arco con el que se llegó a él, se retrocede un nivel, y se pasa al procedimiento de avance.

El procedimiento de anotación de ciclo, en primer lugar, etiqueta como pertenecientes a la componente todos los vértices y arcos que constituyen el circuito encontrado, si lo hubiera. A continuación, busca un nuevo inicio de ciclo dentro de la misma componente. Para ello, repasa ordenadamente los vértices ya etiquetados como pertenecientes a la componente, comprobando los arcos con origen en los mismos. Cuando encuentra un arco sin etiquetar, comprueba el vértice destino. Si éste pertenece a la misma componente, se etiqueta el arco como perteneciente a la misma. Si perteneciese a otra componente, se etiquetaría el arco como puente. Por último, si el vértice destino no estuviese etiquetado, se elegirá el vértice origen como vértice inicial del árbol, el arco actual iniciará la lista de arcos utilizados, y se pasará al procedimiento de avance, de nuevo en el nivel 2. Si no fuese posible iniciar ningún árbol de prueba, se habrán localizado todos los vértices, y todos los arcos, de la componente en curso, pasando entonces al procedimiento de inicio de componente.

El procedimiento de examen de los puentes repasa todos los arcos del grafo. Todos los etiquetados como puente, serán puentes, indicando las etiquetas de sus vértices extremos las componentes que une.

La información utilizada por el algoritmo puede almacenarse en los siguientes vectores enteros unidimensionales:

— WAY, que contiene los números de los vértices incluidos en cada nivel de la rama del árbol que se conserva.

— ARWAY, que contiene el número de orden de los arcos incluidos en la rama, con origen en cada nivel.

— LAST, que contiene el número de orden en T del último arco, con origen en cada vértice, incluido en la rama.

— YA, que indica, para cada vértice, si éste está incluido en la rama.

— LANUD, que contiene las etiquetas que indican la componente a que pertenece cada vértice.

— LANAR, que contiene las etiquetas que indican a qué componente pertenece cada arco.

— WAY y ARWAY podrán tener hasta N + 1 elementos, LAST, YA y LANUD tendrán N elementos, y LANAR tendrá W elementos.

6.3. Ejemplo 4

El cuarto ejemplo es el grafo de la Figura 6, que consta de 13 vértices y 13 arcos, que se han elegido por presentar algunos problemas en la ejecución de los algoritmos para el análisis de la conectividad. El primer algoritmo detecta la existencia de tres partes inconexas entre sí, formadas respectivamente por los vértices 1, 2, 3, 4, 5, 6, 7, 8 y 13 la primera; por los vértices 9, 10 y 11 la segunda; y por el vértice 12 la tercera.

El segundo algoritmo detecta la existencia de 7 componentes y de 4 puentes. Las componentes están formadas respectivamente por los siguientes vértices:

Componente 1: 1.
Componente 2: 2, 3 y 4.
Componente 3: 5.
Componente 4: 6, 7 y 8.
Componente 5: 9, 10 y 11.
Componente 6: 12.
Componente 7: 13.

Los puentes unen las componentes 1 y 2, la 2 y la 3, la 3 y la 4, y la 2 y la 7. Las componentes 5 y 6 están aisladas del resto.

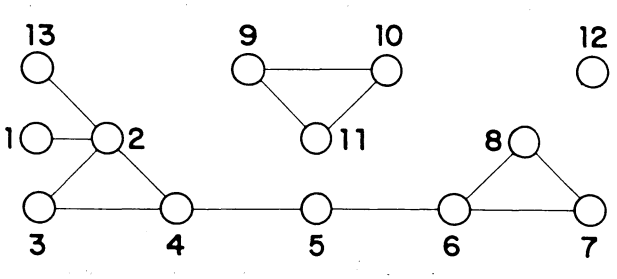

Fig. 6.

7. Conclusiones

Se describen tres algoritmos para la búsqueda de caminos en una red, basados en la generación de un árbol en el que se hace una búsqueda exhaustiva, en amplitud en el primer algoritmo, y en profundidad en el segundo y tercero. El primer algoritmo permite encontrar todos los caminos óptimos entre dos vértices, el segundo permite resolver este mismo problema así como el de hallar los caminos hamiltonianos con origen en un vértice, o los ciclos de cualquier orden, en tanto que el tercero permite buscar todos los caminos o circuitos eulerianos de la red. En los dos primeros algoritmos puede imponerse la condición adicional de evitar que determinados vértices aparezcan en las soluciones.

El primer algoritmo se considera más adecuado para la búsqueda de caminos óptimos entre dos vértices, en tanto que el segundo lo es para la búsqueda de caminos o de circuitos de cualquier orden, incluidos los hamiltonianos.

Se presentan también dos algoritmos que permiten analizar la conectividad de la red, determinando el primero las posibles partes inconexas de la red, y el segundo permite detectar los puentes entre componentes.

Todos los algoritmos minimizan la memoria necesaria, que crece de forma lineal con el orden y tamaño de la red, lo que permite su ejecución total en memoria central incluso para grandes redes. La información se divide en una parte estática, que puede ser generada una vez, almacenada y leída en aplicaciones sucesivas, y una parte dinámica, utilizada durante la ejecución de los algoritmos. Se propone un esquema de almacenamiento para ambos tipos de datos muy compacto y que permite una utilización de los mismos muy efectiva, por los algoritmos descritos.

Minimizan, asimismo, el número de operaciones, pues abortan la generación del árbol en el momento en que se detecta que no conduce a la solución. Algunos problemas pueden resolverse en tiempos reducidos, aun para redes grandes (caminos óptimos, conectividad, puentes), en tanto que otros llevan a una explosión combinatoria (ciclos hamiltonianos, circuitos eulerianos, ciclos largos). En estos últimos, la búsqueda exhaustiva ha de sustituirse por heurísticas adecuadas al problema que permitan obtener alguna solución, tal como la expuesta en el problema del salto del caballo.

Se indica una forma compacta para almacenar esta información, que es manejada con la máxima eficacia por los algoritmos.

Todos los algoritmos pueden ser programados utilizando lenguajes sencillos, QuickBASIC por ejemplo, ya que sólo se usan matrices unidimensionales, sin necesidad de recurrir a punteros. De hecho, para comprobarlos, se ha realizado su programación en dicho lenguaje, utilizando técnicas de programación estructurada.

BIBLIOGRAFÍA

— Knuth, D.: "The art of computer programming", vol. I, Addison Wesley, 1968.

— Gould, R.: "Graph teory", Ed. Cummings P. C., Inc. California, 1980.

— Abellanas: "Teoría de grafos y algoritmos", Domart.

— Elmaghrabi, S.: "Some network models in management science". Sprinter-Verlag, New York, 1970.

— Dolan, A. & Aldous, S.: "Networks and algorithms: An introductory approach". Ed. J. Wiley and Sons, England, 1993.

— Bellmore, M. and G. L. Nemhauser: "The traveling Salesman."

— Harari, F. and Maybe, J.: "Graphs and applications", Wiley, 1985.

— Recuero, A. y Gutiérrez, J. P.: "Reducción del ancho de banda de conjuntos de estructuras independientes", Hormigon y Acero, núm. 173, págs. 43 a 48, 1989.

— Advances in engineering software, en prensa.

— Recuero, A.; Río, O., y Alvarez, M.: "Aplicación de la teoría de grafos a la planificación y programación de proyectos", Informes de la Construcción, vol. 46, núm. 431, 1994, págs. 49-60.

— Recuero, A., y Gutiérrez, J. P.: "Cálculo de flujos en una red multiterminal", Revista de Obras Públicas (en prensa).

— Morán, F.: Carrollia, núm. 42, 1994, págs. 5 y 14.