RIIT Vol.XI. Núm.1. 2010 1-16, ISSN 2594-0732, FI-UNAM
(artículo arbitrado)
DOI: http://dx.doi.org/10.22201/fi.25940732e.2010.11n1.001

La aplicación del álgebra abstracta y las computadoras para la solución de problemas de caminos en redes orientadas

The Application of Abstract Algebra and Computers to the Solution of Path Problems in Oriented Networks

M.A. Murray-Lasso
División de Ingeniería Mecánica e Industrial.
Programa de Posgrado en Ingeniería. Facultad de Ingeniería,
Universidad Nacional Autónoma de México.
E-mail: mamurraylasso@yahoo.com

(Recibido: julio de 2006; reevaluado: enero de 2008; aceptado: agosto de 2009)

Resumen

Se presenta la matriz de conexión de gráficas orientadas y una generalización introducida por Gondran y Minoux para resolver una gran variedad de problemas de caminos, incluyendo diversos problemas de optimización (maximizar o minimizar longitudes, capacidad mínima, probabilidad, etc.), enumeración de caminos, cuenta de caminos, y conexión. Para lograr lo anterior, se tratan a las componentes de las matrices como elementos de una estructura algebraica llamada *semianillo o dioide* (extensión de un monoide). Se explora la posibilidad de utilizar MATLAB en el manejo de matrices y se dan listados de programas cuyo objetivo es educativo y no de producción. Se pretende rescatar un tema que no se ha popularizado debido, en la opinión del autor, a que los originadores Gondran y Minoux (1984) han tratado el tema en forma muy abstracta, orientado a matemáticos y difícil de captar por ingenieros. En este artículo se tratan los temas informalmente y se dan ejemplos ilustrativos (cosa que Gondran y Minoux, no proveen en gran detalle), así como listados de programas en el lenguaje de MATLAB. El tema se presta para seguirlo extendiendo y diseñar proyectos educativos computarizados para el aprendizaje de temas importantes de redes cuyas aplicaciones son muy extensas.

Descriptores: álgebra abstracta, matriz de conexión, MATLAB, caminos en redes, optimización en redes.

Abstract

The connection matrix of oriented graphs and a generalization introduced by Gondran and Minoux to solve a great variety of path problems, including various optimization problems (maximize or minimize lengths, minimum capacity, probability, etc.), ennumeration of paths, path counting, and connection. To achieve this the matrix components are treated as elements of an algebraic structure called semiring or diod (an extension of a monoid.) The possibilities of using MATLAB for handling the matrices are explored and listings of educational programs (not for production runs) are provided. The purpose is to rescue a topic which has not become very popular due, in the authors opinion, to the fact that the originators Gondran and Minoux (Ref. 3) have treated the topic in a

DOI: http://dx.doi.org/10.22201/fi.25940732e.2010.11n1.001

very abstract manner, oriented to mathematicians and difficult to grasp by engineers. In this article the topics are treated informally and illustrative examples are given (something that Ref. 3 does not provide) in great detail as well as listings in the MATLAB language. The topic is ammenable to extensions and it is possible to design educational computerized projects for learning important network topics with very wide applications.

Keywords: *Abstract algebra, connection matrix, MATLAB, paths in networks, network optimization.*

Introducción

En la solución de ciertos problemas de redes que tienen que ver con caminos, la matriz de conexión generalizada juega un papel central. Si los problemas se plantean adecuadamente, algunos algoritmos clásicos como los de Jacobi y Gauss-Seidel para resolver sistemas lineales de ecuaciones por iteración, pueden utilizarse para resolver una variedad muy grande de problemas de caminos en redes. Los algoritmos resultantes son fáciles de automatizar en lenguajes que manejan arreglos.

La matriz de conexión y su generalización

Sea G una gráfica orientada con n nodos, cada par (i, j) de los cuales están conectados en un sentido por, a lo más, una sola rama, a la cual se le da un peso c_{ij}. Se admiten conexiones entre un nodo y sí mismo. (Los pesos son números reales que se pueden interpretar como longitudes, capacidades, confiabilidades, etcétera). Se define la *matriz de conexión* $\boldsymbol{C}$ (también llamada *matriz asociada con la gráfica* o *matriz de adyacencia*) como la matriz $n \times n$ que tiene como componentes c_{ij}, $i = 1, 2, \ldots, n$. (Bellman *et al*., 1970 y Mayeda, 1972). Cuando no existe conexión entre un par de nodos (o de un nodo con sí mismo) se pone un peso de cero. (Estamos pensando en pesos interpretados como capacidades; en caso de que fueran longitudes, se pondrían infinitos que como veremos más adelante juegan un papel similar al cero para las operaciones que se utilizarán al generalizar el concepto de matriz de conexión.)

Por ejemplo, para la gráfica de la figura 1 con los pesos indicados junto a las ramas se tendría la matriz $\boldsymbol{C}$ que se muestra en la figura 1.

Una primera generalización de la matriz de conexión $\boldsymbol{C}$ se puede hacer admitiendo más de una rama en paralelo en la misma dirección entre dos nodos o de un nodo con sí mismo. En este caso, simplemente se suman los pesos de las ramas en paralelo. (De nuevo estamos pensando en la interpretación de capacidad para los pesos, con otras interpretaciones se aplican otras operaciones.) Para la gráfica de la figura 2 su matriz de conexión generalizada es la que se muestra en la propia figura.

$$\boldsymbol{C} = \begin{bmatrix} a & b & 0 & c & 0 \\ 0 & 0 & d & e & 0 \\ 0 & 0 & 0 & f & 0 \\ 0 & 0 & g & 0 & 0 \\ 0 & 0 & h & 0 & 0 \end{bmatrix}$$

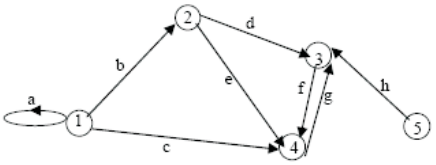


Figura 1

$$\boldsymbol{C} = \begin{bmatrix} a_1+a_2 & b & 0 & c & 0 \\ 0 & 0 & d_1+d_2+d_3 & e & 0 \\ 0 & 0 & 0 & f_1+f_2 & 0 \\ 0 & 0 & g & 0 & 0 \\ 0 & 0 & h_1+h_2 & 0 & 0 \end{bmatrix}$$

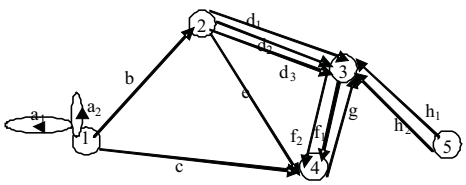


Figura 2

DOI: http://dx.doi.org/10.22201/fi.25940732e.2010.11n1.001

Una generalización adicional que se le puede hacer a la matriz de conexión, consiste en que, en vez que los pesos de las componentes sean números reales, se les suponga elementos de un conjunto más amplio que permite la posibilidad de ser manipulado con operaciones como max, min, ∪ (unión), © (concatenación), y otros. Para muchos de los problemas que nos interesan, basta que los elementos sean miembros de un *semianillo*. (Gondran y Minoux (1984) le llaman una *dioide*) [Un semianillo es un conjunto $\boldsymbol{S}$ de elementos y dos operaciones ⊕ y ⊗. La operación ⊕ a la cual llamaremos *suma* genera en $\boldsymbol{S}$ la estructura de un *monoide conmutativo* del cual llamaremos ε al elemento neutro *cero.*

La operación ⊗, a la cual llamaremos *multiplicación,* genera en $\boldsymbol{S}$ la estructura de un *monoide* (no necesariamente conmutativo) y tiene un elemento neutro e, llamado *unidad*. La ley distributiva de la multiplicación con respecto a la suma se cumple de los dos lados. (MacLane *et al*., 1967 y Gondran y Minoux, 1984)]. Por ejemplo, el sistema algebraico ($\underline{\boldsymbol{R}}$, **min**, +) es un semianillo, donde $\underline{\boldsymbol{R}}$ es el conjunto de los reales complementados con ∞, *cero* → ∞ (un número especial mayor que cualquier real), y la *unidad* → 0 (el cero de los reales). La operación + es la suma entre reales y **min** es la función de dos variables que da como valor la que está más a la izquierda en la recta numérica.

Propiedad fundamental de la matriz de conexión. Cuenta de caminos

La matriz de conexión original de una gráfica orientada con pesos en las ramas, obtiene su utilidad de la siguiente propiedad: el cuadrado de la matriz de conexión tiene como elemento *(i, j)* la suma de los productos de los pesos de las ramas que forman caminos con dos tramos que parten del nodo i y llegan al nodo j. La propiedad proviene de la manera en que se define el producto de dos matrices reales $n \times n$; así por ejemplo, si $\boldsymbol{D} = \boldsymbol{C}^2$, la matriz $\boldsymbol{D}$ tiene componentes

$$d_{ij} = \sum_{k=1}^{n} c_{ik} c_{kj} = c_{i1} c_{1j} + c_{i2}\, c_{2j} \quad c_{i3} c_{3j} + \ldots + c_{in} c_{nj.}$$

Los caminos de dos tramos que parten del nodo i y llegan al nodo j, necesariamente tienen su primera rama *(i,k)* saliendo de i y yendo a un nodo k (el cual podría ser el mismo i) y luego su segunda rama parte del nodo k y llega al nodo j.

Como en la sumatoria se da a k todos los posibles valores, aparecen todos los posibles caminos con dos tramos, excepto aquellos que en uno de los dos tramos tiene peso cero, es decir, que no existe el tramo. Si calculamos el cuadrado de la matriz $\boldsymbol{C}$ de la figura 1 se obtiene:

$$\boldsymbol{D} = \boldsymbol{C}^2 = \begin{bmatrix} a & b & 0 & c & 0 \\ 0 & 0 & d & e & 0 \\ 0 & 0 & 0 & f & 0 \\ 0 & 0 & g & 0 & 0 \\ 0 & 0 & h & 0 & 0 \end{bmatrix} \times \begin{bmatrix} a & b & 0 & c & 0 \\ 0 & 0 & d & e & 0 \\ 0 & 0 & 0 & f & 0 \\ 0 & 0 & g & 0 & 0 \\ 0 & 0 & h & 0 & 0 \end{bmatrix} =$$

$$\begin{bmatrix} a^2 & ab & bd+cg & ac+be & 0 \\ 0 & 0 & eg & df & 0 \\ 0 & 0 & fg & 0 & 0 \\ 0 & 0 & 0 & gf & 0 \\ 0 & 0 & 0 & hf & 0 \end{bmatrix}.$$

El lector puede verificar la aseveración examinando la figura 1.

Si se desea la suma de los productos de los pesos de las ramas de todos los caminos con tres ramas que parten del nodo i y llegan al nodo j, basta examinar la componente *(i, j)* del cubo $\boldsymbol{E} = \boldsymbol{C}^3$ de la matriz de conexión $\boldsymbol{C}$.

$$\boldsymbol{E} = \boldsymbol{C}^2\boldsymbol{C} = \begin{bmatrix} a^2 & ab & bd+cg & ac+be & 0 \\ 0 & 0 & eg & df & 0 \\ 0 & 0 & fg & 0 & 0 \\ 0 & 0 & 0 & gf & 0 \\ 0 & 0 & 0 & hf & 0 \end{bmatrix} \times$$

$$\begin{bmatrix} a & b & 0 & c & 0 \\ 0 & 0 & d & e & 0 \\ 0 & 0 & 0 & f & 0 \\ 0 & 0 & g & 0 & 0 \\ 0 & 0 & h & 0 & 0 \end{bmatrix} =$$

$$\begin{bmatrix} a^3 & a^2 b & abd+acg+beg & a^2 c+abe+bdf+cfg & 0 \\ 0 & 0 & dfg & egf & 0 \\ 0 & 0 & 0 & fgf & 0 \\ 0 & 0 & gfg & 0 & 0 \\ 0 & 0 & hfg & 0 & 0 \end{bmatrix}.$$

Debido a que el componente e_{ij} del cubo de $\boldsymbol{C}$ combina todos los caminos de dos tramos (contenidos en $\boldsymbol{C}^2$) que parten del nodo i y llegan a cualquier nodo k con los caminos de un tramo (contenidos en $\boldsymbol{C}$) que van del nodo k al nodo j, dicha componente es igual a la suma de los productos de los pesos de todos los caminos de

DOI: http://dx.doi.org/10.22201/fi.25940732e.2010.11n1.001

tres ramas que parten del nodo *i* y llegan al nodo *j*. Lo anterior se puede verificar para la gráfica de la figura 1 examinando la matriz ***E.*** Notamos que algunos de los caminos repiten algunas de las ramas y en el caso del lazo que conecta al nodo *1* consigo mismo, hasta 3 veces.

El problema de enumeración de caminos

Si se hubieran dado valores numéricos a las letras *a, b, c, ..., h* quizás no hubiéramos notado que mientras tengamos cuidado de mostrar los productos en el orden que aparecen los factores, las secuencias nos identifican los caminos. Así, por ejemplo, en la componente *(2, 3)* de ***E***, *dfg* denota el camino de tres tramos que parte del nodo 2 y llega al nodo 3 y recorre las ramas *d, f* y *g*. Podemos entonces pensar en *d, f* y *g,* no como números reales, sino como símbolos y *dfg* como una concatenación de símbolos. En el componente *(1,3)* a la expresión *abd* + *acg* + *beg* la podemos interpretar como la unión de los tres caminos de 3 tramos *abd, acg* y *beg.* Concluimos que si en vez de multiplicación y suma usamos concatenación y unión, podemos enumerar los caminos a través de las potencias de la matriz de conexión. En vez del número real 0, lo que corresponde en este caso (para la operación unión) sería Φ, el *conjunto vacío.* Notamos que a la propiedad de los números reales *0. R = 0* corresponde la propiedad $\Phi \copyright c = \Phi$, donde *c* es cualquier cadena de símbolos y © es la operación de concatenación que corresponde a lo que más abstractamente le llamamos la operación de multiplicación ⊗. La interpretación de esta propiedad es que cualquier cadena que se encuentre en un camino con un hueco (ausencia de camino) tiene el efecto de eliminar el camino. También notamos que con respecto a la operación de unión el conjunto vacío de símbolos es neutro, es decir, $\Phi \cup T = T$ donde *T* es un conjunto de cadenas de símbolos. Con respecto a la operación de concatenación el elemento neutro es la cadena vacía Θ, pues se tiene que para cualquier cadena *c*

$\Theta \copyright c = \copyright \Theta = c$.

Regresando a la primera generalización de la matriz de conexión con pesos para las ramas, los cuales representan números de caminos directamente conectados entre 2 nodos, el peso total de un camino que parte del nodo *i* y llega al nodo *j*, representa el número total de caminos diferentes que conectan los nodos *i* e *j*. Por ejemplo, si suponemos en la figura 2 que todas las letras, con o sin índices valen 1, las matrices ***C***, $\boldsymbol{C}^2$ y $\boldsymbol{C}^3$ valdrían

$$\boldsymbol{C} = \begin{bmatrix} 2 & 1 & 0 & 1 & 0 \\ 0 & 0 & 3 & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{D} = \boldsymbol{C}^2 = \begin{bmatrix} 4 & 2 & 4 & 3 & 0 \\ 0 & 0 & 1 & 6 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 4 & 0 \end{bmatrix},$$

$$\boldsymbol{E} = \boldsymbol{C}^3 = \begin{bmatrix} 8 & 4 & 9 & 14 & 0 \\ 0 & 0 & 6 & 2 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \end{bmatrix}.$$

Examinemos la componente $e_{23} = 6$. Su valor indica que hay 6 diferentes caminos de tres tramos en la gráfica de la figura 2 que parten del nodo 2 y llegan al nodo 3. Dichos caminos son (listando las secuencias de aristas): a_1bd_1, a_1bd_2, a_1bd_3, a_2bd_1, a_2bd_2, a_2bd_3.

El lector puede examinar las demás componentes de las matrices ***D*** y ***E*** para revisar los caminos de dos y tres tramos.

Si se hace una correspondencia entre las operaciones que se utilizan para la cuenta de caminos y las que se utilizan para enumerar caminos por medio de concatenación obtenemos lo que se muestra en la tabla 1.

Tabla 1

Conjunto	Problema	Primera Operación	Segunda Operación	Neutro Primera Operación	Neutro Segunda Operación
Números enteros	Cuenta de caminos	+ (suma de enteros)	× (multiplicación de enteros)	0	1
Conjuntos de cadenas de símbolos	Enumeración de caminos	∪ (unión de conjuntos de cadenas de símbolos)	© (concatenación de cadenas de símbolos)	Φ (conjunto vacío de cadenas de símbolos)	Θ (cadena de símbolos vacía)

DOI: http://dx.doi.org/10.22201/fi.25940732e.2010.11n1.001

Enumeración de caminos elementales

Cuando se representan las ramas de una gráfica por medio de una letra diferente para cada una y se obtienen simbólicamente las potencias de la matriz de conexión, aparecen todas las secuencias de las aristas que forman caminos, ya sea que dichos caminos se cierren e incluyan ciclos. Un camino contiene ciclos si aparece un nodo más de una vez. Es difícil detectar si una secuencia de artistas contiene un nodo repetido. Para facilitar la detección resulta conveniente describir los caminos por medio de sucesiones de nodos tales que un nodo repetido se pueda detectar fácilmente. Si adoptamos esta idea, una rama que vaya del nodo i al nodo j queda representada por el par ij.

Si la siguiente arista va del nodo j al nodo k el camino con las dos ramas queda descrito por el trío de nodos ijk. Notamos que la operación para construir las secuencias de nodos resulta ser una concatenación modificada pues en vez de poner $ijjk$, una de las j's se elimina. Adicionalmente se debe revisar que, excepto por el último nodo de la primera cadena y el primer nodo de la segunda cadena, los demás nodos son todos distintos si los caminos son elementales. A este proceso de eliminar uno de los nodos extremos y evitar nodos repetidos se le llama Multiplicación Latina (Kaufman *et al*., 1963).

Concluimos que para encontrar caminos elementales en una gráfica con n nodos con diferentes números de tramos, hay que encontrar la suma de las n potencias de la matriz de conexión $\boldsymbol{C}^0$, $\boldsymbol{C}^1$, $\boldsymbol{C}^2$, . . . , $\boldsymbol{C}^{n-1}$ (como en el caso de la enumeración de caminos sin restrricción) solamente que para la segunda operación se usa la multiplicación latina en vez de la simple concatenación. El elemento neutro de la segunda operación ⊗ (multiplicación latina) es el conjunto N, cuyos elementos son los nodos de la red, ya que bajo la multiplicación latina se tiene que si c es una secuencia de nodos representando un camino

$$N \otimes c = c \otimes N = c.$$

El problema de la ruta más corta

Regresemos ahora a la figura 2 e interpretemos las letras junto a las aristas como longitudes de las ramas. Nos interesa encontrar la ruta más corta entre los nodos 1 y 3. Dado que todos los ciclos de la gráfica tienen longitud positiva, a ninguna ruta más corta entre dos nodos le conviene incluir un ciclo. De esta forma, la ruta más corta a lo mucho, pasará por todos los cinco nodos y por lo tanto, tendrá 4 tramos. Podríamos encontrar todos los caminos de 4 tramos entre los nodos 1 y 3. Esto lo podríamos hacer encontrando la cuarta potencia de su matriz de conexión. El hacerlo simbólicamente y a mano es laborioso; sin embargo, con un poco de reflexión nos podemos ahorrar trabajo. En vez de usar la matriz de conexión de la figura 2, entre cada par de nodos podemos dejar solamente la rama más corta, a la cual nombramos con la letra correspondiente sin índice. Es así que volvemos a la matriz de conexión de la figura 1 que es más simple.

Al elevar la matriz de conexión al cuadrado podemos hacer de nuevo el razonamiento que debemos dejar de entre todos los caminos de dos tramos entre un par de nodos el más corto. También podemos ignorar todos los caminos que incluyen ciclos. Estos razonamientos se pueden extender a las subsecuentes potencias de la matriz de conexión hasta la cuarta que es la que requerimos.

Por otra parte, aunque hemos usado el término potencia, la cual tiene que ver con multiplicaciones, que es el caso para cuenta de caminos, ya que los números de caminos diferentes se obtienen multiplicando los números de alternativas entre cada par de nodos en una secuencia de nodos que definen un camino, para el caso de longitudes de caminos dichas longitudes se suman en vez de multiplicarse y como ya hemos visto entre cada par de nodos se toma la longitud mínima. Con esto hemos establecido las correspondencias que se muestran en la tabla 2.

Tabla 2

Conjunto	Problema	Primera Operación	Segunda Operación	Neutro Primera Operación	Neutro Segunda Operación
Números reales	Ruta más corta	Min	+	∞	0

DOI: http://dx.doi.org/10.22201/fi.25940732e.2010.11n1.001

La razón por la cual ∞ es el elemento neutro de la operación *Min* es que para cualquier número real r se tiene Min (r, ∞) = r que es precisamente la característica que define al elemento neutro, ya que, por ejemplo,

$1 \times k = k \times 1 = k$	(para cualquier entero k).
$0 + k = k + 0 = k$	(para cualquier entero k).
$0 + r = r + 0 = r$	(para cualquier real r).
$\Phi \cup T = T \cup \Phi = T$	(para cualquier conjunto de cadenas de símbolos T).
$\Theta © c = c © \Theta = c$	(para cualquier cadena c de símbolos).

La única operación de las introducidas que no es conmutativa es la concatenación que se usa como segunda operación; sin embargo, todas las primeras operaciones son conmutativas y todas las segundas operaciones distribuyen de ambos lados con las primeras operaciones y todas, primeras y segundas tienen un elemento neutro por ambos lados.

Las operaciones que se corresponden en las tablas 1 y 2 son tan similares que frecuentemente se utilizan los mismos símbolos (en lenguajes de computación, por ejemplo) para denotar dichas operaciones. En BASIC, se utiliza + para concatenación de cadenas de símbolos. En algunos libros de matemáticas se utiliza el símbolo + para la unión de conjuntos.

En algunos libros se utiliza un cero tachado (que se parece mucho a la letra griega *fi*) para denotar el conjunto vacío.

Para clarificar ideas démosle a las letras que denotan los pesos en la figura 2 diversos valores que representan longitudes como sigue:

$a_1=1$, $a_2=2$, $b=3$, $c=4$, $d_1=5$, $d_2=6$, $d_3=7$, e=7.5,$f_1=8$, $f_2=4.5$, $g=2.4$, $h_1=9.3$, $h_2=8.6$.

Aprovechando el trabajo algebraico realizado, podemos ir directamente a la matriz $\boldsymbol{F = C^4}$, sustituir en las expresiones algebráicas los valores numéricos sumando en vez de multiplicar y aplicar en vez del operador suma el operador *Min*. Comenzamos por exhibir la matriz $\boldsymbol{C^4}$ en la tabla 3.

Ante todo, podemos ignorar todos los términos en los que aparece a. También podemos ignorar todos los términos en que un símbolo aparece repetido, pues esto indica un ciclo.

Las distancias más cortas de un nodo a si mismo son cero en todos los casos, por lo que a la matriz final de distancias más cortas (tabla 4) le ponemos ceros en la diagonal principal. Tenemos que comparar las longitudes de un tramo, dos tramos, tres tramos y cuatro tramos, por lo que habrá que examinar las matrices $\boldsymbol{C}$, $\boldsymbol{C^2}$, $\boldsymbol{C^3}$ y $\boldsymbol{C^4}$.

Tabla 3

a^4	a^3b	$a^2bd + a^2cg + abeg + bdfg + cgfg$	$a^3c + a^2be + abdf$ $acgf + begf$	0
0	0	$egfg$	$dfgf$	0
0	0	$fgfg$	0	0
0	0	0	$gfgf$	0
0	0	0	hfhf	0

Tabla 4

0	2	6.4	4	∞
∞	0	5	7.5	∞
∞	∞	0	4.5	∞
∞	∞	2.4	0	∞
∞	∞	8.6	13.1	0

DOI: http://dx.doi.org/10.22201/fi.25940732e.2010.11n1.001

Comentarios a la tabla 4: No hay modo de llegar legalmente (sin irse en sentido contrario) entre los pares de nodos que tienen ∞. Por ejemplo, la única manera de ir al nodo 5 o al nodo 1 es comenzar en dichos nodos. Verificando sobre la figura 2 tenemos las siguientes rutas:

Ruta 1-2. Directo de 1 a 2, longitud b = 3.
Ruta 1-3. Dos tramos. 1 a 4 a 3, c + g = 4 + 2.4 = 6.4
Ruta 1-4. Directo de 1 a 4, longitud d = 5
Ruta 2-3. Directo de 2 a 3, longitud c = 4
Ruta 2-4. Directo de 2 a 4, longitud e = 7.5
Ruta 3-4. Directo de 3 a 4, longitud f = 4.5
Ruta 4-3. Directo de 4 a 3, longitud g = 2.4
Ruta 5-3. Directo de 5 a 3, longitud h = 8.6
Ruta 5-4. Dos tramos 5 a 3 a 4, h + f = 8.6 + 4.5 = 13.1

El calcular las expresiones algebraicas de $\boldsymbol{C}$, $\boldsymbol{C}^2$, $\boldsymbol{C}^3$, y $\boldsymbol{C}^4$ para encontrar todos los caminos de 1, 2, 3 y 4 tramos, luego sustituir valores numéricos y encontrar el valor mínimo para encontrar las longitudes de los caminos más cortos, aunque correcto, es un proceso largo e ineficiente. El proceso se parece al cálculo del determinante de una matriz haciendo la expansión en productos, tomando factores de cada fila y cada columna y afectándolos de un signo que depende de la paridad de la permutación de los índices. El proceso de comenzar dando a los elementos sus valores numéricos y de triangularizar el arreglo por medio de operaciones elementales de las filas, que dejan invariante el valor del determinante y una vez triangularizado calcular el producto de los elementos en la diagonal principal (Método de Gauss) es mucho más eficiente, debido a que va combinando elementos y encontrando sus valores numéricos durante el proceso, evitando la explosión de términos que produce la ley distributiva de la multiplicación respecto a la suma.

En forma análoga, si al calcular las potencias de la matriz de conexión, primero se les dan a los elementos sus valores numéricos y se va aplicando el operador *Min* en el camino, se evita la explosión de términos, eficientando en forma muy importante el trabajo. Un primer paso en este proceso, lo dimos en la matriz $\boldsymbol{C}$ de la figura 2, conservando de las ramas paralelas sólo la que tiene longitud mínima. Invitamos al lector a sentir la explosión de términos encontrando las matrices $\boldsymbol{C}^2$, $\boldsymbol{C}^3$ y $\boldsymbol{C}^4$ y percatarse de la enorme simplificación al haber eliminado las ramas paralelas de la figura 2, y haberla dejado con la estructura de la figura 1. Hacer simplificaciones adicionales en el camino antes de calcular las demás potencias reduce aún más el trabajo.

Lo anterior logra la determinación de las distancias más cortas entre todos los pares de puntos de una red. No siempre interesa todos los nodos contra todos. Frecuentemente se requiere la distancia más corta de un origen a todos los puntos de la red, o de todos los puntos de la red a un destino. Esto se puede obtener calculando una sola columna o una sola fila de la matriz final. Para calcular una sola columna de la matriz final se "premultiplica" (estamos usando la palabra "multiplicar" en el sentido ampliado que en nuestro caso es sumar; la multiplicación de matrices por vectores involucra también "sumas", es decir, en nuestro caso la aplicación del operador *Min*) n – 2 veces la matriz $\boldsymbol{C}$ por la columna en cuestión.

Es interesante hacer notar que esta operación equivale al algoritmo de Programación Dinámica de Bellman *et al*. (1962 y 1970) y para encontrar las rutas más cortas de todos los nodos a un destino en una red. Si se desea obtener solamente una fila de la matriz final, entonces dicha fila se "postmultiplica" n – 2 veces por la matriz $\boldsymbol{C}$. Es instructivo resolver un ejemplo numérico para clarificar ideas, por lo tanto, calcularemos las distancias más cortas desde el nodo 7 a todos los nodos de la red de la figura 3 (Hu, 1969).

En la red de la figura 3 todas las ramas, excepto tres, admiten circulación en ambos sentidos. Dichas ramas pueden ser sustituidas por pares de ramas con sentidos opuestos, cada una con longitud igual a la marcada. Aunque no hemos dibujado los pares de ramas, están consideradas en la matriz de conexión $\boldsymbol{C}$. Como tomamos como origen el nodo 7, vamos a "postmultiplicar" la séptima fila de $\boldsymbol{C}$ repetidamente por $\boldsymbol{C}$. Mostramos el trabajo en gran detalle para las primeras dos "multiplicaciones."

El vector original es $\boldsymbol{C}_7$ = [∞,∞,∞, 20, 1, 1, 0]. Las componentes de la séptima fila de $\boldsymbol{C}^2$ son:

$\boldsymbol{C}^2_{71}$ = Min [∞ + 0, ∞ + 11, ∞ + 30, 20 + ∞, 1 + ∞, 1 +∞, 0 + ∞] = ∞
$\boldsymbol{C}^2_{72}$ = Min [∞ + 11, ∞ + 0, ∞ + ∞, 20 + 12, 1 + 2, 1 + ∞, 0 + ∞] = 3
$\boldsymbol{C}^2_{73}$ = Min [∞ + 30, ∞ + ∞, ∞ + 0, 20 + 19, 1 + ∞, 1 + 4, 0 + ∞] = 5
$\boldsymbol{C}^2_{74}$ = Min [∞ + ∞, ∞ + 12, ∞ + 19, 20 + 0, 1 + 11, 1 + 9, 0 + 20] = 10
$\boldsymbol{C}^2_{75}$ = Min [∞ + ∞, ∞ + 2, ∞ + ∞, 20 + 11, 1 + 0, 1 + ∞, 0 + 1] = 1
$\boldsymbol{C}^2_{76}$ = Min [∞ + ∞, ∞ + ∞, ∞ + 4, 20 + 9, 1 + ∞, 1 + 0, 0 + 1] = 1
$\boldsymbol{C}^2_{77}$ = Min [∞ + ∞, ∞ + ∞, ∞ + ∞, 20 + ∞, 1 + ∞, 1 + ∞, 0 + 0] = 0

DOI: http://dx.doi.org/10.22201/fi.25940732e.2010.11n1.001

$C =$

0	11	30	∞	∞	∞	∞
11	0	∞	12	2	∞	∞
30	∞	0	19	∞	4	∞
∞	12	19	0	11	9	∞
∞	2	∞	11	0	∞	∞
∞	∞	4	9	∞	0	∞
∞	∞	∞	20	1	1	0

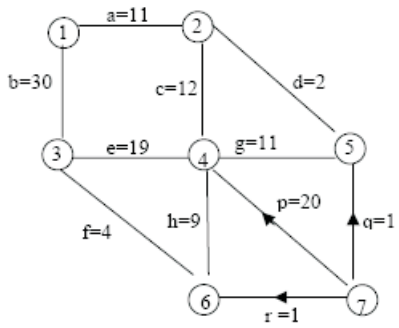


Figura 3

El vector resultante (séptima fila de $\boldsymbol{C}^2$) es $\boldsymbol{C}^2{}_7 = [\infty, 3, 5, 10, 1, 1, 0]$.

Al "sumar" el vector $[\infty, \infty, \infty, \infty, \infty, \infty, 0]$ (séptimo vector unitario) con el vector $[\infty,\infty,\infty, 20, 1, 1, 0]$ y con el vector recientemente calculado $[\infty, 3, 5, 10, 1, 1, 0]$ obtenemos el vector $[\infty, 3, 5, 10, 1, 1, 0]$. (Recordar que "sumar", en este caso, significa tomar el mínimo). A $\boldsymbol{C}^2{}_7$ se "postmultiplica" por $\boldsymbol{C}$ y se obtienen las siguientes componentes de la séptima fila de $\boldsymbol{C}^3$.

$\boldsymbol{C}^3{}_{71} =$ Min $[\infty + 0, 3 + 11, 5 + 30, 10 + \infty, 1 + \infty, 1 + \infty, 0 + \infty] = 14$
$\boldsymbol{C}^3{}_{72} =$ Min $[\infty + 11, 3 + 0, 5 + \infty, 10 + 12, 1 + 2, 1 + \infty, 0 + \infty] = 3$
$\boldsymbol{C}^3{}_{73} =$ Min $[\infty + 30, 3 + \infty, 5 + 0, 10 + 19, 1 + \infty, 1 + 4, 0 + \infty] = 5$
$\boldsymbol{C}^3{}_{74} =$ Min $[\infty + \infty, 3 + 12, 5 + 19, 10 + 0, 1 + 11, 1 + 9, 0 + 20] = 10$
$\boldsymbol{C}^3{}_{75} =$ Min $[\infty + \infty, 3 + 2, 5 + \infty, 10 + 11, 1 + 0, 1 + \infty, 0 + 1] = 1$
$\boldsymbol{C}^3{}_{76} =$ Min $[\infty + \infty, 3 + \infty, 5 + 4, 10 + 9, 1 + \infty, 1 + 0, 0 + 1] = 1$
$\boldsymbol{C}^3{}_{77} =$ Min $[\infty + \infty, 3 + \infty, 5 + \infty, 10 + \infty, 1 + \infty, 1 + \infty, 0 + 0] = 0$

El vector resultante (séptima fila de $\boldsymbol{C}^3$) es $\boldsymbol{C}^3{}_7 = [14, 3, 5, 10, 1, 1, 0]$.

Al sumar el vector obtenido en el anterior paso, con éste obtenemos el mismo vector. Sin mostrar el trabajo de las siguientes iteraciones se tiene que $\boldsymbol{C}^4{}_7 = \boldsymbol{C}^5{}_7 = \boldsymbol{C}^6{}_7 = [14, 3, 5, 10, 1, 1, 0]$ y al sumarlos con el vector que resultó del trabajo previo se obtiene el mismo vector. Cuando un vector se repite, como se multiplica de nuevo por la misma matriz, el vector se sigue repitiendo de allí en adelante, por lo que el trabajo se puede suspender. Las componentes del vector que se repite nos dan las distancias más cortas del nodo 7 a cada uno de los nodos, es decir,

De 7 a 1: distancia = 14 (nodos 7, 5, 2, 1).
De 7 a 2: distancia = 3 (nodos 7, 5, 2).
De 7 a 3: distancia = 5 (nodos (7, 6, 3).
De 7 a 4: distancia = 10 (nodos 7, 6, 4).
De 7 a 5: distancia = 1 (nodos 7, 5).
De 7 a 6: distancia = 1 (nodos 7, 6).
De 7 a 7: distancia = 0 (nodos 7).

Recapitulando, hemos calculado para la red de la figura 3, la séptima fila de la matriz $\boldsymbol{C}^0 \oplus \boldsymbol{C}^1 \oplus \boldsymbol{C}^2 \oplus \boldsymbol{C}^3 \oplus \boldsymbol{C}^4 \oplus \boldsymbol{C}^5 \oplus \boldsymbol{C}^6$, la cual nos da la distancia del nodo 7 a todos los nodos de la red. En esta operación la "suma" $\oplus$ es el operador *Min* y la operación "multiplicación" que se utiliza para "multiplicar" las matrices por sí mismas para calcular las potencias, es la operación "+", entre números reales.

En la "multiplicación" de las matrices también entran "sumas", las cuales también son el operador *Min*. Debido a que las potencias de las matrices se repiten, no hubo necesidad de calcular más allá de $\boldsymbol{C}^3$. La matriz $\boldsymbol{C}^0$ es la matriz "unidad," la cual tiene "unos" en la diagonal principal y "ceros" en las demás componentes. Aquí los "unos" son los ceros de los reales y los "ceros" son ∞ (un

DOI: http://dx.doi.org/10.22201/fi.25940732e.2010.11n1.001

número especial que es mayor que cualquier número real).

Intervención de la computadora. Consideraciones de eficiencia computacional

Una de las principales ventajas de la matriz de conexión, además de permitir manejar ideas con una notación simplificada con pocos símbolos, es que facilita mucho la automatización de la solución de diversos problemas de caminos en redes. Basta automatizar la secuencia de operaciones correspondientes a la suma y multiplicación de matrices. El haber generalizado la matriz de conexión, nos permite hacer la automatización una sola vez y usar el mismo programa para diversos problemas, cambiando solamente la matriz inicial y los operadores que se utilizan para las dos operaciones de "suma" $\oplus$ y "multiplicación" $\otimes$. Así como sus dos elementos neutros "cero"ε y "uno" e. Algunos de los problemas se resuelven buscando $\boldsymbol{C}^{n-1}$ mientras que otros se resuelven buscando $\boldsymbol{C}^0 \oplus \boldsymbol{C}^1 \oplus \boldsymbol{C}^2 \oplus \ldots \boldsymbol{C}^{n-1}$. Por ejemplo, el problema de la determinación de caminos Hamiltonianos requiere la enésima potencia de la matriz de conexión. La enumeración de caminos requiere la "suma" de las n potencias (de la 0 a la $n-1$) de la matriz de conexión generalizada.

Con frecuencia, no es necesario llegar hasta la $n-1$ potencia, pues en muchos casos para alguna potencia k menor que $n-1$ se tiene que las subsecuentes potencias son iguales (a las matrices se le llama entonces *idempotentes*). Asimismo, es posible ahorrar trabajo computacional expresando el número k en notación binaria y notando que sólo se requieren ciertas potencias de 2 de la matriz de conexión. Éstas se pueden calcular multiplicando algunas potencias menores. Por ejemplo, si se requiere $\boldsymbol{C}^{33}$, se nota que $(33)_{10} = (100001)_2$, el cual se puede calcular $2^5 + 2^0$, es decir,

$$\boldsymbol{C}^{33} = ((((((\boldsymbol{C}\otimes\boldsymbol{C})\otimes\boldsymbol{C}^2)\otimes\boldsymbol{C}^4)\otimes\boldsymbol{C}^8)\otimes\boldsymbol{C}^{16})\otimes\boldsymbol{C})$$

$\boldsymbol{C}^2$

$\boldsymbol{C}^4$

$\boldsymbol{C}^8$

$\boldsymbol{C}^{16}$

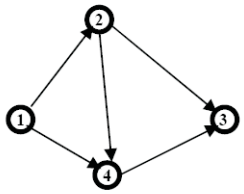
En la anterior operación sólo se realizan explícitamente las multiplicaciones señaladas con $\otimes$ es decir, 6 multiplicaciones, ya que del resultado de multiplicaciones previas ya se tienen calculadas las potencias necesarias. Para ser más explícitos: se multiplica $\boldsymbol{C}$ por $\boldsymbol{C}$ para obtener $\boldsymbol{C}^2$, ésta a su vez se multiplica por si misma para obtener $\boldsymbol{C}^4$; la cual se multiplica por si misma para obtener $\boldsymbol{C}^8$; la cual se multiplica por si misma para obtener $\boldsymbol{C}^{16}$; la cual se multiplica por si misma para obtener $\boldsymbol{C}^{32}$; la cual se multiplica por $\boldsymbol{C}$ para por fin obtener $\boldsymbol{C}^{33}$. (6 multiplicaciones en total.)

Finalmente, no siempre interesa la matriz completa, en ocasiones sólo interesa trabajar algunas filas o algunas columnas o incluso, sólo algunas componentes sueltas de la matriz resultante. A este respecto es conveniente recordar que el número de operaciones depende de cómo se aplique la ley asociativa de la multiplicación de matrices. Por ejemplo, si $\boldsymbol{x}$ es un vector columna con n componentes (que podría ser una columna de $\boldsymbol{C}$) y $\boldsymbol{C}$ es una matriz $n\times n$, y se desea calcular $\boldsymbol{C}^2\cdot\mathrm{x} = \boldsymbol{C}\cdot\boldsymbol{C}\cdot\boldsymbol{x}$. Si primero multiplicamos $\boldsymbol{C}\cdot\boldsymbol{C}$ y luego multiplicamos el resultado por $\boldsymbol{x}$, o sea, hacemos el cálculo $((\boldsymbol{C}\cdot\boldsymbol{C})\cdot\boldsymbol{x})$ el número de operaciones (multiplicaciones-suma) que se requerirán es n^3 para el producto de las matrices y n^2 para el producto de la matriz por el vector, es decir, un total de $n^3 + n^2$. Por otra parte, si se multiplica primero $\boldsymbol{C}\cdot\boldsymbol{x}$ y el resultado se premultiplica por $\boldsymbol{C}$, en otras palabras, se calcula $(\boldsymbol{C}\cdot(\boldsymbol{C}\cdot\mathbf{x}))$ en cada paso se requieren n^2 multiplicaciones-suma, un total de $2n^2$ operaciones. Como suponemos que $n > 1$, se tiene que $n^3 + n^2 > 2n^2$. Por ejemplo, si $n = 10$, $n^3 + n^2 = 1100$, $2n^2 = 200$, es decir una alternativa cuesta más de 5 veces más que la otra en tiempo. Para $n = 1000$, $n^3 + n^2 = 10^9 + 10^6$, $2n^2 = 2\cdot10^6$, es decir, una alternativa es 500 veces más costosa en tiempo que la otra. Lo anterior es cierto cuando se manejan números de punto flotante que en cada secuencia de operaciones multiplicación – suma producen otro número de punto flotante. La situación se complica en otros casos, debido a que las operaciones son más costosas dependiendo de la longitud de cadenas alfabéticas que se van presentando. Aún en el caso de manejar enteros, al ir creciendo los números en magnitud, el trabajo aumenta.

Implantación en la computadora. Cuenta de caminos

La existencia de paquetes de computación comerciales para matrices como MATLAB facilita la implantación de los métodos discutidos. El problema más sencillo de computarizar es el de la cuenta de caminos, la cual sólo requiere la multiplicación convencional de matrices. Consideremos la gráfica de la figura 4, cuya matriz de conexión se muestra a su derecha .

DOI: http://dx.doi.org/10.22201/fi.25940732e.2010.11n1.001



$$\mathbf{C} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Para introducir la matriz **C** a MATLAB se teclea lo siguiente:

```
>> c=[0,1,0,1;0,0,1,1;0,0,0,0;0,0,1,0]
c =
     0     1     0     1
     0     0     1     1
     0     0     0     0
     0     0     1     0
```

La línea marcada >> la teclea el usuario, lo demás lo escribe el programa. Para calcular el número de caminos de 2 tramos se calcula $\mathbf{C}^2$ tecleando lo siguiente:

```
>> c^2
ans =
     0     0     2     1
     0     0     1     0
     0     0     0     0
     0     0     0     0
```

Igualmente se puede calcular el número de caminos de 3 tramos:

```
>> c^3
ans =
     0     0     1     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

Si se calculara $\mathbf{C}^4$, encontraríamos que no hay ningún camino de 4 tramos.

```
>> c^4
ans =
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

Una vez que aparece una matriz cero entre las potencias de **C** todas las demás serán cero. Si ahora nos interesa saber el número de caminos de todas las longitudes en tramos podemos calcular la suma $\mathbf{C} + \mathbf{C}^2 + \mathbf{C}^3$

```
>> c+c^2+c^3
ans =
     0     1     3     2
     0     0     2     1                (*)
     0     0     0     0
     0     0     1     0
```

El lector puede verificar los resultados anteriores. En redes más grandes que no tengan ciclos (como por ejemplo las redes de ruta crítica) la red no puede tener caminos con más tramos que el número de nodos en la red. Por lo tanto, si n es el número de nodos de una red acíclica, al número total de caminos con cualquier número de tramos se puede calcular obteniendo la suma $\mathbf{C} + \mathbf{C}^2 + \mathbf{C}^3 + \ldots + \mathbf{C}^n$.

Cuando n es grande resulta laborioso calcular y sumar tantas potencias. Podemos, sin embargo, echar mano de la teoría de matrices que dice que si la serie infinita de potencias de una matriz converge absolutamente, se puede escribir:

$$\mathbf{C} + \mathbf{C}^2 + \mathbf{C}^3 + \ldots = \mathbf{C}(\mathbf{I} - \mathbf{C})^{-1}.$$

Para el caso de redes acíclicas, la serie siempre converge absolutamente pues sólo tiene un número finito de términos.

Por lo tanto, podemos calcular el número total de caminos con la fórmula, la cual es mucho más eficiente computacionalmente. Si hacemos esto para nuestro ejemplo se tendrá

```
>> c/(eye(4)-c)
ans =
     0     1     3     2
     0     0     2     1
     0     0     0     0
     0     0     1     0
```

donde en MATLAB la matriz unidad de orden n se escribe eye(n) y la raya de quebrado nos indica la matriz

DOI: http://dx.doi.org/10.22201/fi.25940732e.2010.11n1.001

inversa. Vemos que hemos obtenido el mismo resultado que sumando las potencias no cero de **C**.

Implantación en la computadora. Conexiones entre nodos

Otro problema que se puede computarizar con facilidad con MATLAB es el de determinar la conectividad de una red. En ese caso, según se vio antes, en vez de la operación + se usa la operación lógica OR, que en MATLAB se escribe "|", y en vez de la operación ×, se usa la operación lógica AND, que en MATLAB se escribe "&". Los operadores se pueden aplicar a pares de elementos de una matriz y también a matrices completas. En el caso de aplicárselo a matrices completas, la operación se aplica elemento por elemento en cada uno de los componentes de las dos matrices. No existe la operación de multiplicar matrices usando las operaciones "|" y "&" en vez de multiplicación y suma, por lo que hay que elaborar un M – file que la realice. Escribimos una muy sencilla, simplemente para ilustrar las ideas. En MATLAB los valores "verdadero" y "falso" se representan con 1 y 0, que juegan los papeles de la unidad y el cero del conjunto que se va a manejar. El listado del programa es

```
function lgicMMul(a, b)
A=logical(a);
B=logical(b);
for i=1:size(A,1)
    for j=1:size(A,1)
        s=logical(0);
        for k=1:size(A,1)
            s=s|(A(i,k)&B(k,j));
        end
        M(i,j)=s;
    end
end;
lgicMMul = M
```

Para la matriz de conexión que usamos en el ejemplo anterior, su "cuadrado" se obtiene como sigue:

```
>> C=[0,1,0,1;0,0,1,1;0,0,0,0;0,0,1,0];
>> lgicMMul (C, C)
lgicMMul =
     0     0     1     1
     0     0     1     0
     0     0     0     0
     0     0     0     0
>> C2=M;
```

Hemos guardado el resultado en C2. La matriz lógica M nos indica con 1's los nodos que están conectados por caminos con dos tramos. Para calcular el "cubo" de la matriz de conexión, ponemos M en a y C en b y volvemos a invocar el M – file "lgicMMul" y desplegamos la M resultante

```
>> lgicMMul (M, C)
lgicMMul =
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     1     0
>> C3 = M;
```

Hemos guardado el resultado en C3. Para calcular la "cuarta potencia" de C ponemos M en a y C en b y una vez más invocamos > lgicMMul

```
>> lgicMMul (M,C)
lgicMMul =
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
>> C4 = M
```

Hemos guardado el resultado en C4. Como la "cuarta potencia" de C resultó cero, es decir, no existen caminos de cuatro tramos en la gráfica, todas las potencias más altas que la tercera serán cero.

Si queremos saber cuáles nodos están conectados (con cualquier número de tramos) obtenemos la matriz $\mathbf{T} = \mathbf{C}$ OR $\mathbf{C}^2$ OR $\mathbf{C}^3$ OR $\mathbf{C}^4$ que están guardadas en C, C1, C2, C3 y C4.

```
>> T = C | C2 | C3 | C4
T =
     0     1     1     1
     0     0     1     1
     0     0     0     0
     0     0     1     0
```

Existe otra manera de determinar todas las conexiones de hasta *n* tramos. Consiste en encontrar la *n – ésima* potencia (lógica) de (I + C) y a este resultado restarle la matriz unidad, es decir, calcular $\mathbf{T} = (\mathbf{I} + \mathbf{C})^n - \mathbf{I}$ con operaciones lógicas.

Por ejemplo, para el ejemplo que venimos resolviendo se tiene

DOI: http://dx.doi.org/10.22201/fi.25940732e.2010.11n1.001

$$(I + C)^3 - I = I^3 + 3IC + 3IC^2 + C^3 - I = 3C + 3C^2 + C^3 = C + C^2 + C^3,$$

donde los tres primeros miembros están manejados algebraicamente (como números reales con multiplicación y suma) y el cuarto lógicamente (con variables booleanas y operaciones AND y OR). Una interpretación gráfica de lo anterior es que se agregan ramas que van de un nodo a sí mismo, lo cual permite que, por ejemplo, los caminos de tres tramos se formen con todos los "auténticos" de un tramo más dos vueltas a la rama que sale y regresa al nodo inicial (o una vuelta en el nodo inicial y una en el nodo final, o dos vueltas en el nodo final) y una situación similar para los caminos "auténticos" de dos ramas. Al final, se resta la matriz unidad para desaparecer los tramos que van de un nodo a sí mismo. Vale la pena señalar que si no se cuenta con un programa con operadores lógicos la matriz de conexión se puede deducir fácilmente de la matriz de número de caminos tomando cualquier componente mayor que cero como 1 y cualquiera que es cero como 0. Esto se puede verificar aplicándole el criterio a la matriz de la ecuación (*).

Implantación en la computadora. Rutas más cortas

Además de encontrar las longitudes de las rutas más cortas también es importante determinar por dónde pasan esas rutas más cortas. Para ello se utiliza una matriz de nodos N que se inicializa, para el ejemplo que nos ocupa, como sigue:

$$N = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{bmatrix}$$

Durante el proceso la matriz N se va modificando cada vez que se mejora una distancia con la función *Min*. Se utilizará un nuevo M – file para resolverlo, ambos con MATLAB, mismo que se muestra a continuación.

```
function [x,y] = rutcort(a, b, n)
for i=1:size(a,1)
    for j=1:size(a,1)
        s=999999;
        for k=1:size(a,1)
            s=min([s,(a(i,k)+b(k,j))]);
           if s==(a(i,k)+b(k,j))
                n(i,j)=k;
           end;
        end
        M(i,j)=s;
    end
end;
y=n;
x=M;
```

Vamos a calcular las longitudes de las rutas más cortas de la gráfica de la figura 3, cuya matriz de conexión se muestra junto a ella. Iremos guardando las potencias de C en las matrices C1, C2, C3. Comenzamos por guardar la matriz de conexión de la figra 3 en C y la matriz inicial de nodos en N. (Estas operaciones no se anotan). A continuación, tecleamos en MATLAB lo que se muestra. Las líneas precedidas de >> las teclea el usuario, mientras que lo demás lo produce el programa.

```
>> [C2,N2]=rutcort(C,C,N)
C2 =
      0    11    30    23    13    34    9999
     11     0    31    12     2    21    9999
     30    31     0    13    30     4    9999
     23    12    13     0    11     9    9999
     13     2    30    11     0    20    9999
     34    21     4     9    20     0    9999
   9999     3     5    10     1     1       0
N2 =
      1     2     3     2     2     3     7
      2     2     4     4     5     4     7
      3     4     3     6     4     6     7
      2     4     6     4     5     6     7
      2     5     4     5     5     4     7
      3     4     6     6     4     6     7
      7     5     6     6     7     7     7
>> [C3,N3]=rutcort(C2,C,N2)
C3 =
      0    11    30    23    13    32    9999
     11     0    25    12     2    21    9999
     30    25     0    13    24     4    9999
     23    12    13     0    11     9    9999
     13     2    24    11     0    20    9999
     32    21     4     9    20     0    9999
     14     3     5    10     1     1       0
```

DOI: http://dx.doi.org/10.22201/fi.25940732e.2010.11n1.001


```
N3 =
     1     2     3     4     5     4     7
     2     2     6     4     5     6     7
     3     4     3     6     4     6     7
     2     4     6     4     5     6     7
     2     5     6     5     5     6     7
     2     4     6     6     5     6     7
     2     5     6     6     7     7     7
>> [C4,N4]=rutcort(C3,C,N3)
C4 =
     0    11    30    23    13    32    9999
    11     0    25    12     2    21    9999
    30    25     0    13    24     4    9999
    23    12    13     0    11     9    9999
    13     2    24    11     0    20    9999
    32    21     4     9    20     0    9999
    14     3     5    10     1     1       0
N4 =
     1     2     3     4     5     6     7
     2     2     6     4     5     6     7
     3     4     3     6     5     6     7
     2     4     6     4     5     6     7
     2     5     6     5     5     6     7
     2     4     6     6     5     6     7
     2     5     6     6     7     7     7
```

Comparamos la séptima fila de la última matriz con los resultados calculados a mano previamente y vemos que son iguales, tomando en consideración que en vez de infinito hemos escrito 9999 que es un número muy grande comparado con el resto de los números de la matriz **C**. Como ya coincidieron C3 y C4, así como N3 y N4, el proceso se puede suspender, pues todas las demás "potencias" de C serán iguales.

Para mostrar la utilidad de la matriz de nodos N4, supongamos que se desea encontrar la ruta más corta entre el nodo *7* y el nodo *1*. Como el arreglo es de *últimos nodos intermedios,* en la posición *(7, 1),* es decir, en la fila *7* y columna *1,* se encuentra el último nodo intermedio en el camino más corto de *1* a *7*.

Dicho nodo es 2. Es decir, la ruta es 7, ... , 2, 1, donde los puntos suspensivos representan otros posibles nodos aún desconocidos. Para encontrar el resto de la ruta ahora se busca el último nodo intermedio en la ruta más corta de *7* a *2*. Dicho nodo se encuentra en la posición *(7, 2)* y para nuestro ejemplo es *5.* La ruta hasta donde sabemos es ahora 7, ... , 5, 2, 1. El siguiente paso es consultar la posición *(7, 5)* cuyo valor es *7.* Como hemos llegado al nodo de partida, se ha determinado por completo la ruta que es: 7, 5, 2, 1. Consultando la figura 3, verificamos que la longitud, que vale 1 + 2 + 11 = 14, es igual a la cantidad en la posición *(7, 1)* en el arreglo de distancias mínimas C4.

Implantación en la computadora. Enumeración de caminos

Para computarizar la enumeración de caminos es conveniente hacerlo en un lenguaje que maneje estructuras de tamaños variables. También resulta muy conveniente que el lenguaje tenga buenas facilidades para hacer recurrencias. Esto, en general, lo manejan muy bien lenguajes funcionales como LISP y Logo, ambos lenguajes no son muy manejados por los ingenieros, excepto los de ciencias de la computación. No los utilizamos en este artículo para evitar alargarlo más, ya que tendríamos que incluir varios listados para leer y desplegar datos, así como el procedimiento para hacer los "productos" de las matrices y las funciones necesarias para hacer las concatenaciones de los símbolos y otros detalles como el manejo de arreglos por medio de listas. El lector interesado puede comunicarse con el autor quien le proveerá todos estos elementos en el caso de Logo. El lenguaje de MATLAB, no obstante de no manejar listas, tiene funciones muy versátiles para manejar cadeas alfabéticas, y será con ellos que implantaremos en la computadora la enumeración de caminos. Se debe aclarar que esencialmente todo se puede lograr en virtualmente cualquier lenguaje de computación, pero con diferentes grados de dificultad. Esto quedó ilustrado en el caso de la computarización del número de caminos entre cada par de nodos. El lenguaje de MATLAB nos permite hacer multiplicaciones entre las matrices A y B simplemente escribiendo A*B, lo que en otros lenguajes requeriría la escritura de una subrutina con consiguientes declaraciones y dimensionamientos, así como subrutinas para desplegarlas, acción que en MATLAB se logra simplemente tecleando el nombre de la matriz. A este respecto, la computarización de la enumeración de caminos se puede lograr con MATLAB siempre que el usuario entre en temas como arreglos de células de cadenas simbólicas y algunas funciones para manejo de cadenas simbólicas.

Pasamos ahora a computarizar la enumeración de caminos usando cadenas simbólicas. Para denotar varios caminos escribiremos un tramo con una letra, un camino con las letras minúsculas de los tramos que lo componen yuxtapuestas y los diferentes caminos separados con signos "+". Para simplificar, sólo consideraremos gráficas con tramos sencillos entre un par de nodos dejando fuera las que tengan tramos en paralelo. Requerimos una subrutina que a una suma de varias

DOI: http://dx.doi.org/10.22201/fi.25940732e.2010.11n1.001

secuencias de letras yuxtapuestas le anteponga a cada sumando un letra para añadirle a cada conjunto de caminos un tramo más. Así si llamamos addchar a la subrutina (función) que logra lo anterior y la letra a añadir es

a = 'x' y los caminos previos son

b = 'pqr+stu+vwz', entonces la función

addchar(a,b) = 'xpqr+xsty+xvwz'. El listado de la función addchar(a,b) es el siguiente

```
function [c]=addchar(a,b)
s=size(b,2);
k=0;
c=strcat(a,b);
i=2;
k=k+1;
while i<=s
    if b(i:i)=='+'
    c=strcat(c(1:i+k),a,b(i+1:s));
    k=k+1;
    i=i+1;
    end
    i=i+1;
end
c
return
```

Ahora se requiere un programa que "multiplique" matrices, una de ellas con una sola letra en algunas componentes y vacía en otros y la segunda con algunas componentes parecidas a b, que fue mostrada arriba y otras vacías. Como en la multiplicación de matrices se requiere tanto la "multiplicación" de escalares y la "suma" de escalares, como se indicó (tabla 1) nuestros escalares serán cadenas alfabéticas, el "cero" corresponde al conjunto vacío y el "uno" a la cadena alfabética vacía, la "multiplicación" a la concatenación de cadenas alfabéticas y la "suma" a la unión de conjuntos de cadenas de símbolos. La subrutina para hacer la multiplicación de matrices se muestra a continuación; está diseñada para ir multiplicando la matriz de conexión en la izquierda y las potencias de la matriz de conexión a la derecha. Para multiplicar dos potencias de C es necesario hacerle cambios a la subrutina mataddchar y a la subrutina addchar. Esto no lo mostramos:

```
function d=mataddchar(c1,c2)
for i=1:size(c1,2)
    for j=1:size(c1,2)
        s='';
        d{i,j}='';
        for k=1:size(c1,2)
        s1=c1{i,k};
        s2=c2{k,j};
        if size(s1,2)==0 |
size(s2,2)==0
        elseif size(d{i,j},2)==0
            s=strcat(s,'+',addchar(c1{,
            k},c2{k,j}));
            end
        end
        if size(s)==0
        elseif s(1)=='+'
            s=s(2:length(s));
        end
        d{i,j}=s;
    end
end
```

Ahora, introducimos la matriz de conexión de la figura 1 que aparece a la izquierda de la gráfica. Nótense las llaves "{" y "}" que denotan arreglos de células en vez de matrices. Las matrices tienen elementos del mismo tipo en todas las componentes y con igual longitud, los arreglos de células no.

```
C={'a','b','','c','';'','','d','e','';'
','','','f','';'','','g','','';'','','h
','',''}
C =
    'a'    'b'     ''    'c'     ''
    ''     ''    'd'    'e'     ''
    ''     ''     ''    'f'     ''
    ''     ''    'g'    ''      ''
    ''     ''    'h'    ''      ''
```

Nótese que donde no hay ramas se pone la cadena vacía que se representa con dos apóstrofes seguidos. En seguida se calcula el "cuadrado", C2, de la matriz de conexión

```
>> C2=mataddchar(C,C)
C2 =
    'aa'   'ab'   'bd+cg'   'ac+be'    ''
     ''     ''    'eg'      'df'      ''
     ''     ''    'fg'       ''       ''
     ''     ''     ''       'gf'      ''
     ''     ''     ''       'hf'      ''
```

y el "cubo", C3, de la matriz de conexión

DOI: http://dx.doi.org/10.22201/fi.25940732e.2010.11n1.001


```
>> C3=mataddchar(C,C2)
C3 =
      'aaa'       'aab'       'abd+acg+beg'       'aac+abe+bdf+cgf'       ''
       ''          ''             'dfg'               'egf'              ''
       ''          ''               ''                'fgf'              ''
       ''          ''             'gfg'                ''                ''
       ''          ''             'hfg'                ''                ''
```

Estas matrices se habían obtenido arriba manualmente y coinciden con lo calculado por el programa.

Existen varios otros problemas de caminos que se pueden resolver con los programas dados, pero en cuyos detalles no entramos por falta de espacio. Por ejemplo, en la aplicación del Método de Ruta Crítica interesa encontrar el camino más largo en una red acíclica. Para ese problema el semianillo adecuado son los números reales aumentados con ∞ y las dos operaciones adecuadas son: "suma": Max (x, y), "multiplicación": + (suma de reales), neutro de la "suma": –∞, neutro de la "multiplicación": 0. Otro ejemplo es encontrar el camino de máxima capacidad. Aquí el semianillo adecuado es el de los reales positivos aumentados con ∞ y las dos operaciones adecuadas son "suma": Max (x, y); "multiplicación": Min (x, y); neutro de la "suma": 0; neutro de la "multiplicación": ∞.

Como último ejemplo a mencionar es el problema de encontrar el camino de máxima confiabilidad, en el cual el semianillo adecuado es el intervalo de los reales entre cero y uno, la "suma" es Max (x, y); la "multiplicación" es × (multiplicación entre reales), el neutro de la "suma" es 0 (cero de los reales) y el neutro de la "multiplicación" es 1 (la unidad de los reales).

Comentarios matemáticos

Muchos problemas de caminos en redes orientadas tienen una estructura común si dichos problemas se plantean como problemas algebraicos con elementos más generales que el álgebra lineal con números reales y complejos. Un sistema algebraico adecuado es el semianillo o dioide. Las técnicas para resolver problemas algebraicos con dioides, aunque difieren de las del álgebra lineal estándar son muy parecidas. Aunque no fueron tratadas en este artículo desde el punto de vista matemático con demostración de teoremas y todo, esencialmente se basan en que las soluciones están dadas por potencias de las matrices o series infinitas de las mismas. En muchos casos de interés las series se truncan porque arriba de ciertas potencias todas las potencias se anulan. Para algunos casos la serie completa se puede calcular utilizando algoritmos que son generalizaciones de los algoritmos clásicos del álgebra lineal, tales como el algoritmo de Gauss-Jordan, el de Jacobi y el de Gauss-Seidel. En el artículo, sin mencionarlo, se utilizó el algoritmo de Jacobi para resolver el problema de rutas más cortas, que coincide con el método de Programación Dinámica de Bellman. Para que el artículo sea fácilmente accesible a ingenieros, se prefirió sacrificar demostraciones matemáticas restringiendo la discusión a explicaciones heurísticas para tener espacio para mostrar programas en lenguajes correspondientes a paquetes de uso corriente de los ingenieros como MATLAB y dar ejemplos numéricos resueltos en todo detalle de diversos tipos de problemas de caminos en redes.

Conclusiones

Se ha presentado un enfoque unificado para plantear y resolver con auxilio de una computadora problemas de caminos en redes orientadas. La principal herramienta ha sido la matriz de conexión generalizada para que sus componentes sean elementos de un semianillo o dioide. Los elementos pueden ser números enteros, números reales, variables booleanas y conjuntos de cadenas alfanuméricas. Todos ellos se pueden manejar en cualquier lenguaje de programación con diversos grados de dificultad para hacerlo. Su manejo se ilustró por medio de programas en MATLAB haciendo algunos cambios para resolver problemas distintos. Se dieron varios ejemplos ilustrativos en todo detalle, ambos ingredientes que faltan en la obra de Gondran y Minoux (1984).

Referencias

Bellman R., Cooke K.L. y Lockett J.A. *Algorithms, Graphs and Computer.* New York. Academic Press. 1970.

Bellman R.E. y Dreyfus S.E. *Applied Dynamic Programming.* Princeton, NJ. Princeton University Press. 1962.

Dalhquist G. y Björck A. *Numerical Methods.* Englewood Cliffs. NJ. Prentice-Hall, Inc., 1974. Pp. 146–161.

Gondran M. y Minoux M. *Graphs and Algorithms.* New York. John Wiley & Sons, Inc. 1984.

DOI: http://dx.doi.org/10.22201/fi.25940732e.2010.11n1.001

Hu T.C. *Integer Programming and Network Flows.* Addison-Wesley Publishing Company. Reading, MA. 1969. Pp. 158–160.

Kaufman A. y Malgrange Y. Recherche des chemins et circuit hamiltoniens d'un graphe. *Rev. Fr.: Rech. Op.,* 26:61–73.1963.

MacLane S. y Birkhoff G. *Algebra.* New York. The Macmillan Company. 1967.

Mayeda W. *Graph Theory.* New York. John Wiley & Sons, Inc. 1972.

Semblanza del autor

Marco Antonio Murray-Lasso. Es ingeniero mecánico electricista de la UNAM, maestro en ciencias en ingeniería eléctrica y doctor en ciencias en control automático, ambos del Massachusetts Institute of Technology (MIT). Ha sido profesor de Case Western Reserve University y del Newark College of Engineering, investigador de Bell Telephone Laboratories y asesor de la NASA. Ha sido investigador nacional y fue presidente fundador de la Academia Nacional de Ingeniería, miembro fundador y posteriormente presidente mundial del Consejo de Academias Nacionales de Ingeniería y Ciencias Tecnológicas (CAETS). También ha sido secretario de la Academia Mexicana de Ciencias y presidente del Consejo de Honor de la Academia Mexicana de Ciencias, Artes, Tecnología y Humanidades, así como miembro de la Academia de Ciencias de Nueva York, de la Academia Mexicana de Informática y miembro fundador de la Academia Mexicana de Tecnología. Actualmente es académico de honor de la Academia Mexicana de Ciencia de Sistemas, de la Academia de Ingeniería y de la Academia Mexicana de Ciencias, Artes, Tecnología y Humanidades. Ha sido profesor en la Facultad de Ingeniería durante 48 años. Autor o coautor de 12 libros y 250 artículos técnicos. Durante 25 años ha sido Consejero Educativo del MIT. Su biografía aparece en *Who is who in Science and Engineering* y en *Who is who in the World.*