

MATRICES DISPERAS

Basada en la presentación de Miguel Vargas- Felix. CIMAT

Imparte: Maria Trinidad Pimentel Villegas . ITSUR

investigacion@itsur.edu.mx

maria.pimentel@cimat.mx



Matrices dispersas

Son matrices en las cuales la gran mayoría de las entradas son cero.

En inglés se les conoce como “sparse matrices”, en algunos países de habla hispana también se les conoce como “matrices ralas”.

Sea \mathbf{A} una matriz dispersa de tamaño $n \times n$. Definamos la notación $\eta(\mathbf{A})$, que indica el número de entradas no cero de \mathbf{A} .

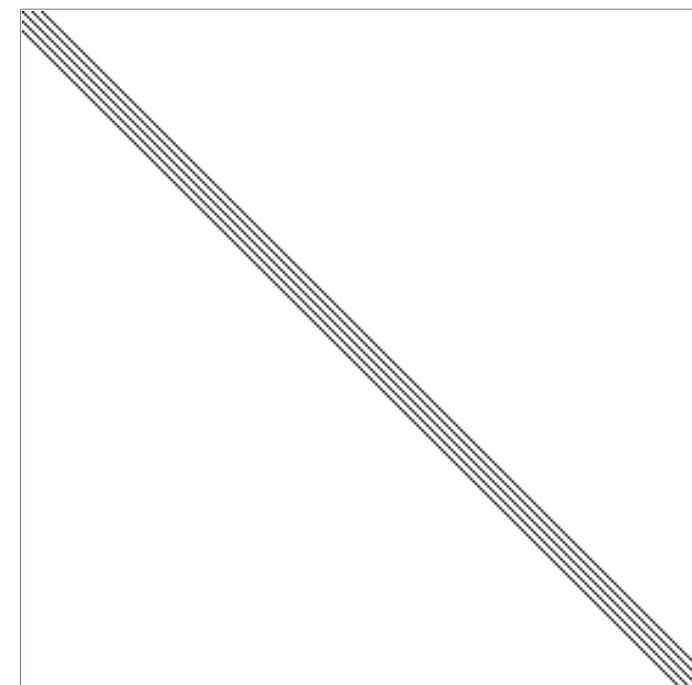
Por ejemplo

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & 0 & a_{24} & a_{25} & 0 & 0 \\ a_{31} & 0 & a_{33} & a_{34} & 0 & 0 & 0 \\ 0 & a_{42} & a_{43} & a_{44} & 0 & a_{46} & 0 \\ 0 & a_{52} & 0 & 0 & a_{55} & 0 & a_{57} \\ 0 & 0 & 0 & a_{64} & 0 & a_{66} & a_{67} \\ 0 & 0 & 0 & 0 & a_{75} & a_{76} & 0 \end{pmatrix},$$

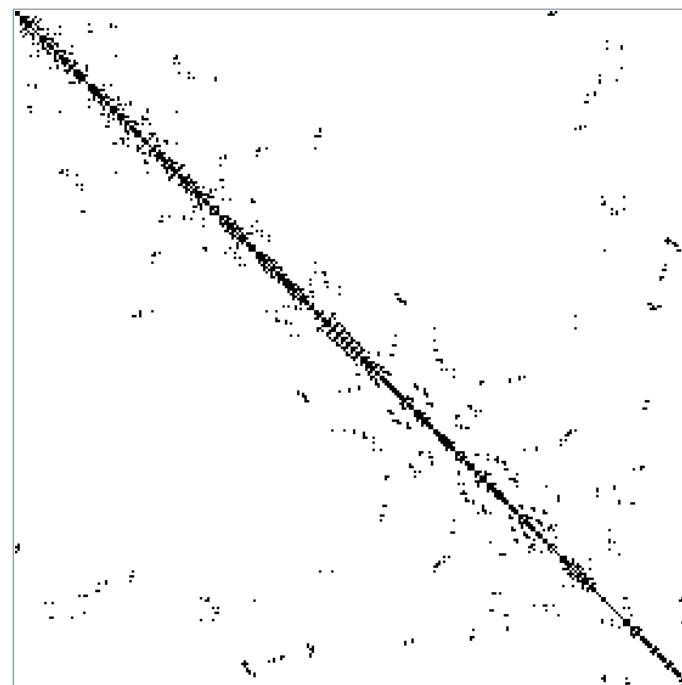
el número de entradas de \mathbf{A} es $7 \times 7 = 49$, el número de entradas distintas de cero es $\eta(\mathbf{A}) = 22$.

Con las matrices dispersas se pueden trabajar fácilmente matrices de millones de renglones por millones de columnas.

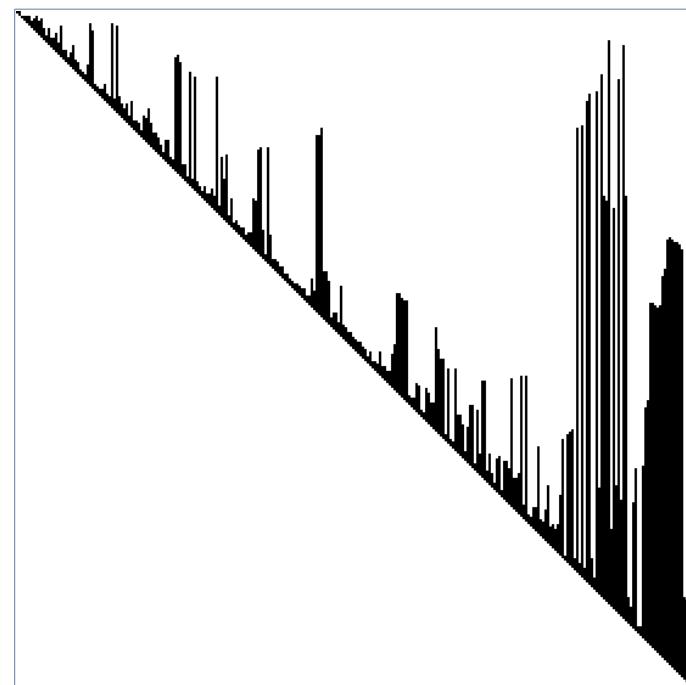
A veces se requiere solo visualizar las entradas distintas de cero, para ello se usan imágenes como las siguientes:



Bandada



Simétrica



Triangular

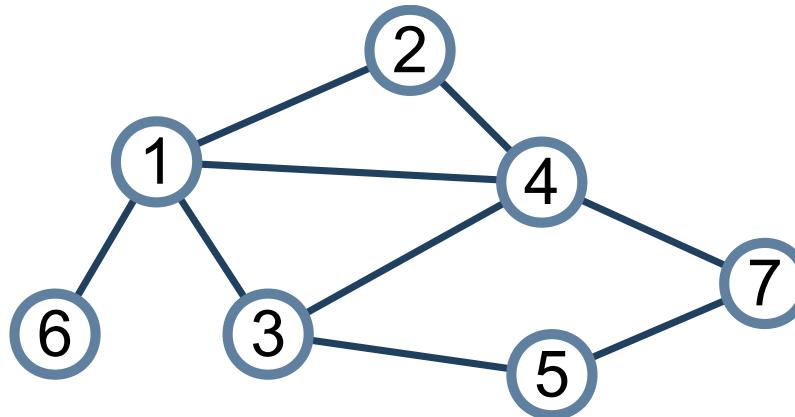
Los puntos negros representan las entradas de la matriz que son distintas de cero.

Matrices dispersas como grafos

Un grafo está formado por un conjunto de vértices y un conjunto de aristas $G = (\mathbf{V}, \mathbf{E})$.

- Los vértices están numerados, \mathbf{V} es el conjunto de todos los vértices.
- Cada arista conecta dos vértices. El conjunto de aristas \mathbf{E} está formado por pares no ordenados de vértices. Los grafos en los cuales no importa el orden se llaman grafos no dirigidos.

Por ejemplo:



En este caso, el conjunto de vértices es

$$\mathbf{V} = \{1, 2, 3, 4, 5, 6, 7\},$$

el conjunto de aristas es

$$\mathbf{E} = \{(1, 2), (1, 3), (1, 4), (1, 6), (2, 4), (3, 4), (3, 5), (4, 7), (5, 7)\}.$$

Una matriz dispersa con estructura simétrica se puede representar como un grafo no dirigido.

En el grafo no están representados los valores de la matriz, sólo la estructura de las entradas distintas de cero.

Sea \mathbf{A} una matriz dispersa de tamaño $n \times n$.

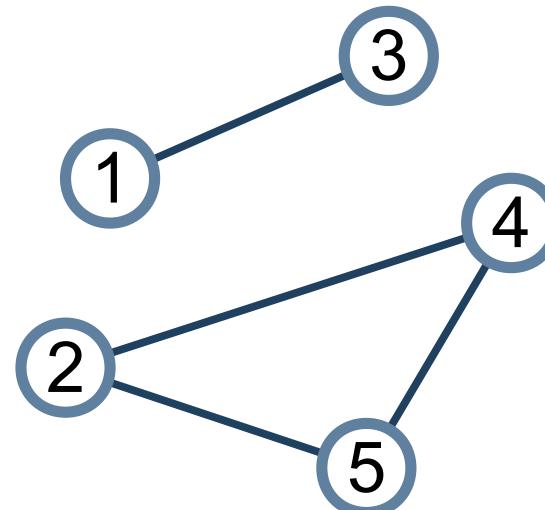
Cada vértice del grafo de \mathbf{A} representa un renglón (columna) de la matriz, $\mathbf{V} = \{v_1, v_2, \dots, v_n\}$.

El número de aristas es igual al número de entradas no cero de la matriz $\eta(\mathbf{A})$, así $\mathbf{E} = \{e_1, e_2, \dots, e_{\eta(\mathbf{A})}\}$.

Cada entrada no cero de la matriz $a_{ij} \neq 0$ es representada con una arista $e_k = (v_i, v_j)$, con $k = 1, 2, \dots, \eta(\mathbf{A})$.

Ejemplo:

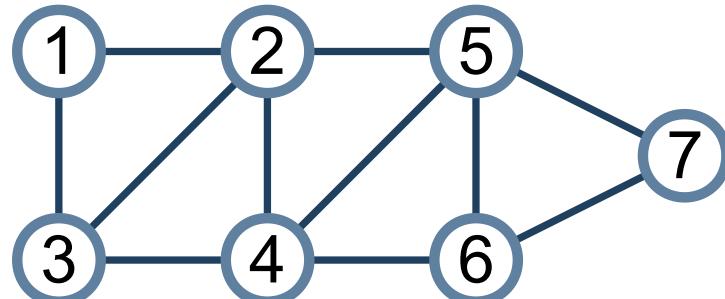
$$\begin{pmatrix} 3 & 0 & 9 & 0 & 0 \\ 0 & 4 & 0 & 6 & 1 \\ 9 & 0 & 2 & 0 & 0 \\ 0 & 6 & 0 & 8 & 5 \\ 0 & 1 & 0 & 5 & 3 \end{pmatrix}$$



Matrices dispersas de mallas de elemento finito

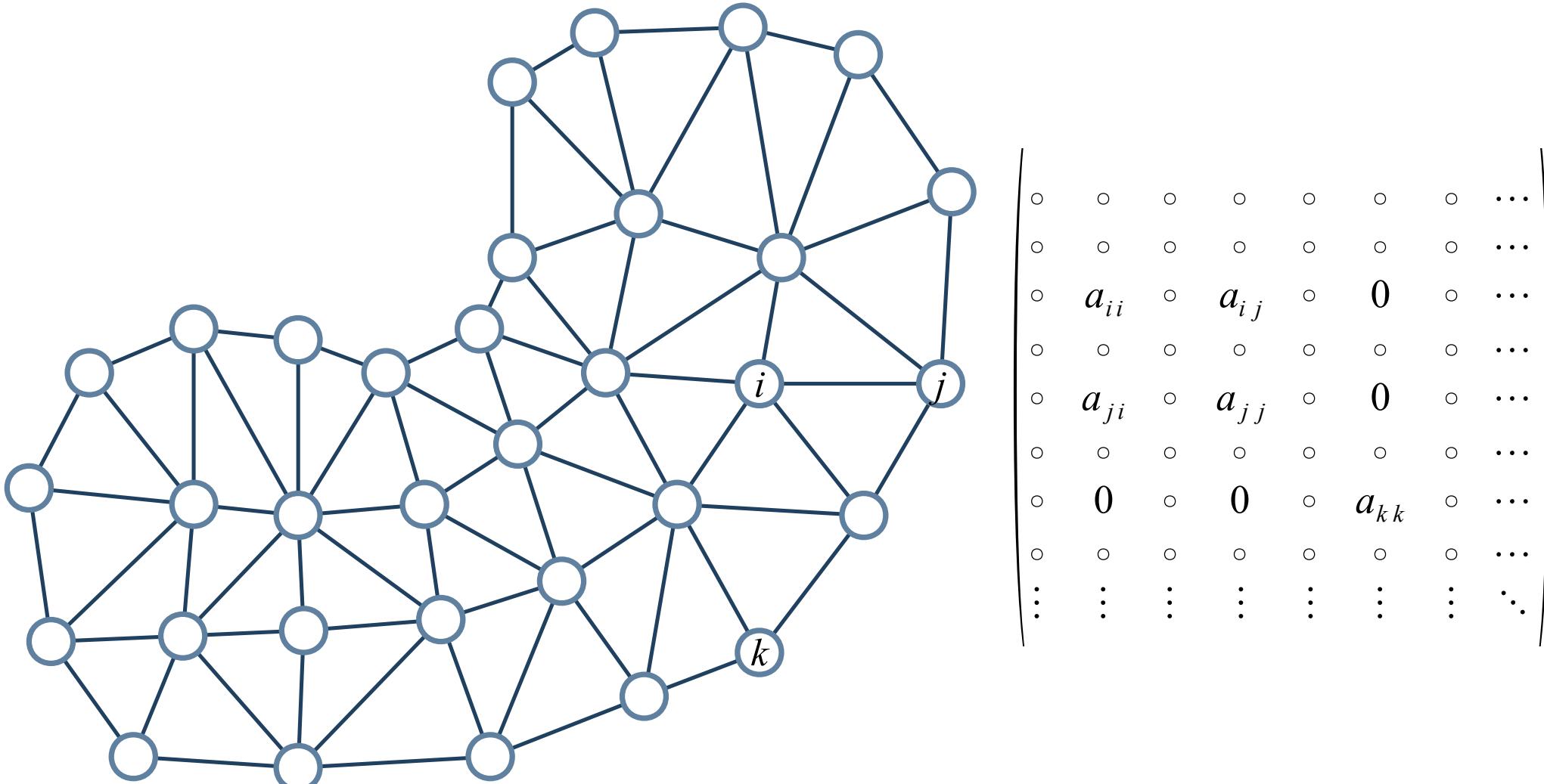
En el método de elemento finito, se trabaja de forma contraria, se parte de la malla (grafo) y con la matriz de conectividades se genera la matriz dispersa.

En las mallas de elemento finito tenemos un dominio discretizado en elementos, triángulos o cuadriláteros en 2D, tetraedros o hexaedros en 3D.



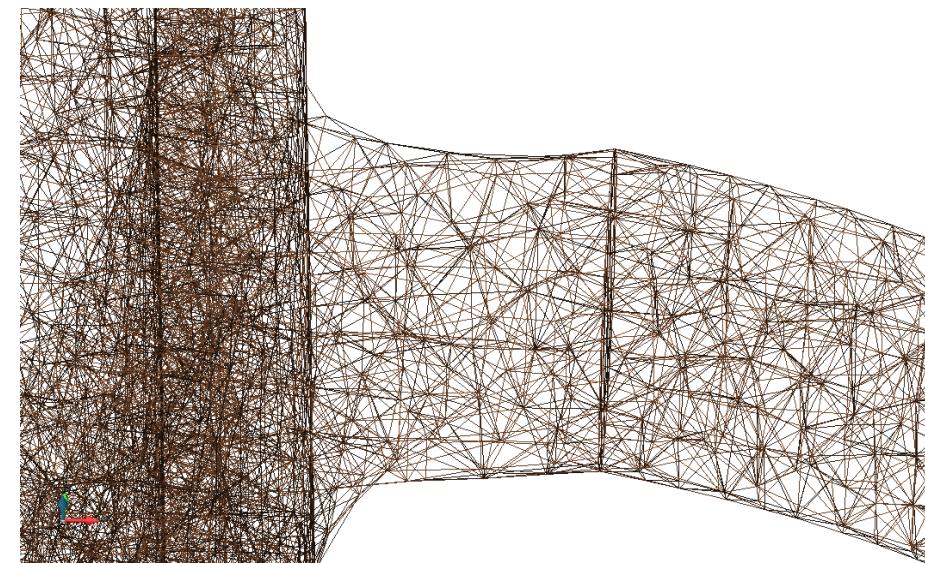
$$\begin{vmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & 0 & 0 \\ a_{31} & a_{32} & a_{33} & a_{34} & 0 & 0 & 0 \\ 0 & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & 0 \\ 0 & a_{52} & 0 & a_{54} & a_{55} & a_{56} & a_{57} \\ 0 & 0 & 0 & a_{64} & a_{65} & a_{66} & a_{67} \\ 0 & 0 & 0 & 0 & a_{75} & a_{76} & a_{77} \end{vmatrix}$$

Dado que un nodo de la malla se conecta sólo con pocos nodos, tendremos una matriz muy dispersa. En general, el tamaño del número de entradas distintas de cero será $O(n)$.

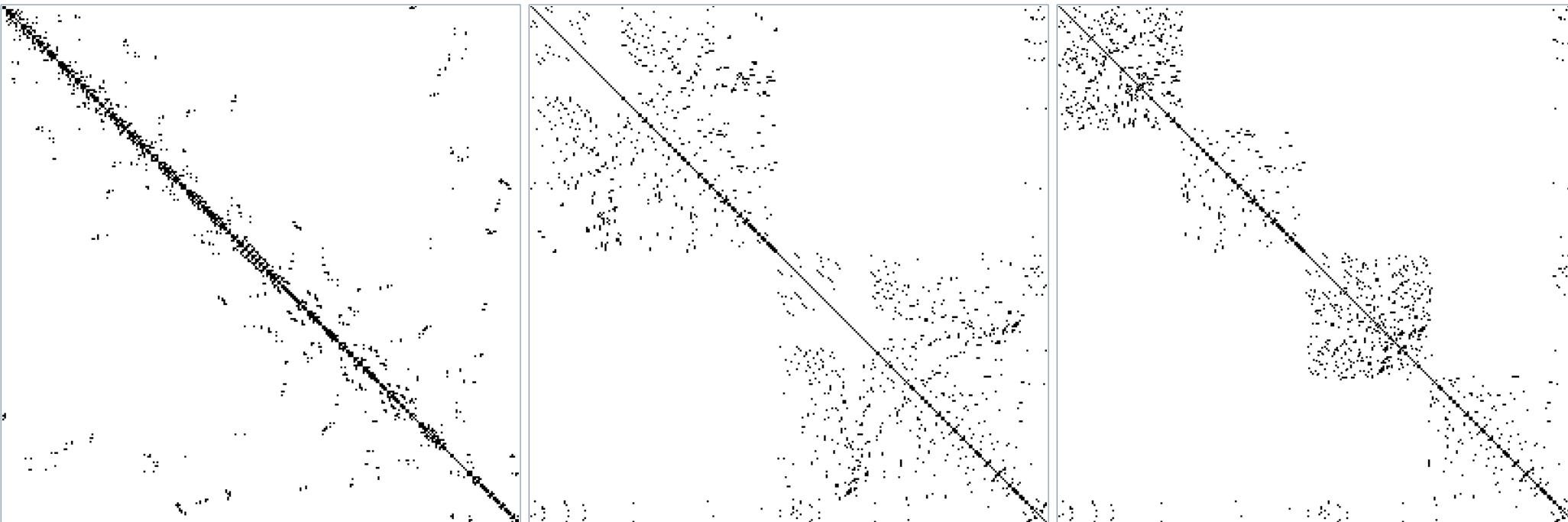


En el ejemplo, $a_{ij} \neq 0$ dado que están conectados con la arista (i, j) . En cambio $a_{ik} = 0$ puesto que no existe una arista (i, k) . De igual forma $a_{kj} = 0$ dado que no existe (k, j) .

Las mallas pueden llegar a ser muy complejas. El siguiente es el ejemplo de una malla de elemento finito en 3D formada por tetraedros.



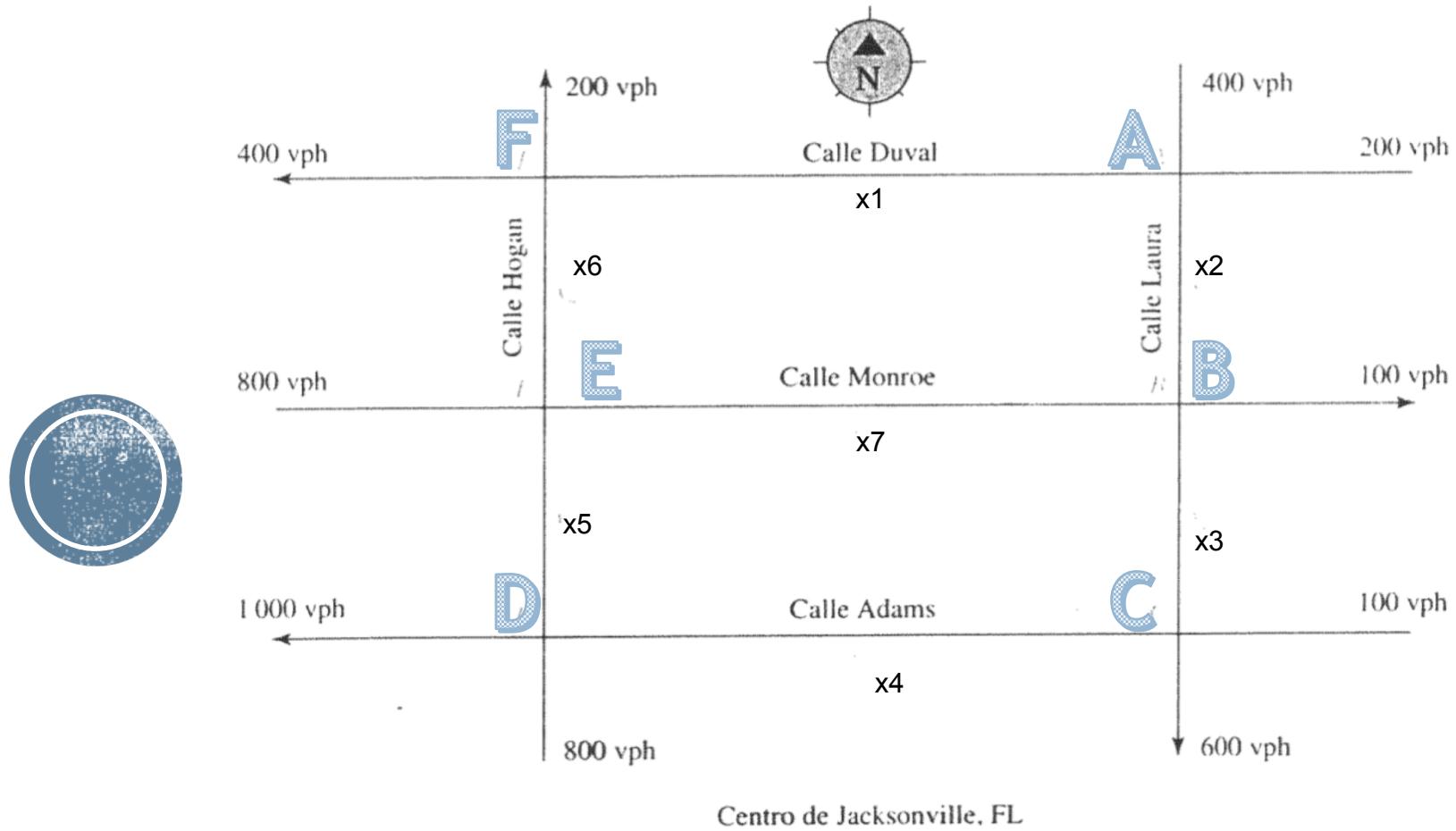
Poder representar la estructura de matrices dispersas como grafos nos permitirá aplicar varios conceptos de teoría de grafos que resultarán en transformaciones a la matriz.



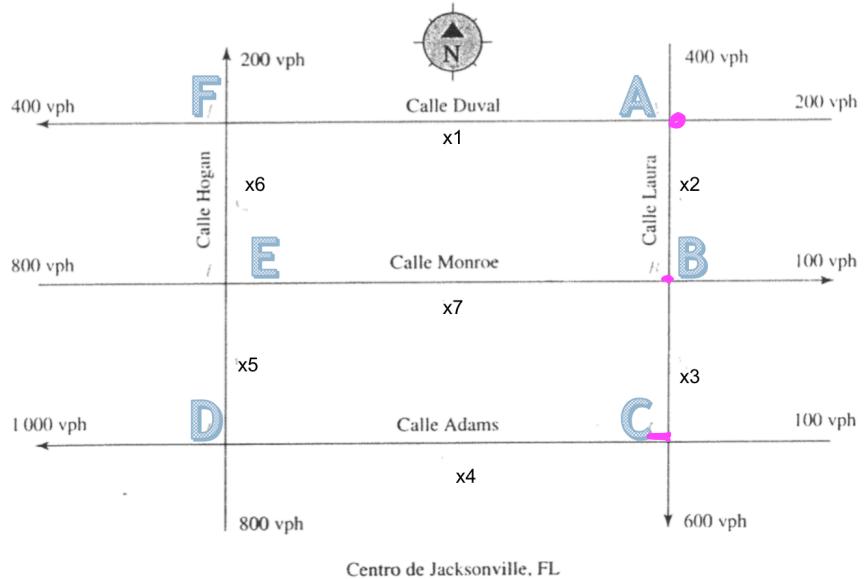
Por ejemplo:

- Reordenar la matriz para reducir el número de cálculos.
- Partir la matriz para enviarla a distintas computadoras y parallelizar los cálculos.

Y si vemos las matrices dispersas en la practica...



- Considere una red de avenidas típica del centro de Jacksonville:
- Todas las calles son de un solo sentido
- El trafico se mide en vph
- Los números dados se basan en horas pico 7 a 9 a.m. y 4 a 6 p.m.



Entradas = Salidas

- A $X_1 + X_2 = 600$
- B $X_2 - X_3 + X_7 = 100$
- C $X_3 - X_4 = 500$
- D $X_4 - X_5 = 200$
- E $X_5 - X_6 + X_7 = -800$
- F $X_1 + X_6 = 600$

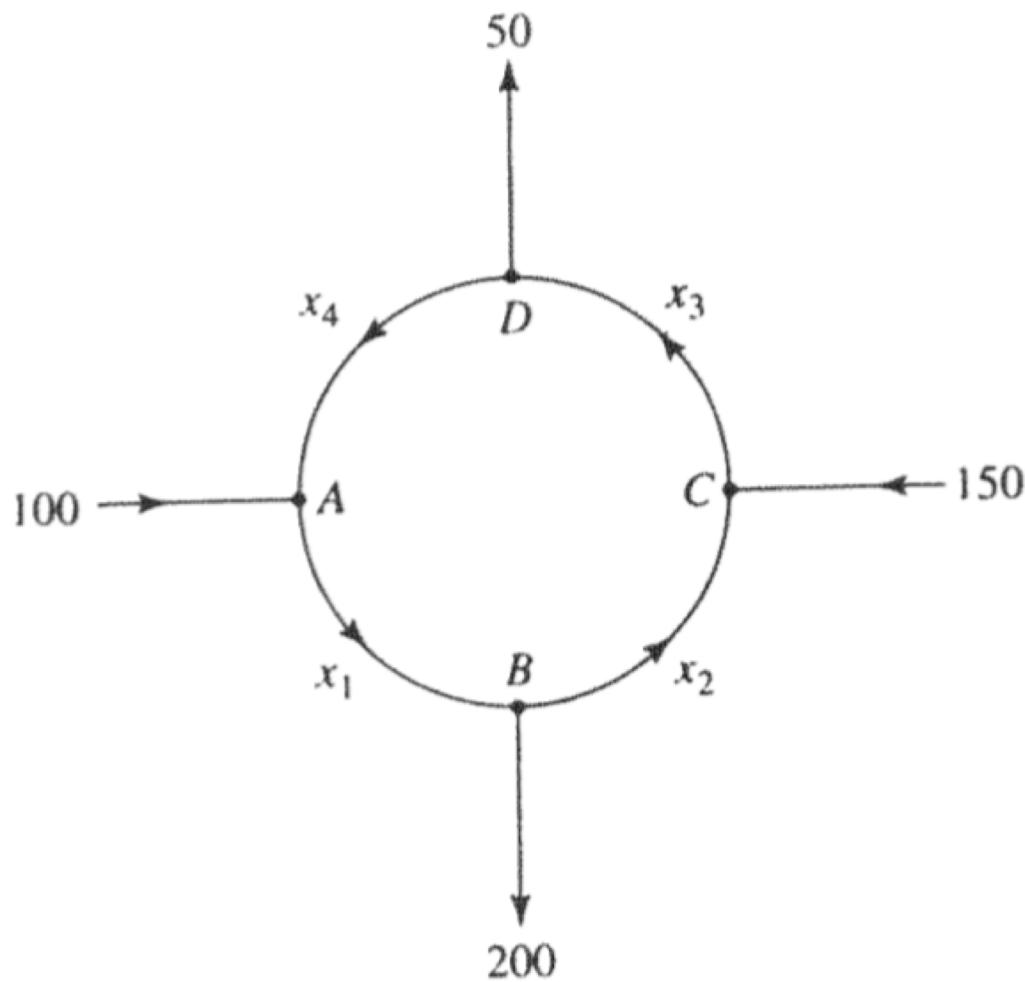
$$(6 \times 7) =$$

$$\begin{array}{ccccccc}
 1 & 1 & 0 & 0 & 0 & 0 & 0 = 600 \\
 0 & 1 & -1 & 0 & 0 & 0 & 1 = 100 \\
 0 & 0 & 1 & -1 & 0 & 0 & 0 = 500 \\
 0 & 0 & 0 & 1 & -1 & 0 & 0 = 200 \\
 0 & 0 & 0 & 0 & 1 & -1 & -1 = -800 \\
 1 & 0 & 0 & 0 & 0 & 1 & 0 = 600
 \end{array}$$

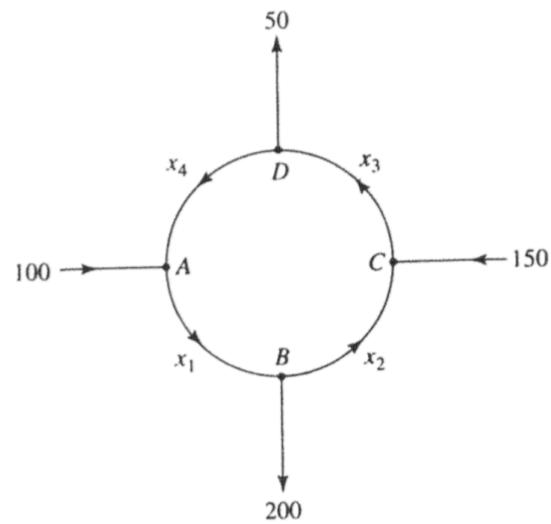
? de datos

Matriz dispersa





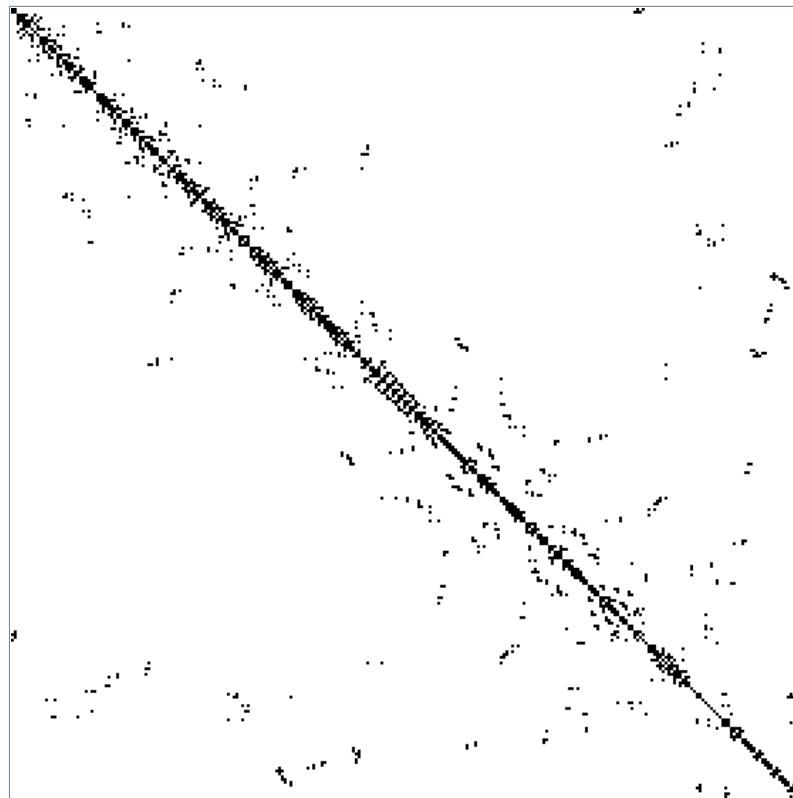
- ¿Qué pasa en el caso de una glorieta?
- ¿Cómo queda la matriz?

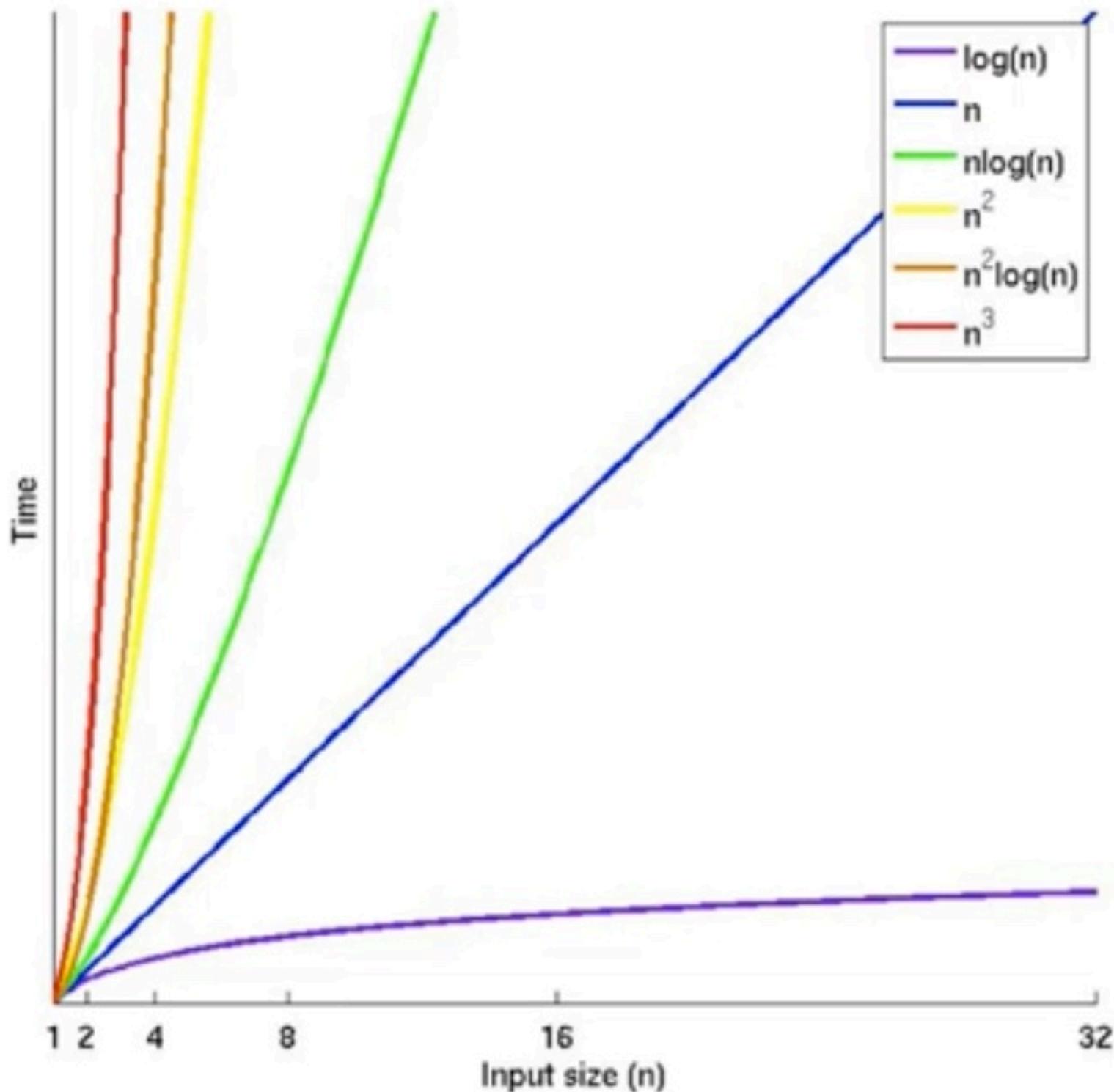


Almacenamiento de matrices dispersas

Las matrices llenas de tamaño $n \times n$ tienen un costo de almacenamiento de $O(n^2)$, las matrices dispersas suelen tener un costo de almacenamiento de $O(n)$.

Como ejemplo, la siguiente matriz $\mathbf{A} \in \mathbb{R}^{556 \times 556}$ contiene 309,136 entradas, con $\eta(\mathbf{A})=1810$, es decir sólo el 0.58% de las entradas son no cero.





Para ahorrar tanto memoria como tiempo de procesamiento sólo almacenaremos los elementos de la matriz \mathbf{A} que sean distintos de cero.

Hay varias estrategias de almacenamiento en memoria de matrices dispersas, dependiendo de la forma en que se accesarán las entradas.

Por coordenadas

Para una matriz dispersa \mathbf{A} de tamaño $m \times n$, las entradas diferentes de cero serán enumeradas con $k=1, 2, \dots, \eta(\mathbf{A})$.

Las entradas se alamacenan por medio de tres vectores \mathbf{V} , \mathbf{I} y \mathbf{J} . Estos contendrán el valor \mathbf{V}^k , el número de renglón \mathbf{I}^k y el número de columna \mathbf{J}^k de la k -ésima entrada no cero.

$$\begin{pmatrix} 8 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 0 & 0 \\ 2 & 0 & 1 & 0 & 7 & 0 \\ 0 & 9 & 3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix} \quad \begin{array}{cccccccccccc} 3 & 3 & 8 & 7 & 4 & 2 & 1 & 1 & 1 & 5 & 9 \\ 4 & 2 & 1 & 3 & 1 & 3 & 4 & 3 & 2 & 5 & 4 \\ 3 & 4 & 1 & 5 & 2 & 1 & 6 & 3 & 3 & 6 & 2 \end{array} \quad \begin{array}{l} \mathbf{V} \\ \mathbf{I} \\ \mathbf{J} \end{array}$$

Entre más grande sea el tamaño de la matriz más notorio será el ahorro en memoria.

Además de utilizar menos memoria, la otra gran diferencia con respecto a las matrices completas es que en las matrices dispersas las **entradas de la matriz deben ser buscadas**.

$$\begin{pmatrix} 8 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 0 & 0 \\ 2 & 0 & 1 & 0 & 7 & 0 \\ 0 & 9 & 3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix}$$

3	3	8	7	4	2	1	1	1	5	9
4	2	1	3	1	3	4	3	2	5	4
3	4	1	5	2	1	6	3	3	6	2

V I J

Por ejemplo ¿cuál es el valor de la entrada a_{46} ?

¿Cuál es el valor de la entrada a_{55} ?

¿Cuál es la primer entrada distinta de cero de la columna 6?

¿Cuántas entradas distintas de cero tiene el renglón 3?

El costo de búsqueda es muy elevado, ya que en el peor caso hay que buscar en todos las entradas.

Este método no es adecuado si se necesitan accesar todos los elementos de un renglón o columna de forma secuencial.

Se pueden ordenar los vectores (por renglón o por columna). Al ordenar se puede utilizar una búsqueda binaria para encontrar una entrada a_{ij} , ésto reduce el costo de la búsqueda a $\log_2 \eta(\mathbf{A})$.

$$\begin{pmatrix} 8 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 0 & 0 \\ 2 & 0 & 1 & 0 & 7 & 0 \\ 0 & 9 & 3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix}$$

8	2	4	9	1	1	3	3	7	1	5
1	3	1	4	2	3	4	2	3	4	5
1	1	2	2	3	3	3	4	5	6	6

V
I
J

Acceso a:	Costo máximo
Entrada a_{ij} (sin ordenar)	$\eta(\mathbf{A})$
Entrada a_{ij} (ordenando)	$\log_2 \eta(\mathbf{A})$

Este tipo de almacenamiento es adecuado cuando se lee de forma incremental las entradas la matriz. Este formato es muy utilizado para almacenar matrices dispersas en archivos.

Compressed Row Storage

El método *Compressed Row Storage* (compresión por renglones) [Saad03 p362], se guardan las entradas no cero de cada renglón de \mathbf{A} por separado.

$$\begin{pmatrix} 8 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 0 & 0 \\ 2 & 0 & 1 & 0 & 7 & 0 \\ 0 & 9 & 3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix}$$

8	4	
1	2	
1	3	
3	4	
2	1	7
1	3	5
9	3	1
2	3	6
5		
6		

$\mathbf{V}_3 = \{2, 1, 7\}$

$\mathbf{J}_3 = \{1, 3, 5\}$

Con este método, por cada renglón de la matriz se guardan dos arreglos para las entradas distintas de cero de cada renglón. Un vector \mathbf{J}_i conteniendo los índices y otro \mathbf{V}_i los valores, con $i=1, \dots, m$.

Este esquema es adecuado cuando todos (o casi todos) los renglones tienen al menos una entrada distinta de cero.

Este tipo de esquema se puede crear utilizando un vector de vectores.

¿Como cambia el costo de búsqueda en relación al almacenamiento por coordenadas?

$$\begin{pmatrix} 8 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 0 & 0 \\ 2 & 0 & 1 & 0 & 7 & 0 \\ 0 & 9 & 3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix}$$

8	4
1	2
1	3
3	4
2	1
1	3
9	3
2	3
5	
6	

$\mathbf{V}_3 = \{2, 1, 7\}$

$\mathbf{J}_3 = \{1, 3, 5\}$

¿Cuál es el valor de la entrada a_{46} ?

¿Cuál es el valor de la entrada a_{55} ?

¿Cuál es la primer entrada distinta de cero de la columna 6?

¿Cuántas entradas distintas de cero tiene el renglón 3?

Sea $\eta(\mathbf{J}_i)$ el número de entradas no cero del renglón i de \mathbf{A} .

Los cóstos de búsqueda de las entradas ahora son:

Acceso a:	Costo máximo
Entrada a_{ij}	$\eta(\mathbf{J}_i)$
Primer valor no cero de un renglón	1
j -ésimo valor no cero de un renglón	1
Primer valor no cero de una columna	$\sum_{i=1}^m \eta(\mathbf{J}_i) = \eta(\mathbf{A})$
i -ésimo valor no cero de una columna	$\sum_{i=1}^m \eta(\mathbf{J}_i) = \eta(\mathbf{A})$

Podemos reducir el costo de acceso ordenando por índice las entradas de cada renglón.

Así, al momento de accesar podremos utilizar un patrón búsqueda binaria.

Acceso a:	Costo máximo
Entrada $a_{i,j}$	$\log_2(\eta(\mathbf{J}_i))$
Primer valor no cero de un renglón	1
j -ésimo valor no cero de un renglón	1
Primer valor no cero de una columna	$\sum_{i=1}^m \log_2(\eta(\mathbf{J}_i))$
i -ésimo valor no cero de una columna	$\sum_{i=1}^m \log_2(\eta(\mathbf{J}_i))$

Multiplicación matriz vector

Realizar la multiplicación matriz-vector utilizando compresión por renglones resulta muy fácil.

En la multiplicación matriz-vector $\mathbf{c} = \mathbf{A} \mathbf{b}$ el orden de búsqueda es $O(1)$, esto es porque no se hace una búsqueda de las entradas del rengón, se toman las entradas una tras otra.

Sea \mathbf{J}_i el conjunto de entradas no cero del renglón i de \mathbf{A} .

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{pmatrix} = \left(\begin{array}{cccccc} 8 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 0 & 0 \\ 2 & 0 & 1 & 0 & 7 & 0 \\ 0 & 9 & 3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{array} \right) \begin{pmatrix} 5 \\ 4 \\ 0 \\ 1 \\ 2 \\ 9 \end{pmatrix}$$

$$\begin{matrix} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \\ \hline 8 & 4 & & & & \\ 1 & 2 & & & & \\ 1 & 3 & & & & \\ 3 & 4 & & & & \\ 2 & 1 & 7 & & & \\ 1 & 3 & 5 & & & \\ 9 & 3 & 1 & & & \\ 2 & 3 & 6 & & & \\ \hline 5 & & & & & \\ 6 & & & & & \end{matrix} = \begin{matrix} 5 \\ 4 \\ 0 \\ 1 \\ 2 \\ 9 \end{matrix}$$

$$c_i = \sum_{j=1}^n a_{ij} b_j$$

$$c_i = \sum_{k \in \mathbf{J}_i} \mathbf{V}_i^k b_{\mathbf{J}_i^k}$$

La ventaja de utilizar compresión por renglones es que los datos de cada renglón de la matriz de rigidez son accesados en secuencia uno tras otro, esto producirá una ventaja de acceso al entrar el bloque de memoria de cada renglón en el *cache* del CPU.

Compressed Column Storage

Análogamente a la compresión por renglones, existe el método *Compressed Column Storage* en el cual se almacena por columna en vez de por renglón.

$$\begin{pmatrix} 8 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 0 & 0 \\ 2 & 0 & 1 & 0 & 7 & 0 \\ 0 & 9 & 3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix}$$

8	2	
1	3	
4	9	
1	4	
1	1	3
2	3	4
3		
2		
7		
3		
1	5	
4	5	

$$\mathbf{V}_3 = \{1, 1, 3\}$$

$$\mathbf{I}_3 = \{2, 3, 4\}$$

Por cada columna de la matriz se guardan dos arreglos para las entradas distintas de cero. Uno \mathbf{I}_j conteniendo los índices y otro \mathbf{V}_j los valores, con $j=1, \dots, n$.

Este esquema es adecuado cuando se quiere buscar entradas por columna y no por renglón.

Sea $\eta(\mathbf{I}_j)$ es el número de entradas no cero de la columna j de \mathbf{A} . Los costos de búsqueda de las entradas ahora son:

Acceso a:	Costo máximo
Entrada a_{ij}	$\eta(\mathbf{I}_j)$
Primer valor no cero de un renglón	$\sum_{j=1}^n \eta(\mathbf{I}_j) = \eta(\mathbf{A})$
j -ésimo valor no cero de un renglón	$\sum_{j=1}^n \eta(\mathbf{I}_j) = \eta(\mathbf{A})$
Primer valor no cero de una columna	1
i -ésimo valor no cero de una columna	1

Este esquema es útil cuando se van a buscar entradas por columna y no por renglón. Por ejemplo, al multiplicar dos matrices dispersas.

Sean $\mathbf{A} \in \mathbb{R}^{m \times p}$ y $\mathbf{B} \in \mathbb{R}^{p \times n}$, la multiplicación $\mathbf{C} = \mathbf{A} \mathbf{B}$,

$$c_{ij} = \sum_{k=0}^p a_{ik} b_{kj}$$

es conveniente almacenar \mathbf{A} utilizando compresión por renglones y \mathbf{B} con compresión por columnas. El acceso de las entradas de ambas matrices dispersas será conveniente para el cache.

Por ejemplo:

$$\begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \\ c_{51} & c_{52} & c_{53} & c_{54} \end{pmatrix} = \begin{pmatrix} 8 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 0 & 0 \\ 2 & 0 & 1 & 0 & 7 & 0 \\ 0 & 9 & 3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 3 & 0 & 4 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \\ c_{51} & c_{52} & c_{53} & c_{54} \end{pmatrix} = \begin{pmatrix} 8 & 4 \\ 1 & 2 \\ 1 & 3 \\ 3 & 4 \\ 2 & 1 & 7 \\ 1 & 3 & 5 \\ 9 & 3 & 1 \\ 2 & 3 & 6 \\ 5 \\ 6 \end{pmatrix} \begin{pmatrix} 3 & 1 & 2 \\ 2 & 3 & 5 \\ 1 & 1 \\ 1 & 4 \\ 4 & 7 \\ 2 & 6 \\ 1 & 1 \\ 4 & 5 \end{pmatrix}$$

Nota importante: En el formato de matrices dispersas, las entradas deben estar ordenadas, primero por el índice de columna y segundo por el índice de renglón.

Por ejemplo, la siguiente matriz rala:

$$\begin{pmatrix} 8 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 0 & 0 \\ 2 & 0 & 1 & 0 & 7 & 0 \\ 0 & 9 & 3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix}$$

3	3	8	7	4	2	1	1	1	5	9
4	2	1	3	1	3	4	3	2	5	4
3	4	1	5	2	1	6	3	3	6	2

V I J

Para poder ser leída correctamente por MatLab hay que reordenar las entradas en base a los índices de columna y renglón antes de guardarla. De la siguiente forma:

$$\begin{pmatrix} 8 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 0 & 0 \\ 2 & 0 & 1 & 0 & 7 & 0 \\ 0 & 9 & 3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix}$$

8	2	4	9	1	1	3	3	7	1	5
1	3	1	4	2	3	4	2	3	4	5
1	1	2	2	3	3	3	4	5	6	6

V I J

Octave es menos riguroso con el orden de los índices, pero por compatibilidad es mejor ordenarlos.

Matrices dispersas en python

```
# Almacenamiento por coordenadas
I = array([0,3,1,0])
J = array([0,3,1,2])
V = array([4,5,7,9])
A = sparse.coo_matrix((V,(I,J)),shape=(4,4))
print(A)
```

Otros códigos de utilidad:

```
A.setdiag(rand(4))
A = A.tocsr()
C = A.toarray()
C = A.todense()
```



Matrices dispersas en python

Sparse matrix classes

<code>bsr_matrix(arg1[, shape, dtype, copy, blocksize])</code>	Block Sparse Row matrix
<code>coo_matrix(arg1[, shape, dtype, copy])</code>	A sparse matrix in COOrdinate format.
<code>csc_matrix(arg1[, shape, dtype, copy])</code>	Compressed Sparse Column matrix
<code>csr_matrix(arg1[, shape, dtype, copy])</code>	Compressed Sparse Row matrix
<code>dia_matrix(arg1[, shape, dtype, copy])</code>	Sparse matrix with DIAGONAL storage
<code>dok_matrix(arg1[, shape, dtype, copy])</code>	Dictionary Of Keys based sparse matrix.
<code>lil_matrix(arg1[, shape, dtype, copy])</code>	Row-based linked list sparse matrix
<code>spmatrix([maxprint])</code>	This class provides a base class for all sparse matrices.



¿Preguntas?

Referencias

- [Piss84] S. Pissanetzky. *Sparse Matrix Technology*. Academic Press, 1984.
- [Saad03] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2003.