ICNPG 2023

Clase 9:

Álgebra lineal Densa, Rala y Grafos...

Instituto Balseiro

Algebra Lineal Numérica

- **CUFFT**: Una transformación lineal particular.
- **CUBLAS**: operaciones tipo I, II y III, optimizado para formatos de matriz densa.
- **NVBLAS**: si tu viejo programa C, octave, etc, usa BLAS, lo acelerás recompilando.
- **CUSPARSE**: operaciones de CUBLAS optimizadas para formatos ralos, etc.
- **CUSOLVER**: Resolución de Sistemas lineales, Autovalores, factorizaciones, etc.
- **NVGRAPH/CUGRAPH:** formatos y algoritmos para redes o grafos.
- **CUPY**: Python interfaces to many of the functions in the CUDA device/runtime, CUBLAS, CUFFT, and CUSOLVER.
- **MAGMA**: Como CUSOLVER+CUBLAS+CUSPARSE pero pensada para sistemas híbridos CPU/GPU.
- **Array-Fire**: librería de C++ library paralela con una API sencilla, para cpu, cuda y opencl!

A Common Library Workflow

Many CUDA libraries share concepts, features, and a common workflow when being called from a host application. A common workflow while using NVIDIA libraries is as follows:

1. Create a library-specific handle that manages contextual information useful for the library's operation.
2. Allocate device memory for inputs and outputs to the library function.
3. If inputs are not already in a library-supported format, convert them to be accessible by the library.
4. Populate the pre-allocated device memory with inputs in a supported format.
5. Configure the library computation to be executed.
6. Execute a library call that offloads the desired computation to the GPU.
7. Retrieve the results of that computation from device memory, possibly in a library-determined format.
8. If necessary, convert the retrieved data to the application's native format.
9. Release CUDA resources.
10. Continue with the remainder of the application.

CuBLAS: Basic Linear Algebra Subprograms

- **Operaciones (números reales o complejos):**
  - Level I: functions that perform scalar and vector based operations.
  - Level II: functions that perform matrix-vector operations.
  - Level III: functions that perform matrix-matrix operations.
- **Cublas API**: Mas control, pero hay que manejar memoria del host y del device.
- **CublasXt API:** Menos control, pero solo hace falta manejar memoria del host.
- **Paralelismo:** soporta streams y multi-gpu, útiles para "batchear".
- **Compilación**:
  - nvcc myCublasApp.c -lcublas -o myCublasApp
  - g++ myCublasApp.c -lcublas_static -lculibos -lcudart_static -lpthread -ldl -I <cuda-toolkit-path>/include -L <cuda-toolkit-path>/lib64 -o myCublasApp
- **Ejemplo (level III)**: /share/apps/icnpg/clases/Cuda_Basico/MultMat/soluciones/multmat_solucion.cu

Multiplicacion de Matrices de 1024x1024 en nuestro cluster

naive CPU:	*6023.591623 ms*	*(1x)*
naive GPU:	*120.343554 ms*	*(50x)*
CUBLASXT:	*17.587757 ms*	*(300x)*
CUBLAS:	*9.224054 ms*	*(600x)*

https://docs.nvidia.com/cuda/cublas/index.html

CuBLAS levels

scalar and vector based operations

▽ 2.5. cuBLAS Level-1 Function Reference

- 2.5.1. cublasI<t>amax()
- 2.5.2. cublasI<t>amin()
- 2.5.3. cublas<t>asum()
- 2.5.4. cublas<t>axpy()
- 2.5.5. cublas<t>copy()
- 2.5.6. cublas<t>dot()
- 2.5.7. cublas<t>nrm2()
- 2.5.8. cublas<t>rot()
- 2.5.9. cublas<t>rotg()
- 2.5.10. cublas<t>rotm()
- 2.5.11. cublas<t>rotmg()
- 2.5.12. cublas<t>scal()
- 2.5.13. cublas<t>swap()

matrix-vector operations.

▽ 2.6. cuBLAS Level-2 Function Reference

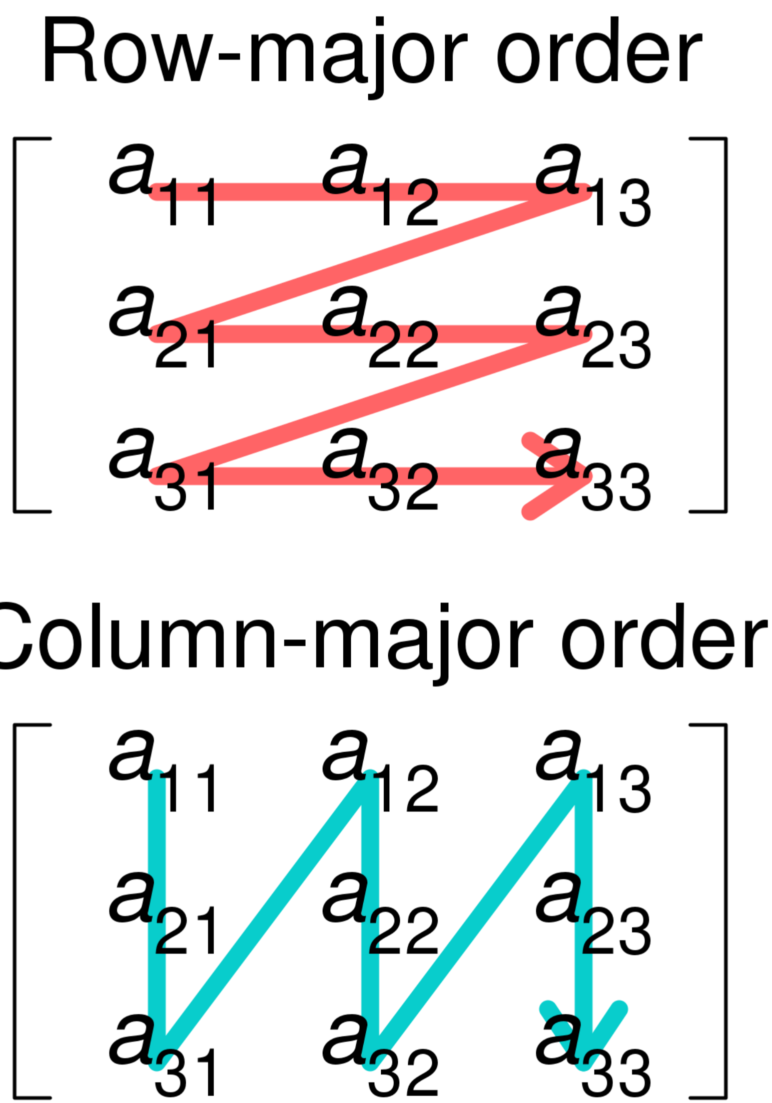
- 2.6.1. cublas<t>gbmv()
- 2.6.2. cublas<t>gemv()
- 2.6.3. cublas<t>ger()
- 2.6.4. cublas<t>sbmv()
- 2.6.5. cublas<t>spmv()
- 2.6.6. cublas<t>spr()
- 2.6.7. cublas<t>spr2()
- 2.6.8. cublas<t>symv()
- 2.6.9. cublas<t>syr()
- 2.6.10. cublas<t>syr2()
- 2.6.11. cublas<t>tbmv()
- 2.6.12. cublas<t>tbsv()
- 2.6.13. cublas<t>tpmv()
- 2.6.14. cublas<t>tpsv()
- 2.6.15. cublas<t>trmv()
- 2.6.16. cublas<t>trsv()
- 2.6.17. cublas<t>hemv()
- 2.6.18. cublas<t>hbmv()
- 2.6.19. cublas<t>hpmv()
- 2.6.20. cublas<t>her()
- 2.6.21. cublas<t>her2()
- 2.6.22. cublas<t>hpr()
- 2.6.23. cublas<t>hpr2()

matrix-matrix operations.

▽ 2.7. cuBLAS Level-3 Function Reference

- 2.7.1. cublas<t>gemm()
- 2.7.2. cublas<t>gemm3m()
- 2.7.3. cublas<t>gemmBatched()
- 2.7.4. cublas<t>gemmStridedBatched()
- 2.7.5. cublas<t>symm()
- 2.7.6. cublas<t>syrk()
- 2.7.7. cublas<t>syr2k()
- 2.7.8. cublas<t>syrkx()
- 2.7.9. cublas<t>trmm()
- 2.7.10. cublas<t>trsm()
- 2.7.11. cublas<t>trsmBatched()
- 2.7.12. cublas<t>hemm()
- 2.7.13. cublas<t>herk()
- 2.7.14. cublas<t>her2k()
- 2.7.15. cublas<t>herkx()

Data layout



1.1. Data layout

For maximum compatibility with existing Fortran environments, the cuBLAS library uses column-major storage, and 1-based indexing. Since C and C++ use row-major storage, applications written in these languages can not use the native array semantics for two-dimensional arrays. Instead, macros or inline functions should be defined to implement matrices on top of one-dimensional arrays. For Fortran code ported to C in mechanical fashion, one may chose to retain 1-based indexing to avoid the need to transform loops. In this case, the array index of a matrix element in row "i" and column "j" can be computed via the following macro

```
#define IDX2F(i,j,ld) ((((j)-1)*(ld))+((i)-1))
```

Here, ld refers to the leading dimension of the matrix, which in the case of column-major storage is the number of rows of the allocated matrix (even if only a submatrix of it is being used).

For natively written C and C++ code, one would most likely choose 0-based indexing, in which case the array index of a matrix element in row "i" and column "j" can be computed via the following macro

```
#define IDX2C(i,j,ld) (((j)*(ld))+(i))
```

CuSPARSE: cuBLAS para matrices ralas

The cuSPARSE library contains a set of GPU-accelerated basic linear algebra subroutines used for handling sparse matrices that perform significantly faster than CPU-only alternatives. Depending on the specific operation, the library targets matrices with sparsity ratios in the range between 70%-99.9%.

- Operations between a **sparse vector** and a **dense vector**: sum, dot product, scatter, gather
  - Operations between a **dense matrix** and a **sparse vector**: multiplication
  - Operations between a **sparse matrix** and a **dense vector**: multiplication, triangular solver, tridiagonal solver, pentadiagonal solver
  - Operations between a **sparse matrix** and a **dense matrix**: multiplication, triangular solver, tridiagonal solver, pentadiagonal solver
  - Operations between a **sparse matrix** and a **sparse matrix**: sum, multiplication
  - Operations between **dense matrices** with output a **sparse matrix**: multiplication
  - **Sparse matrix preconditioners**: Incomplete Cholesky Factorization (level 0), Incomplete LU Factorization (level 0)
  - Reordering and Conversion operations between different **sparse matrix storage formats**

- **Formateo**: operations that allow conversion between different matrix formats, and compression of **csr** matrices.
- **Compilación:**
  - nvcc myCusparseApp.c -lcusparse -o myCusparseApp
  - g++ myCusparseApp.c -lcusparse_static -lculibos -lcudart_static -lpthread -ldl -I <cuda-toolkit-path>/include -L <cuda-toolkit-path>/lib64 -o myCusparseApp
- **Ejemplos:** copiar del toolkit...

https://docs.nvidia.com/cuda/cusparse/index.html

CuSPARSE: Formatos de vectores

- Vectores Densos

1.0 0.0 0.0 2.0 3.0 0.0 4.0

single data array that is stored linearly in memory

- Vectores Ralos

1.0 2.0 3.0 4.0

The data array has the nonzero values from the equivalent array in dense format.

0 3 4 6

The integer index array has the positions of the corresponding nonzero values in the equivalent array in dense format.

CuSPARSE: Formatos de Matrices

- Matrices Densas

$$\begin{matrix} X_{1,1} & X_{1,2} & \cdots & X_{1,n} \\ X_{2,1} & X_{2,2} & \cdots & X_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ X_{m,1} & X_{m,2} & \cdots & X_{m,n} \\ \vdots & \vdots & \ddots & \vdots \\ X_{ldX,1} & X_{ldX,2} & \cdots & X_{ldX,n} \end{matrix}$$

The dense matrix X is assumed to be stored in **column-major format in memory**

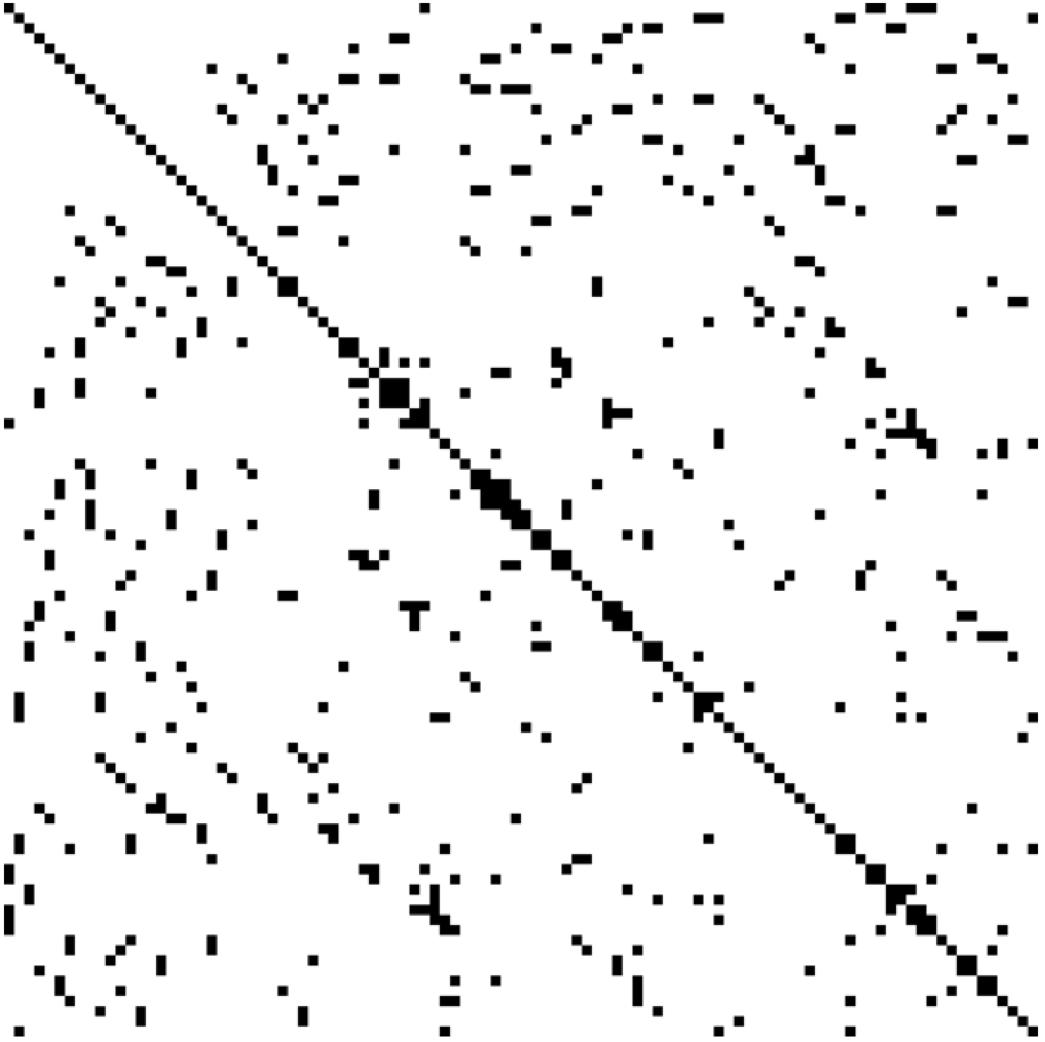
$$X_{1,1} \quad X_{2,1} \quad \cdots \quad X_{m,1} \quad \cdots \quad X_{ldX,1} \quad \cdots \quad X_{1,n} \quad X_{2,n} \quad \cdots \quad X_{m,n} \quad \cdots \quad X_{ldX,n}$$

and is represented by the following parameters.

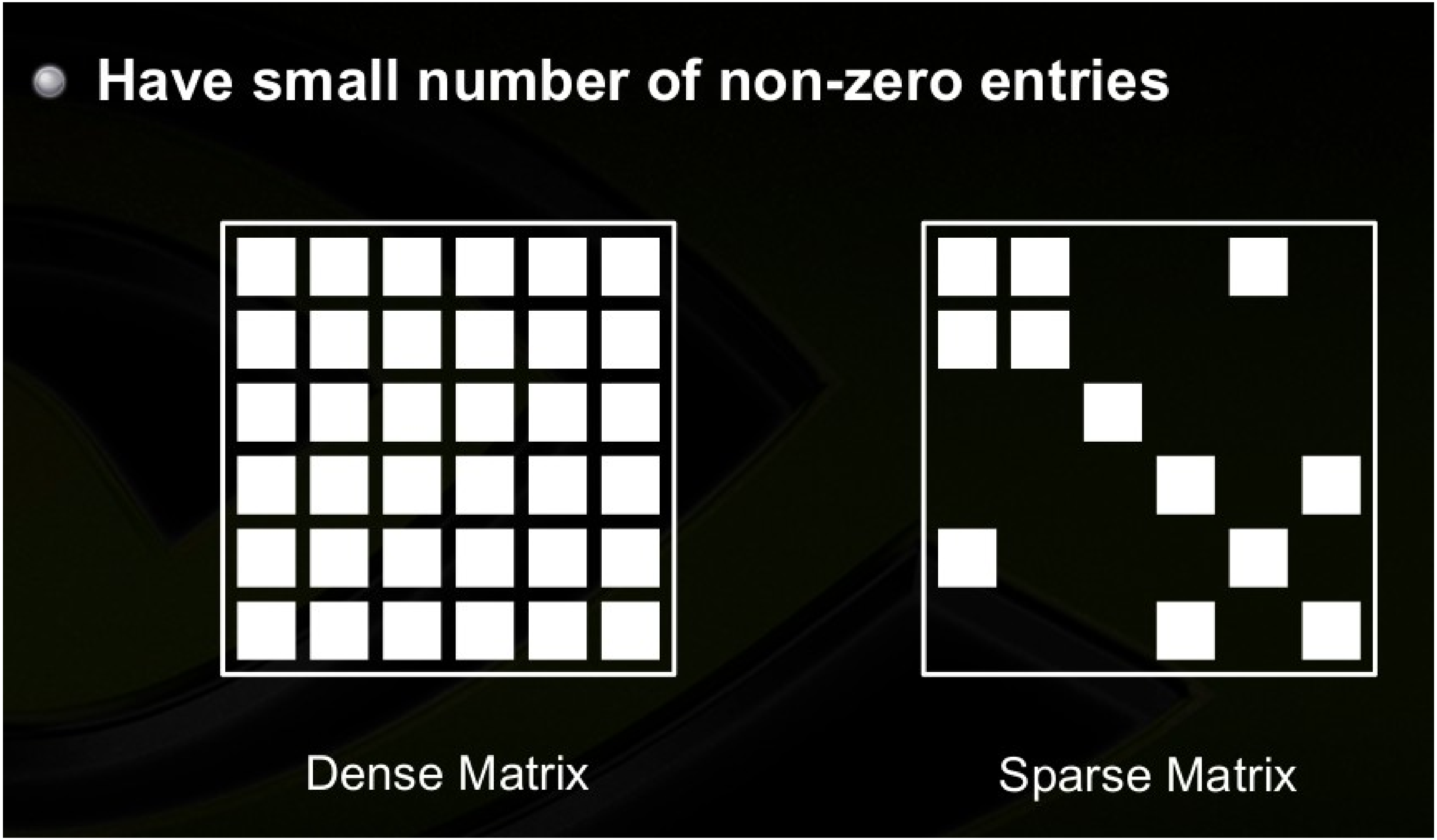
`m`	(integer)	The number of rows in the matrix.
`n`	(integer)	The number of columns in the matrix.
`ldX`	(integer)	The leading dimension of `X`, which must be greater than or equal to `m`. If `ldX` is greater than `m`, then `X` represents a sub-matrix of a larger matrix stored in memory
`X`	(pointer)	Points to the data array containing the matrix elements. It is assumed that enough storage is allocated for `X` to hold all of the matrix elements and that cuSPARSE library functions may access values outside of the sub-matrix, but will never overwrite them.

CuSPARSE: Formatos de Matrices

- Matrices Ralas

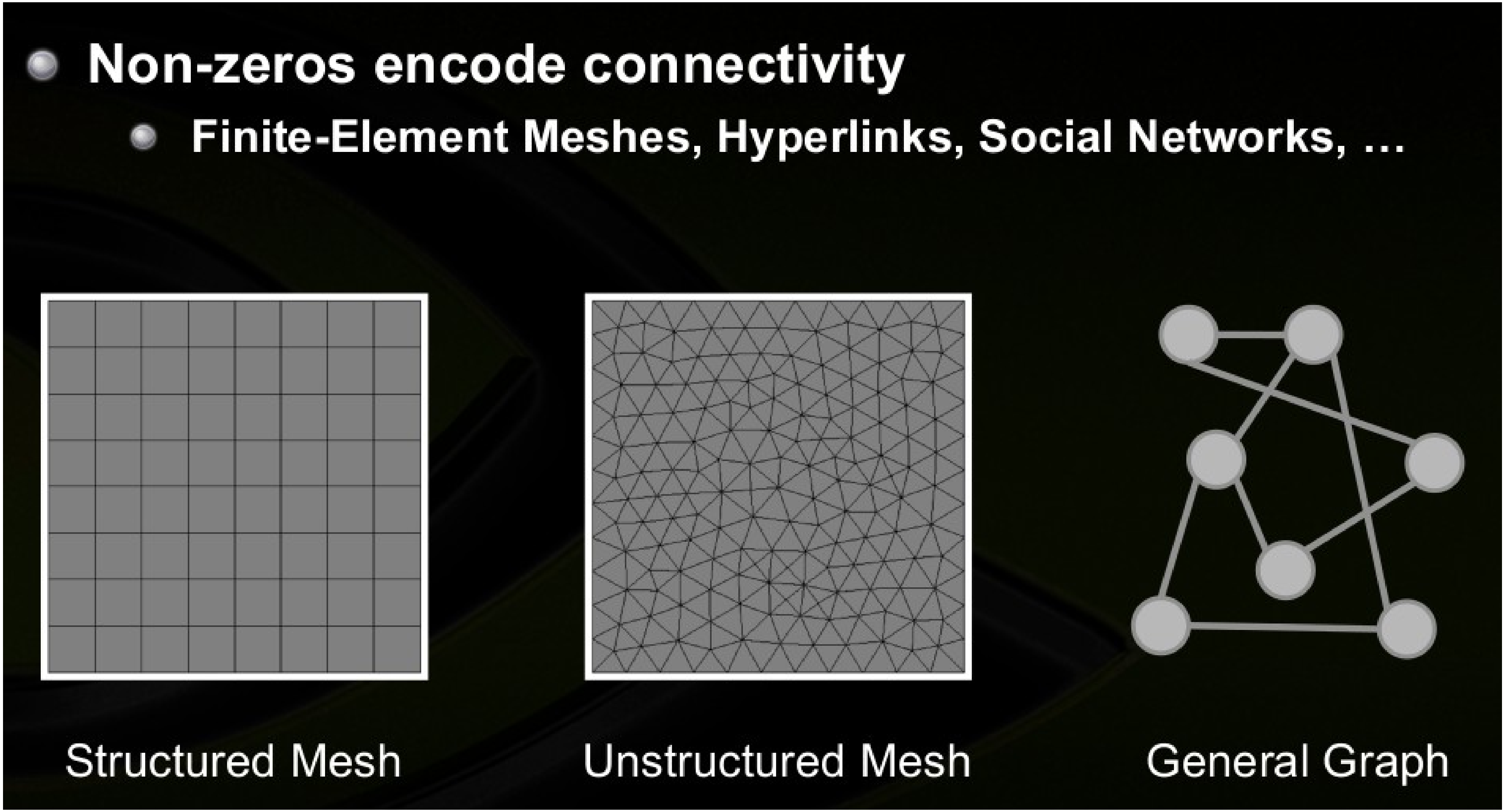


Que son las matrices ralas (sparse matrix)?



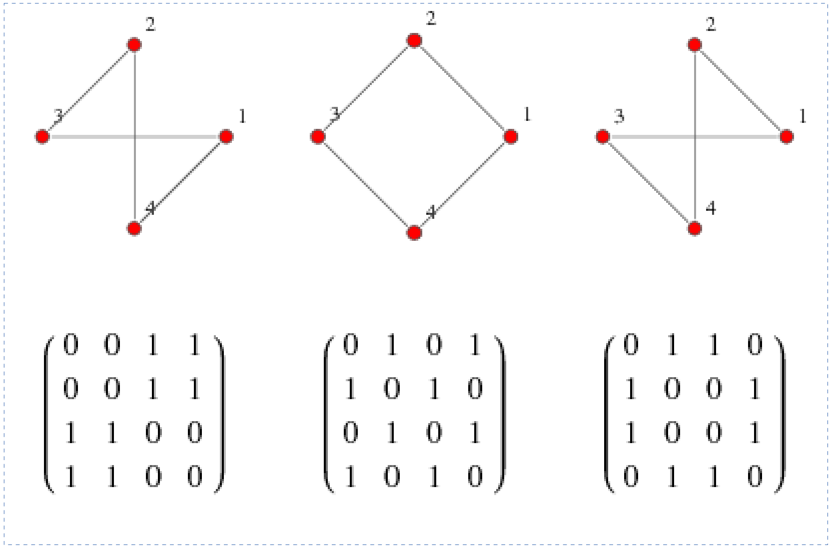
Nathan Bell, NVIDIA

Donde aparecen las matrices ralas?

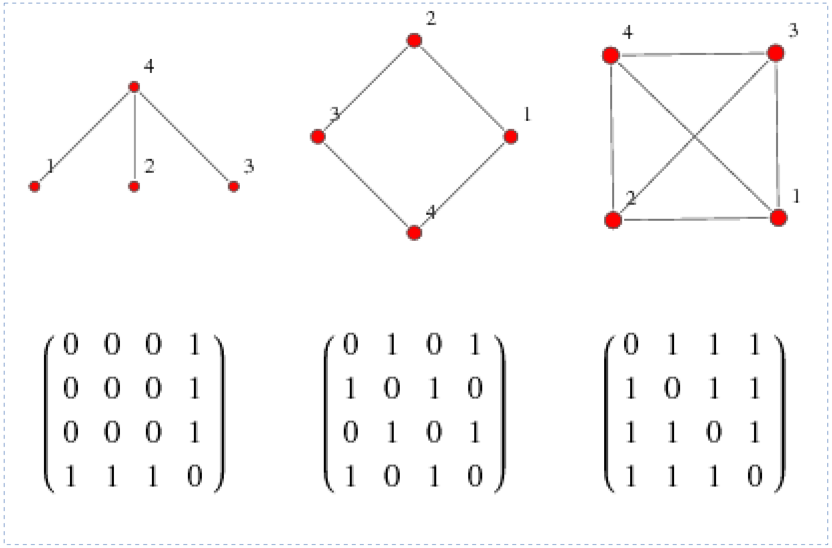


Nathan Bell, NVIDIA

Matriz de conectividad o adyacencia



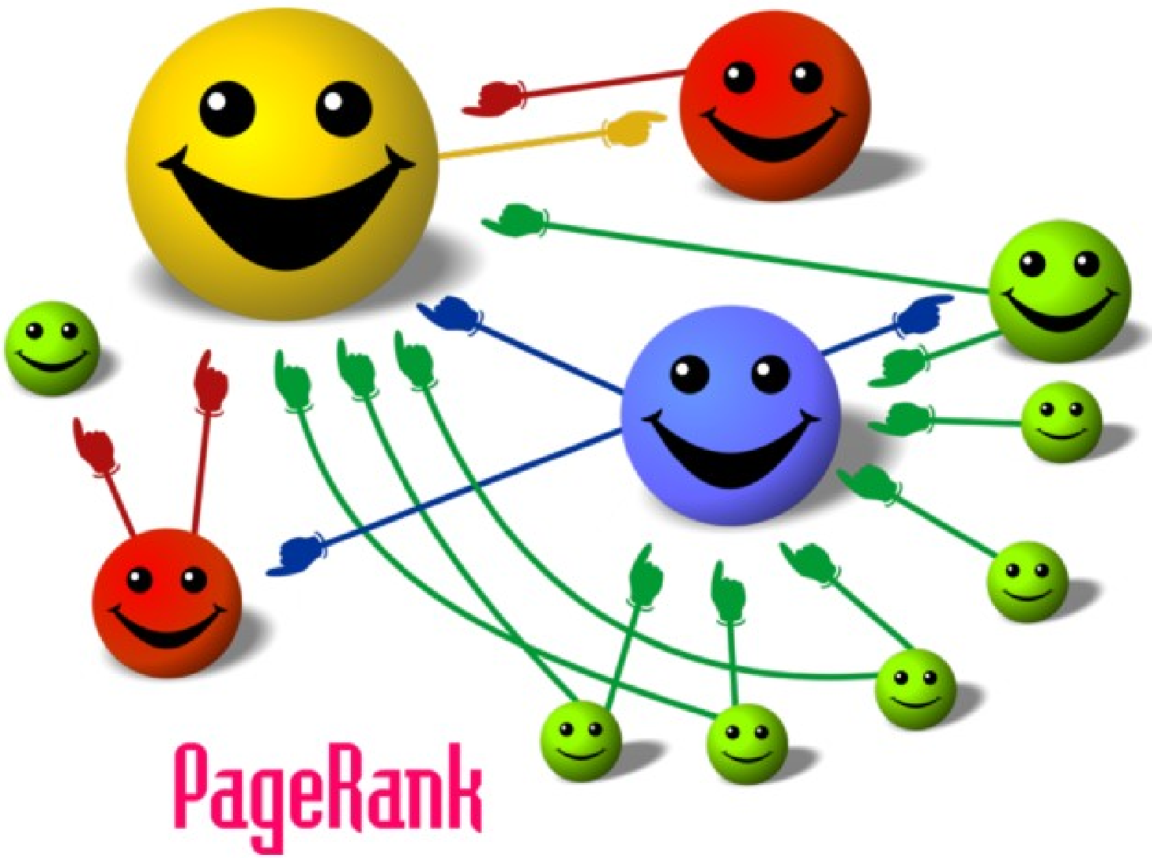
La matriz es basicamente una estructura de datos muy útil para representar y manipular grafos en programas de computadora.



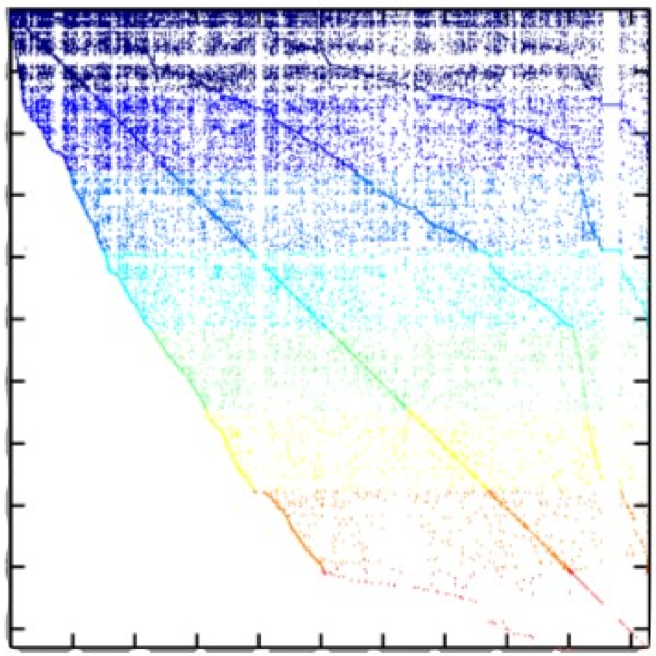
The adjacency matrix, sometimes also called the connection matrix, of a simple labeled graph is a matrix with rows and columns labeled by graph vertices, with a 1 or 0 in position according to whether and are adjacent or not. For a simple graph with no self-loops, the adjacency matrix must have 0s on the diagonal. For an undirected graph, the adjacency matrix is symmetric.

http://mathworld.wolfram.com/AdjacencyMatrix.html

Matrices: Page Rank (Brin, Page)



$A_{ij}=1$ si $i \longrightarrow j$



Ejercicio

- Mitre, Moreno, Elflein, Gallardo.
- Quaglia, Villegas, Rolando, Palacios.

- *Salgo de (A) Elflein y Palacios*
- *Llego a (B) Mitre y Quaglia*
- *¿camino mas corto?*
- *¿cuantos caminos de n cuadras hay?*
- *¿Cuantos caminos de hasta n cuadras?*
- *¡A pie no es igual que en auto!*
- *Etc.*

Matriz de conectividad en auto (2016!)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1																
2																
3																
4																
5																
6																
7																
8																
9																
10																
11																
12																
13																
14																
15																
16																

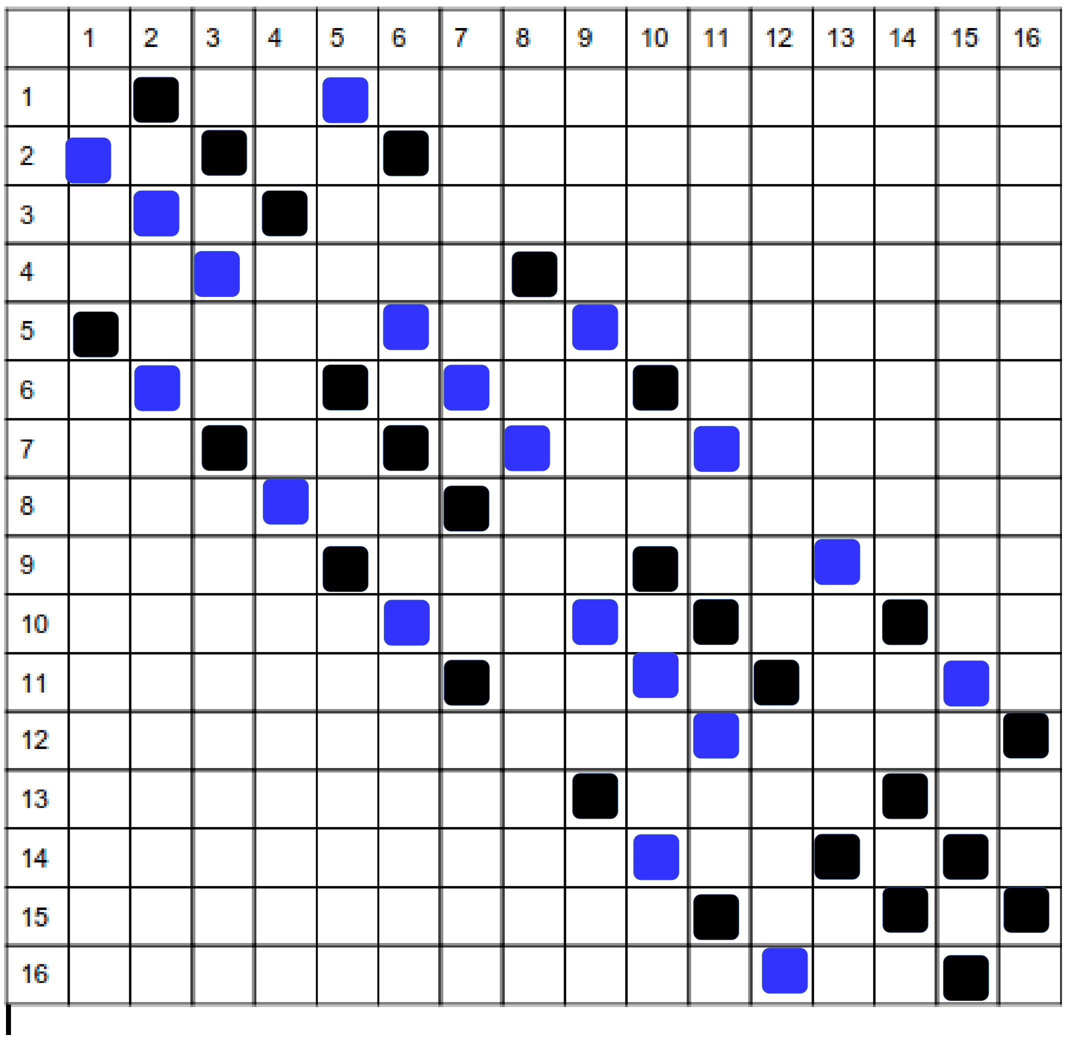
Matriz de conectividad en auto

No simétrica

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1		■														
2			■			■										
3				■												
4								■								
5	■															
6					■					■						
7			■			■										
8							■									
9					■					■						
10											■			■		
11							■					■				
12																■
13									■					■		
14													■		■	
15											■			■		■
16															■	

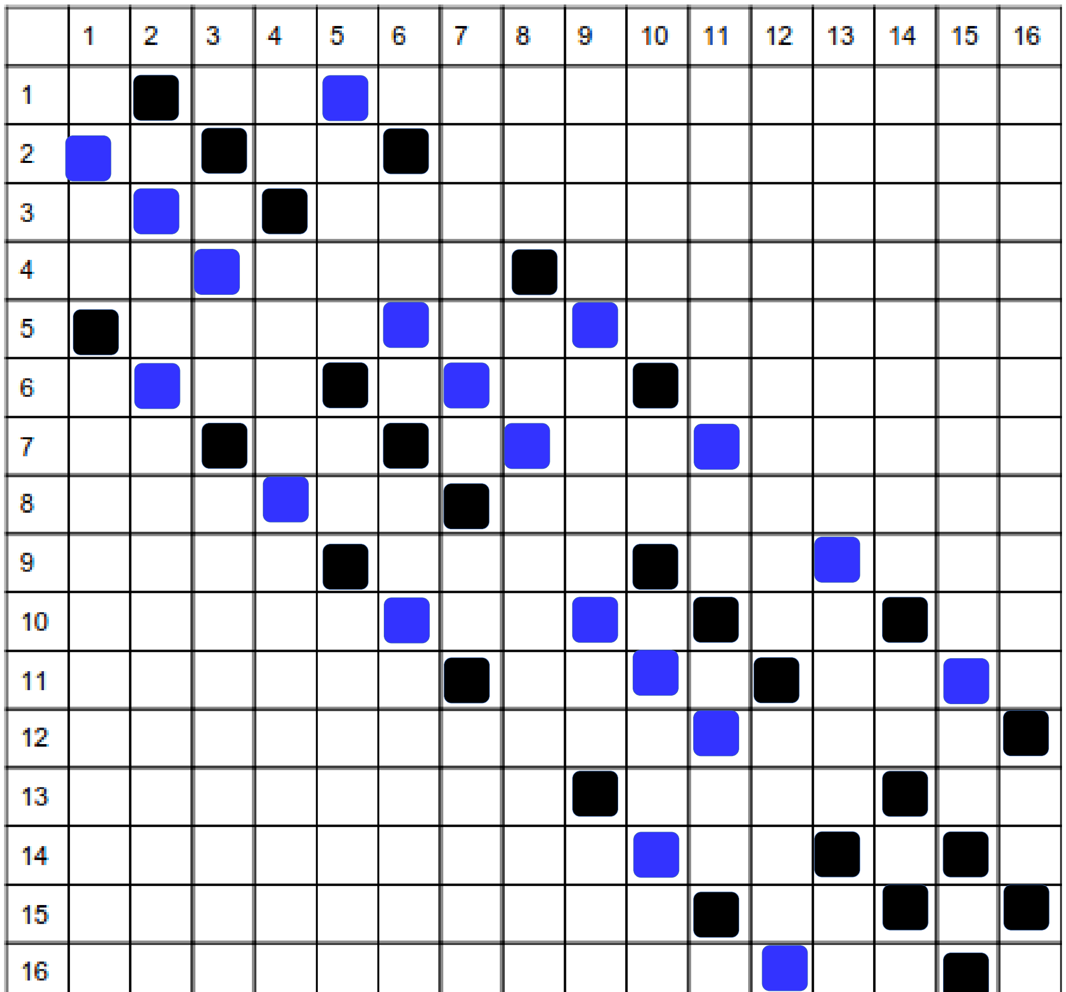
¿Matriz de conectividad a pie?

Simétrica



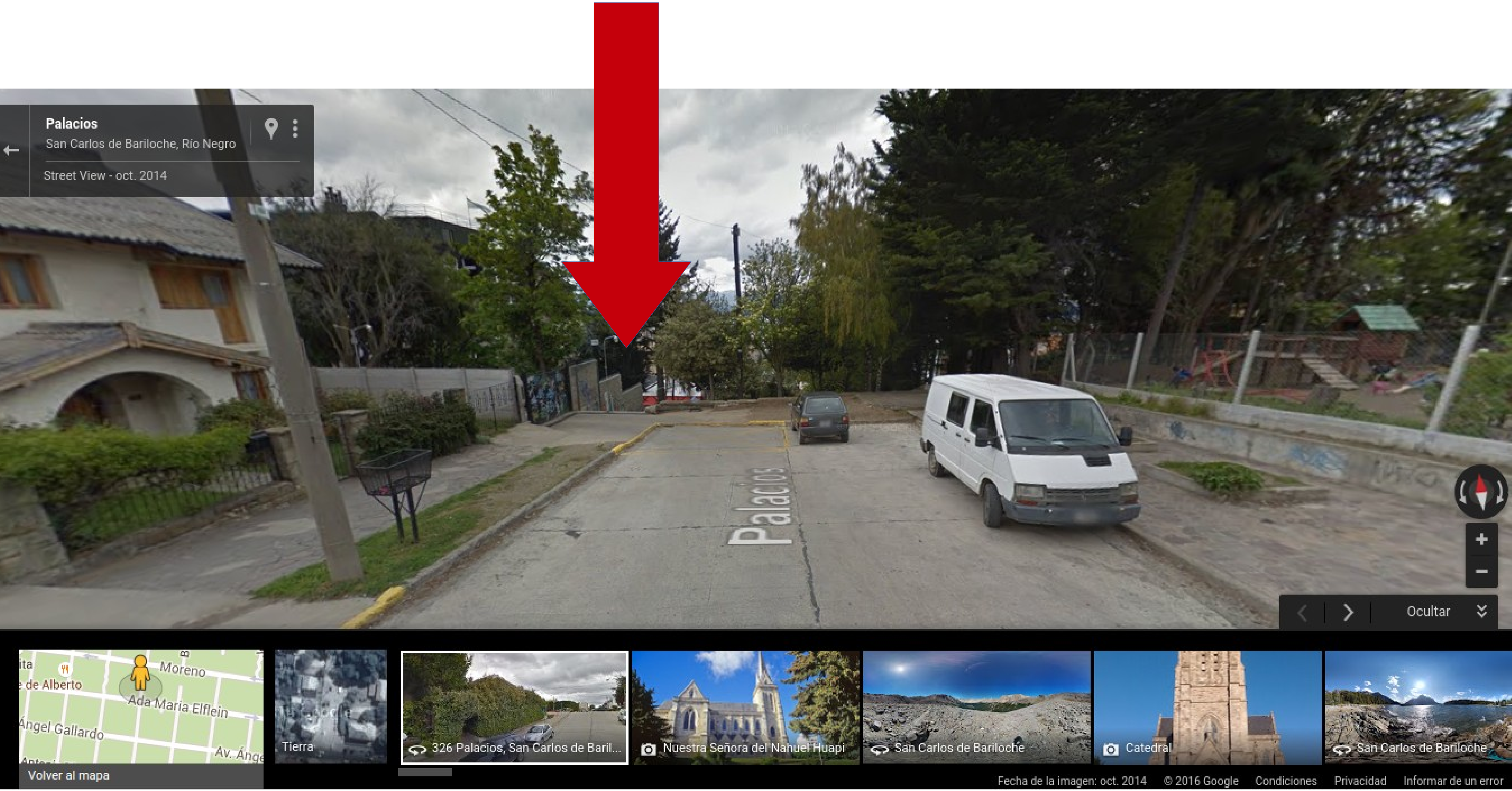
Test de vigilia ...

Simétrica



Conectividad a pie...


Moreno
Hotel Pacifico
Hotel Aguas del Sur
Moreno
Monteclaro
La Jirafa
Moreno
Rolando
Rolando
Hotel Patagonia Sur
da Maria Elflein
Ada Maria Elflein
Palacios
Ada Maria Elflein

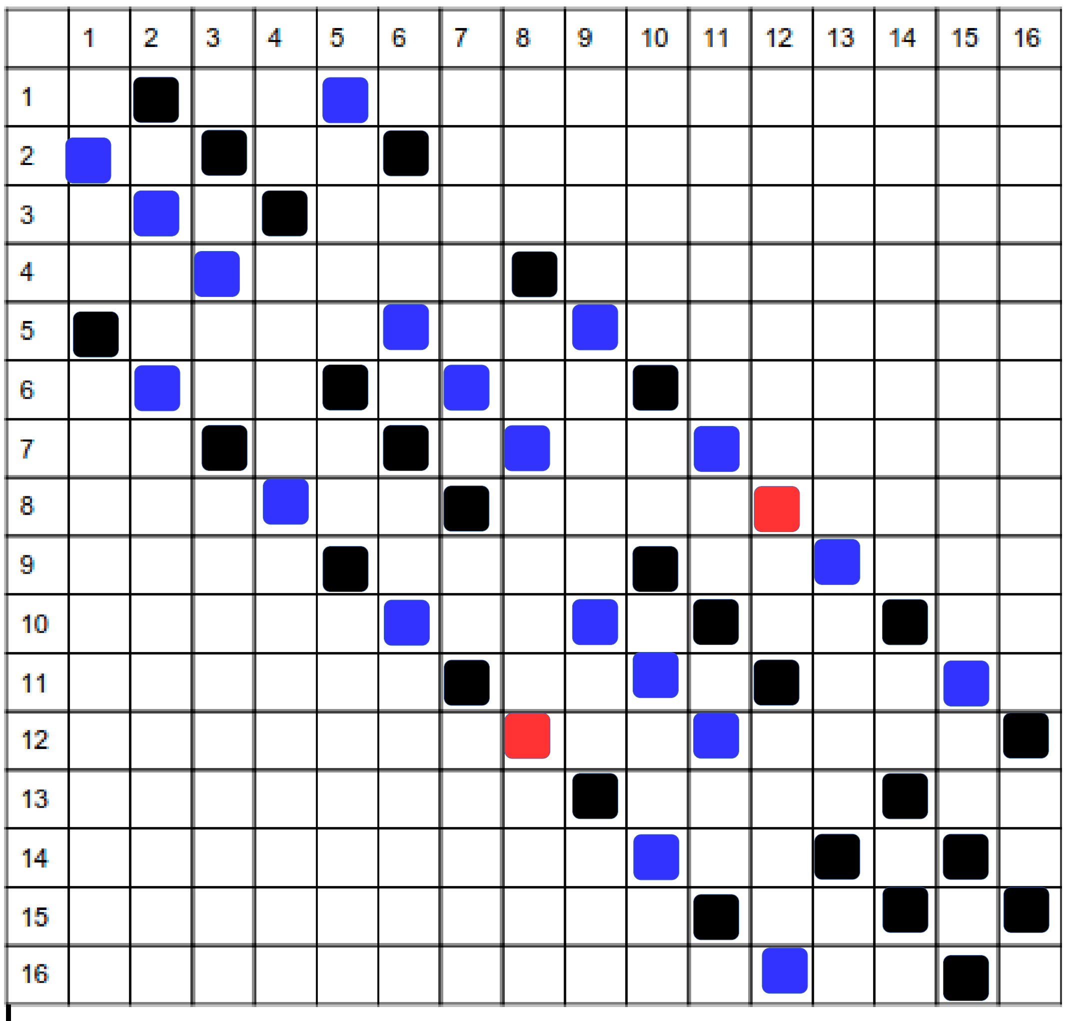


Palacios
San Carlos de Bariloche, Río Negro
Street View - oct. 2014
Palacios
Ocultar
Moreno
de Alberto
Ada María Elflein
Ángel Gallardo
Av. Ánge
Volver al mapa
Tierra
326 Palacios, San Carlos de Baril...
Nuestra Señora del Nahuel Huapi
San Carlos de Bariloche
Catedral
San Carlos de Bariloche
Fecha de la imagen: oct. 2014
© 2016 Google
Condiciones
Privacidad
Informar de un error

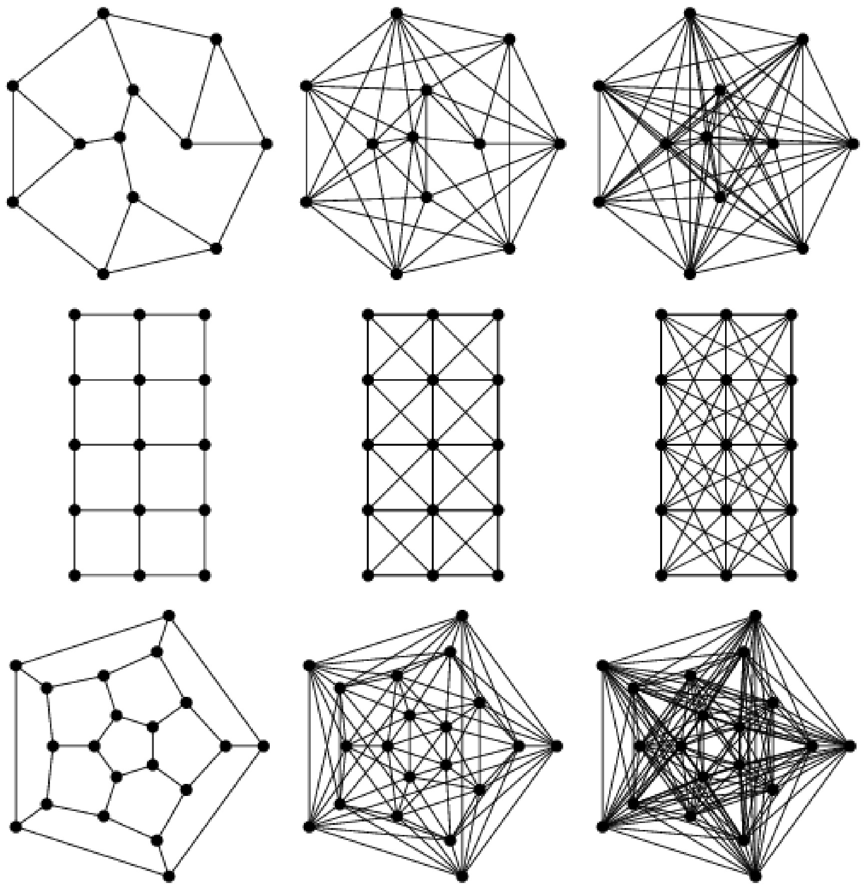
Moreno
San Carlos de Bariloche, Río Negro
Street View - dic. 2014
dosveintidos
DE LA CIUDAD
12 CUOTAS
La Baita
El Boliche de Alberto
Moreno
Ada María Elflein
Volver al mapa
Tierra
402 Moreno, San Carlos de Baril...
Añadir una foto
Ocultar
Fecha de la imagen: dic. 2014
© 2016 Google
Condiciones
Privacidad
Informar de un error

¿Matriz de conectividad a pie?

simétrica



Propiedades de la matriz de adyacencia



http://mathworld.wolfram.com/GraphPower.html

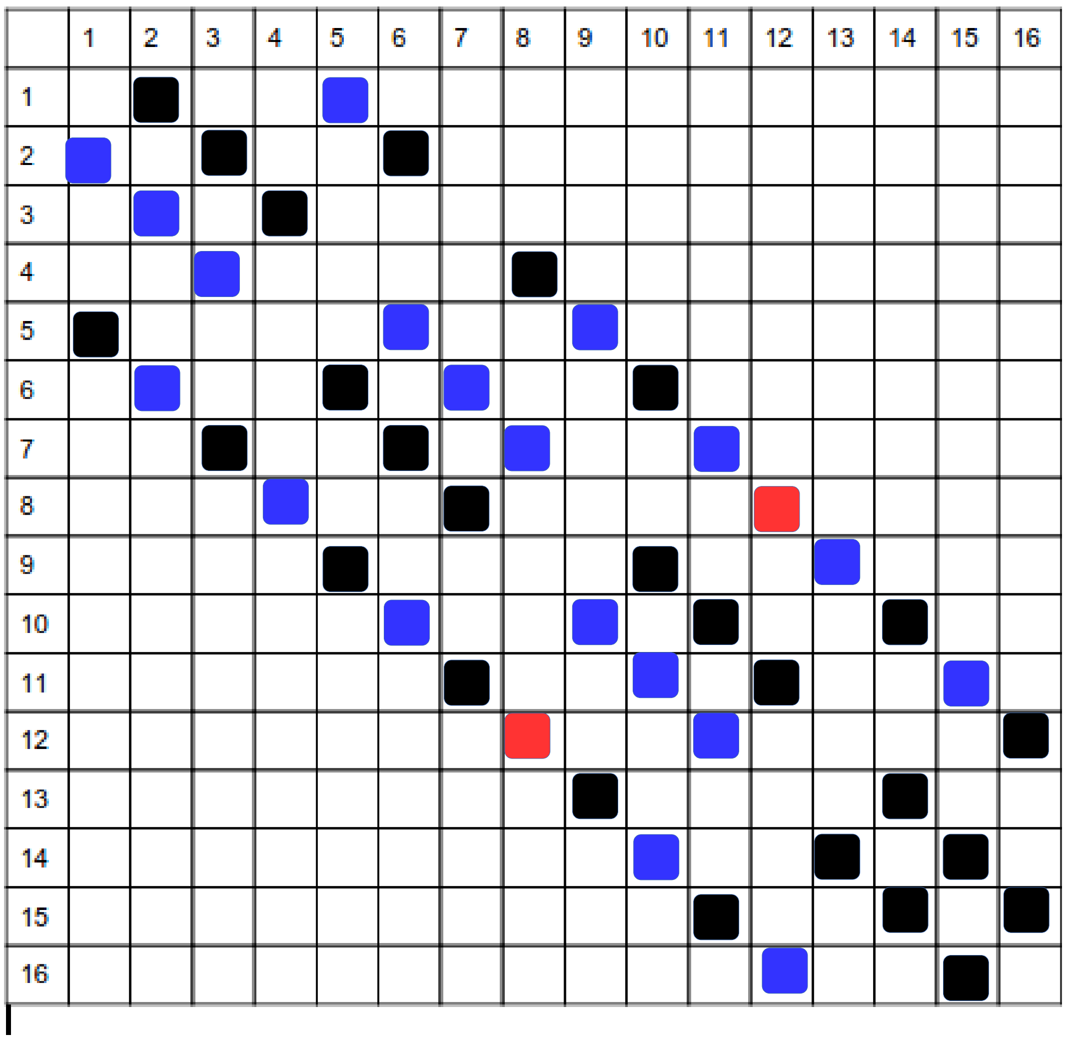
- The kth power of a graph G is a graph with the same set of vertices as G and an edge between two vertices iff there is a path of length at most k between them.
- Since a path of length two between vertices u and v exists for every vertex w such that {u,w} and {w,v} are edges in G, the square of the adjacency matrix of G counts the number of such paths.
- Similarly, the (u,v)th element of the kth power of the adjacency matrix of G gives the number of paths of length k between vertices u and v.
- The graph kth power is then defined as the graph whose adjacency matrix given by the sum of the first k powers of the adjacency matrix,

  adj(G^k)=sum_(i=1)^k[adj(G)]^i,

  which counts all paths of length up to k.

Caminos mas cortos, entre uno o mas puntos, o número de caminos de k pasos entre dos puntos, etc, etc.

Simétrica y rala (sparse)



Ecuación de Laplace

- Ecuación de difusión
- Ecuación del calor
- Membrana elástica
- Electrostática...
- Etc.

$$V = f_N(x)$$

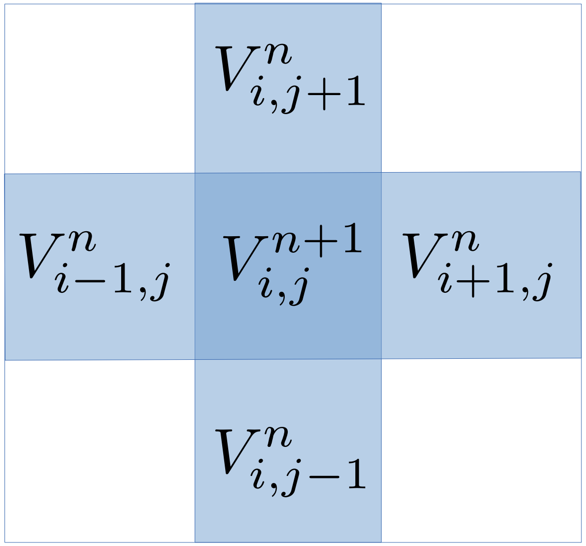
$$(\partial_x^2 + \partial_y^2)V = 0$$

$$V = f_O(y) \qquad V(x,y) =? \qquad V = f_E(y)$$

$$V = f_S(x)$$

Método de Jacobi

$$V_{i,j}^{n+1} = \frac{1}{4}(V_{i+1,j}^{n} + V_{i-1,j}^{n} + V_{i,j+1}^{n} + V_{i,j-1}^{n})$$



Método de Jacobi

$$V_{i,j}^{n+1} = \frac{1}{4}(V_{i+1,j}^{n} + V_{i-1,j}^{n} + V_{i,j+1}^{n} + V_{i,j-1}^{n})$$

En algebra lineal numérica toda matriz es un vector

$$\alpha(i,j) = i + jL = 0, 1, ..., L^2 - 1 \rightarrow V_{\alpha}^{n+1} = \sum_{\beta} A_{\alpha,\beta} V_{\beta}^{n}$$

Método de Jacobi

$$V_{i,j}^{n+1} = \frac{1}{4}(V_{i+1,j}^{n} + V_{i-1,j}^{n} + V_{i,j+1}^{n} + V_{i,j-1}^{n})$$

$$V_{\alpha}^{n+1} = \sum_{\beta} A_{\alpha,\beta} V_{\beta}^{n}$$

$$\mathbf{V^{n+1}} = \mathbf{A.V^{n}}$$

¿Cuantos elementos distintos de cero tiene A?

A matriz real de $L^2 \times L^2$ rala

Convolución o Correlación

$$y_i = \sum_k h_k x_{k+i} \to \mathbf{y} = \mathbf{H}.\mathbf{x}$$

¿Cuantos elementos distintos de cero tiene H si el filtro es local?

El filtro puede representar un operador diferencial discretizado

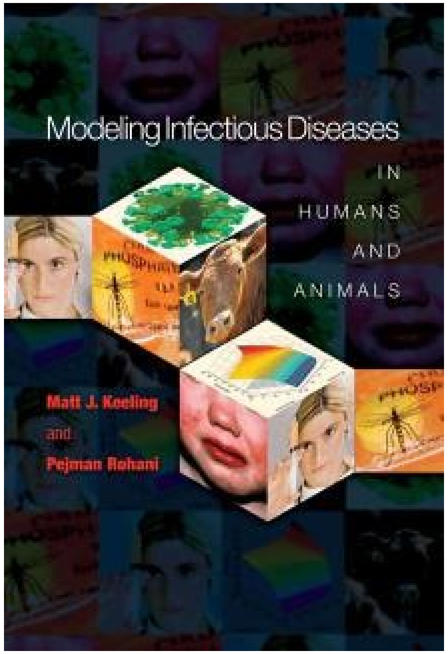
Por ejemplo en 1D

$$\mathbf{h} = \{1, -2, 1\}/\Delta x^2 \to$$

$$y_i = \sum_k h_k u_{k+i} = \frac{u_{i+1} + u_{i-1} - 2u_i}{\Delta x^2} \equiv \partial_x^2 u$$

En general los operadores diferenciales locales dan una H rala.

Ejemplo: epidemias

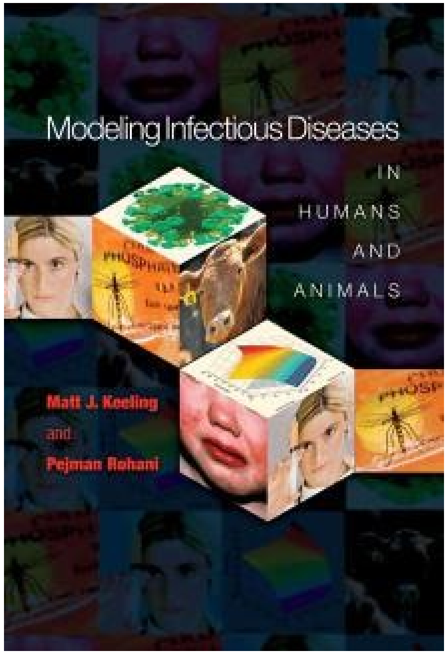


$$\frac{dS}{dt} = \mu - \beta SI - \mu S,$$

$$\frac{dI}{dt} = \beta SI - \gamma I - \mu I,$$

$$\frac{dR}{dt} = \gamma I - \mu R.$$

Ejemplo: epidemias

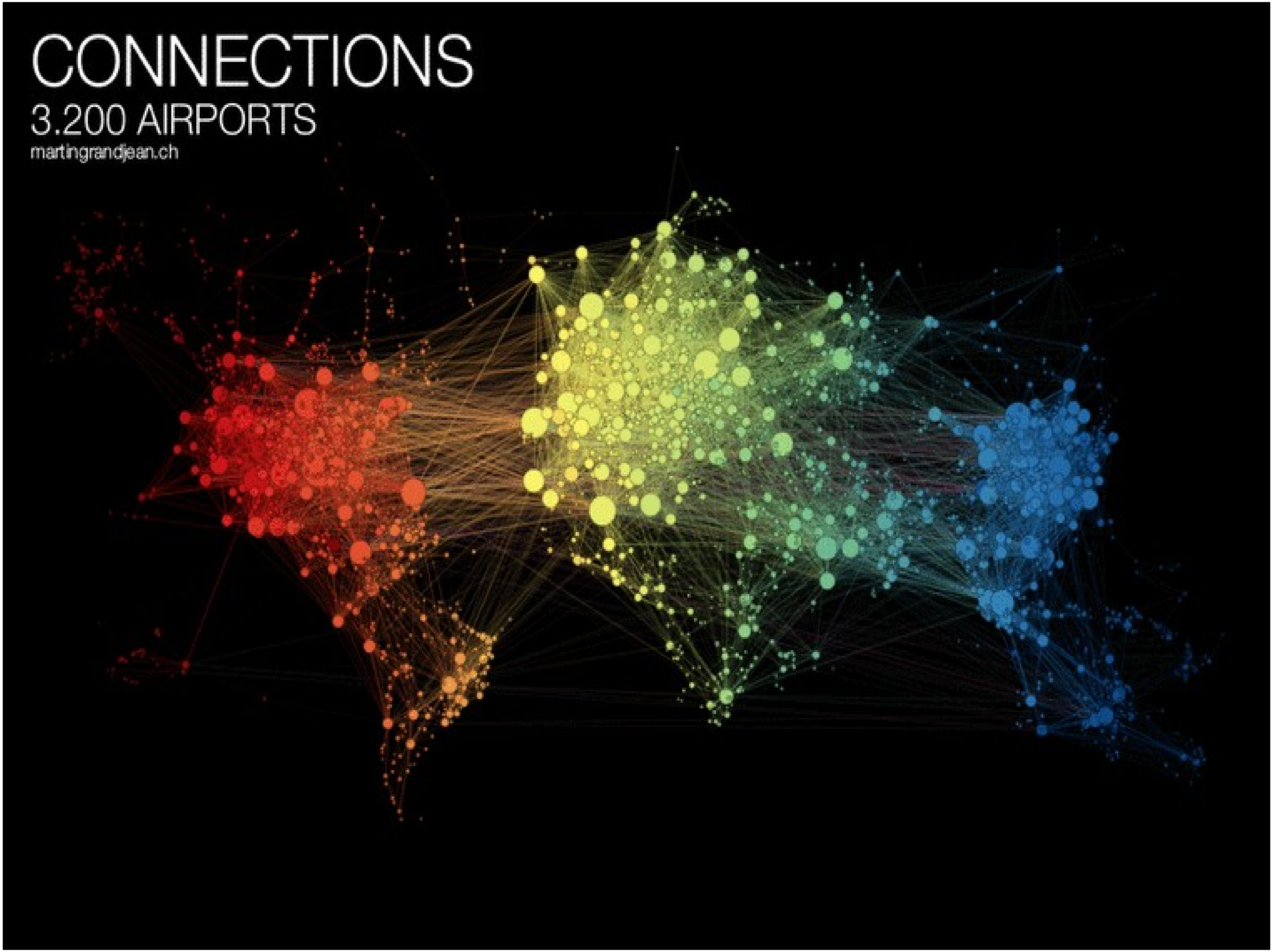


$$\frac{dX_i}{dt} = \nu_i N_i - \lambda_i X_i - \mu_i X_i,$$

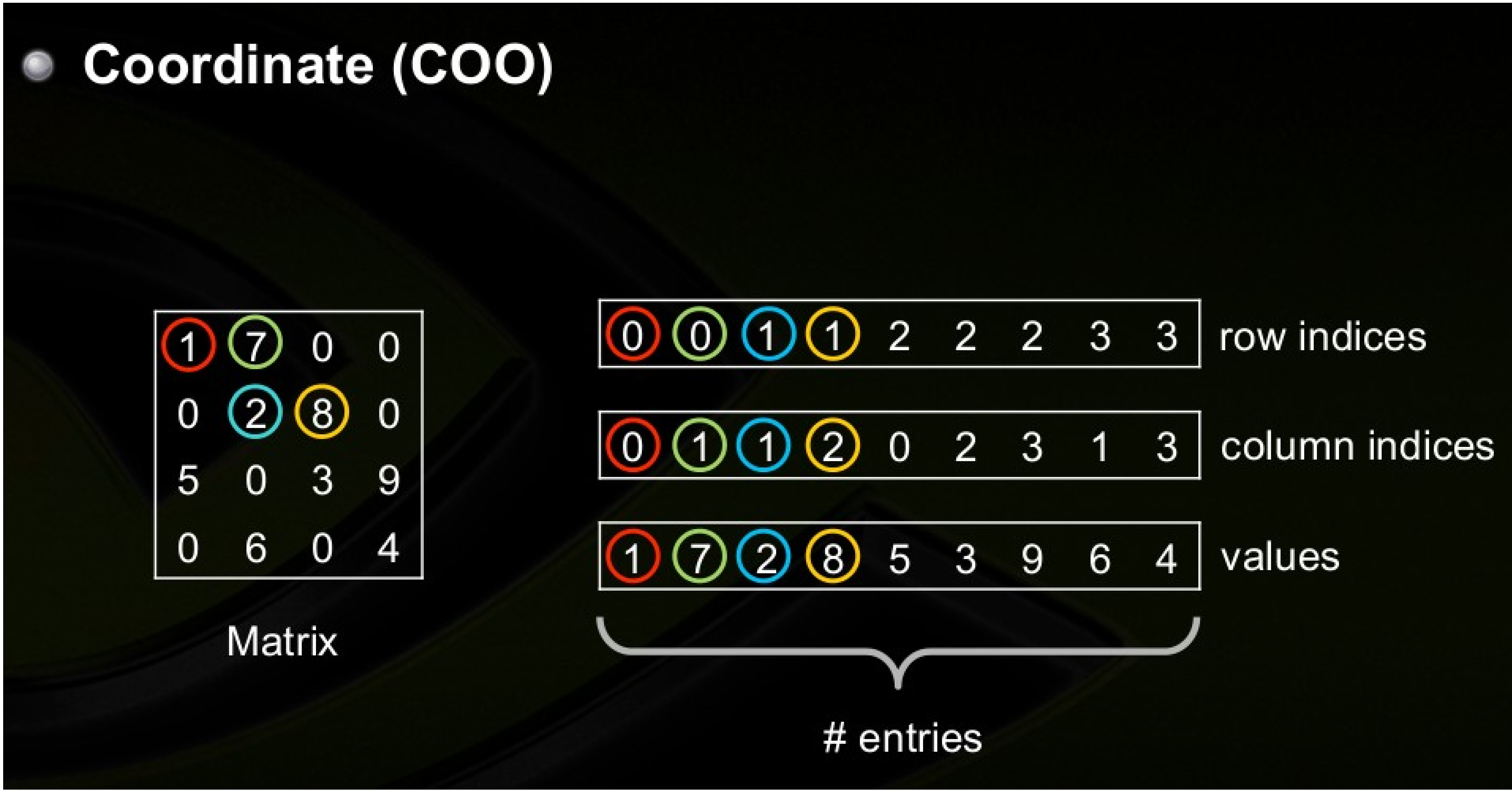
$$\frac{dY_i}{dt} = \lambda_i X_i - \gamma_i Y_i - \mu_i Y_i,$$

$$\boxed{\lambda_i = \beta_i \sum_j \rho_{ij} \frac{X_j}{N_i},}$$

Elementos de matriz
Típicamente rala

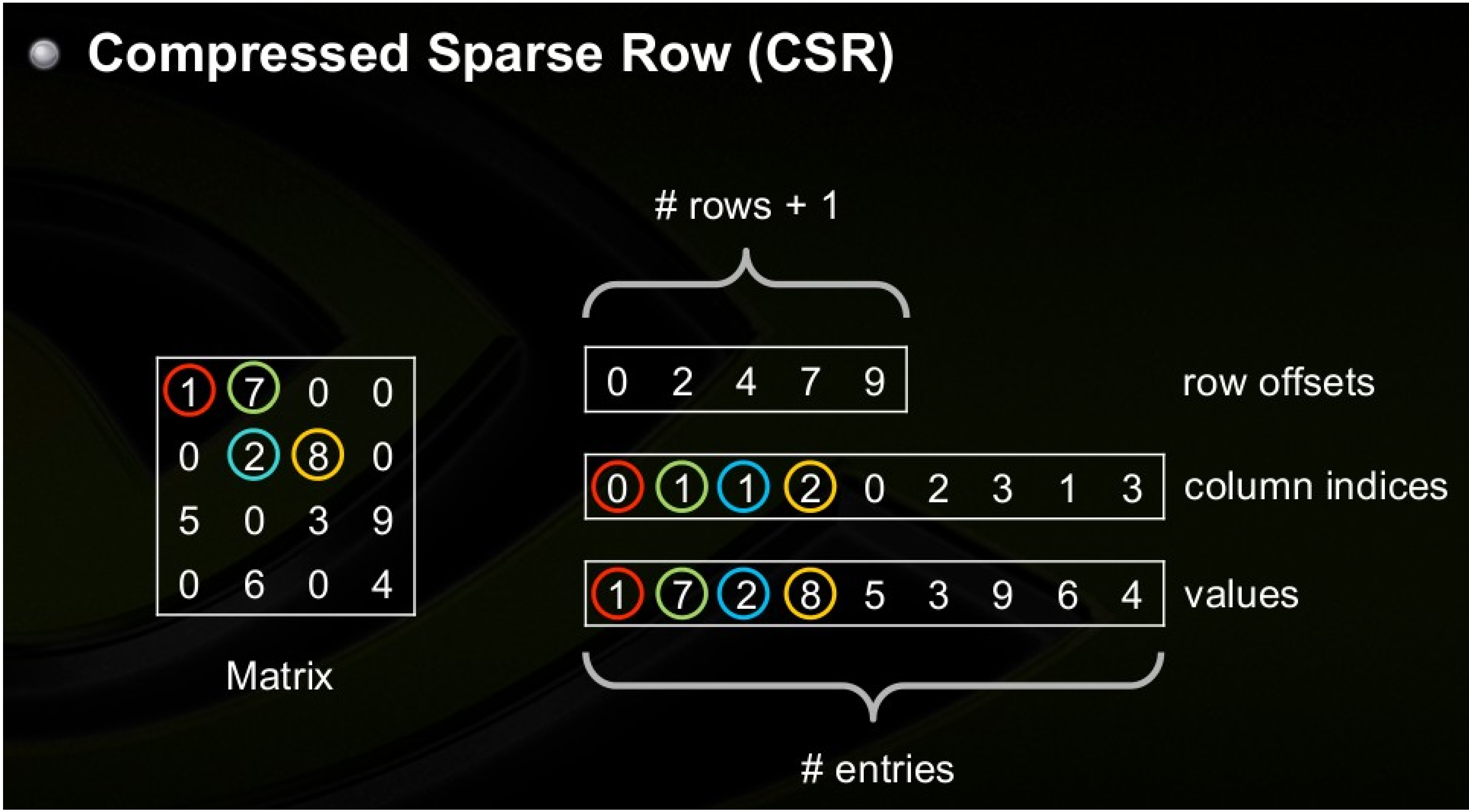


Matriz de adyacencia rala…
Viajan personas... portando enfermedades, mercancías, etc!



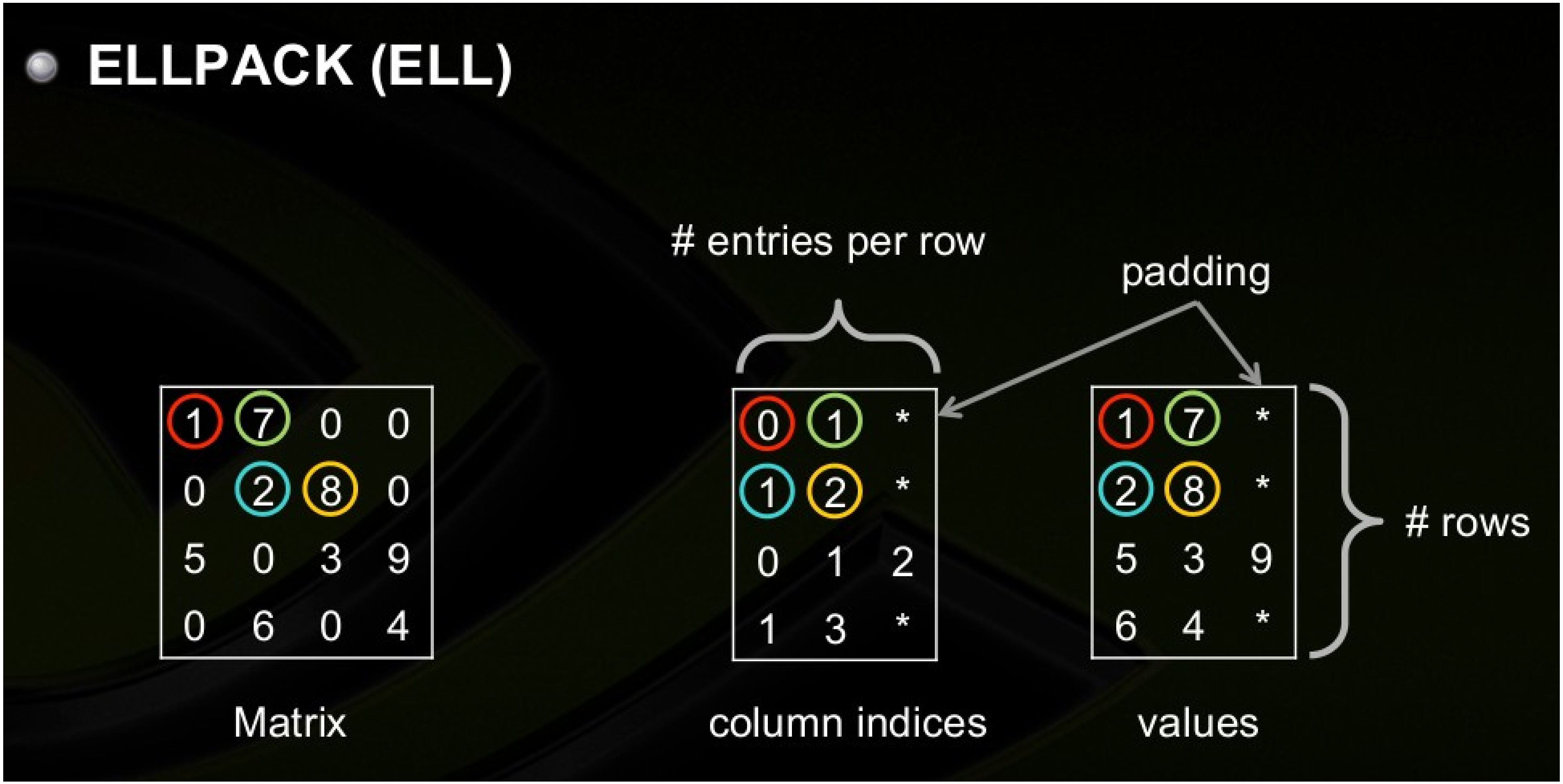
Nathan Bell, NVIDIA

Formatos para matrices ralas



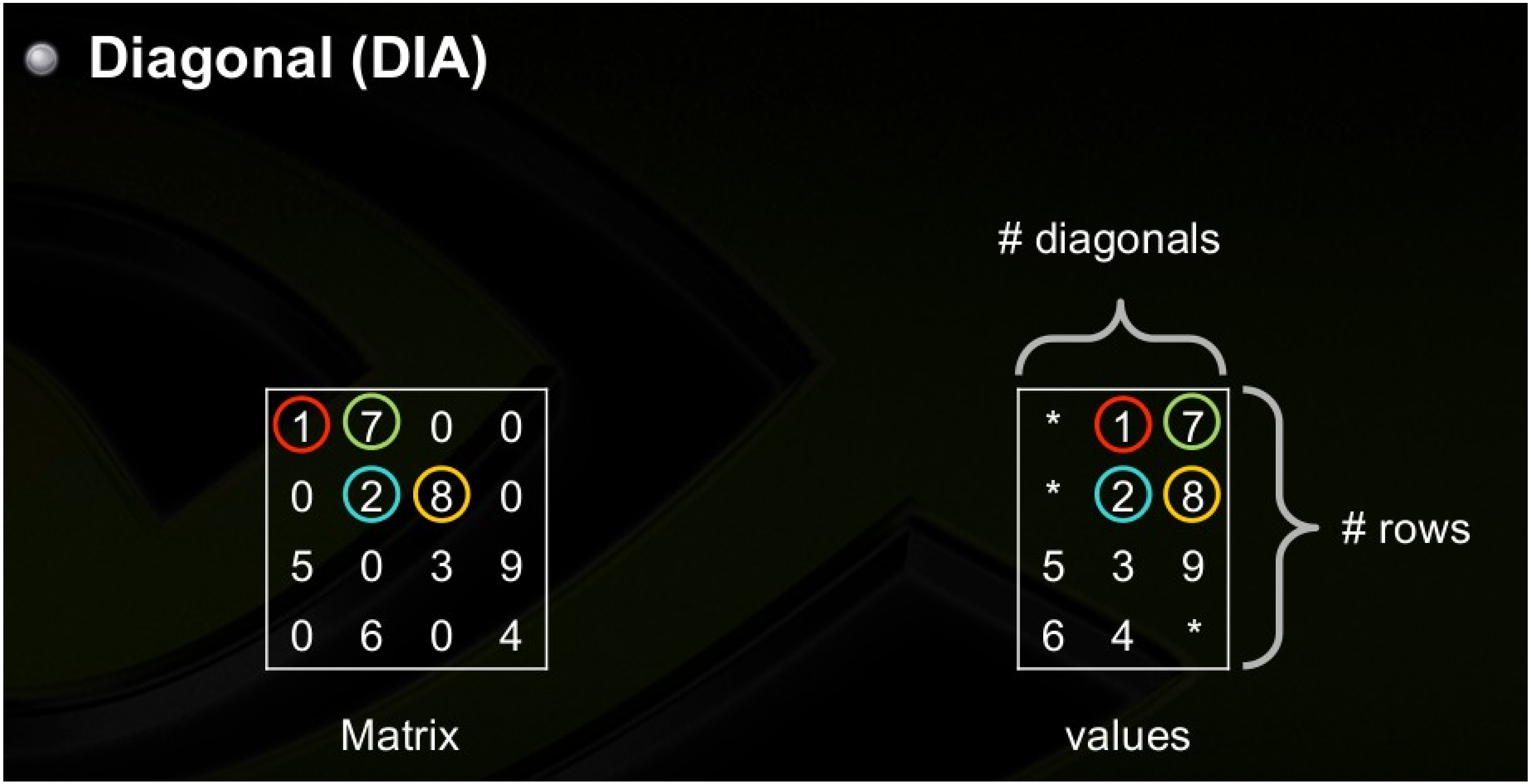
Nathan Bell, NVIDIA

Formatos para matrices ralas



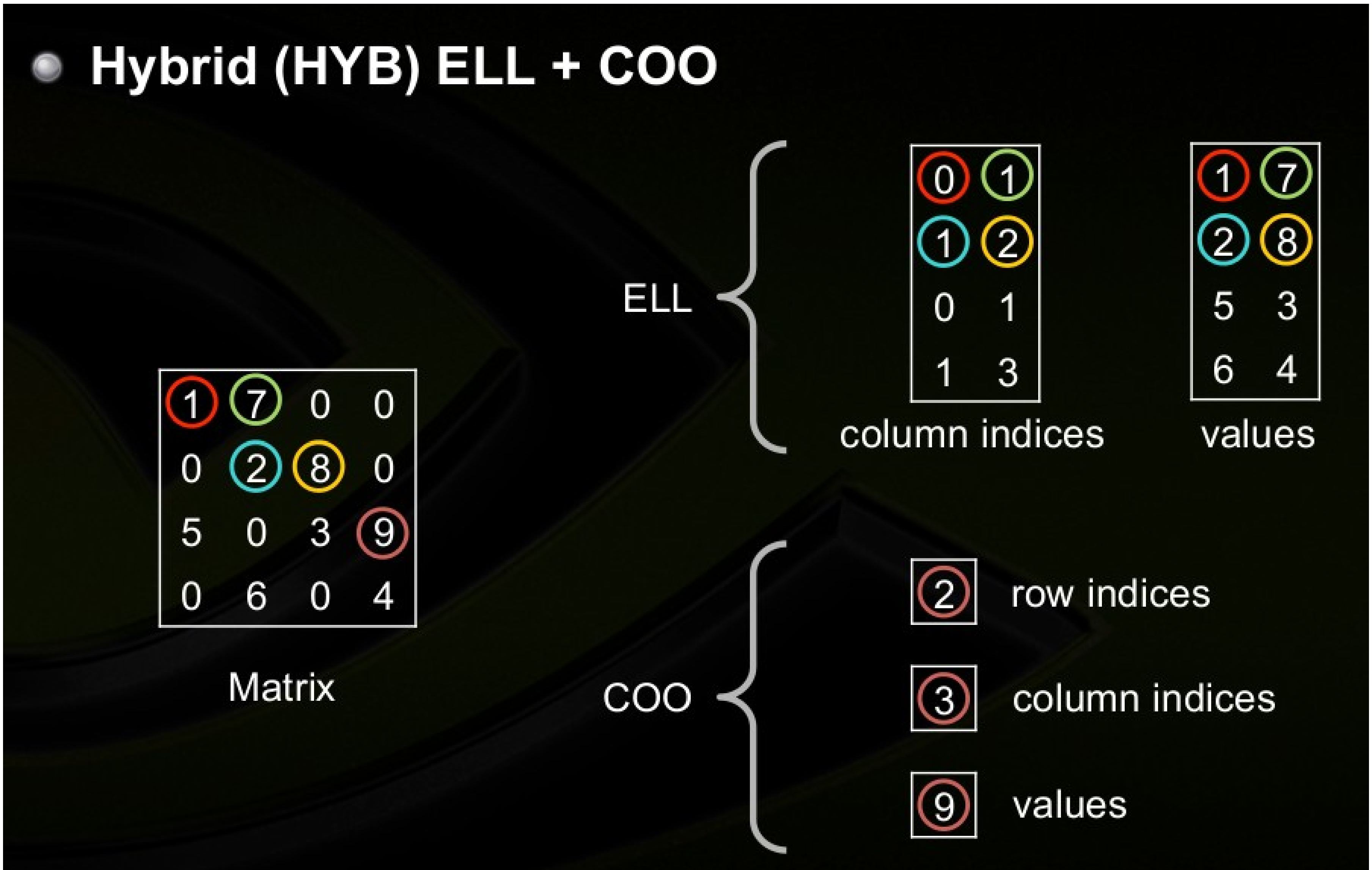
Nathan Bell, NVIDIA

Formatos para matrices ralas



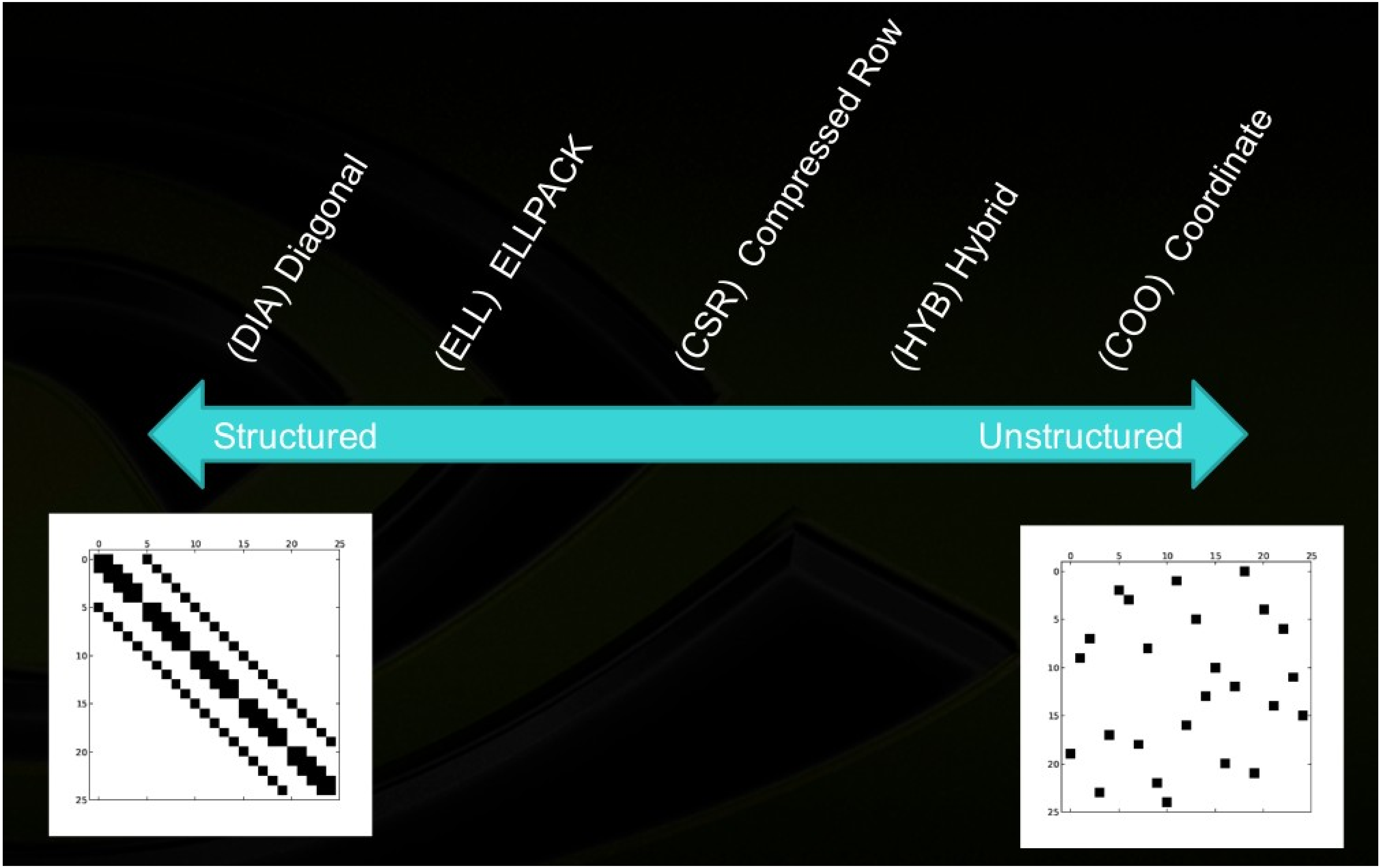
Nathan Bell, NVIDIA

Formatos para matrices ralas



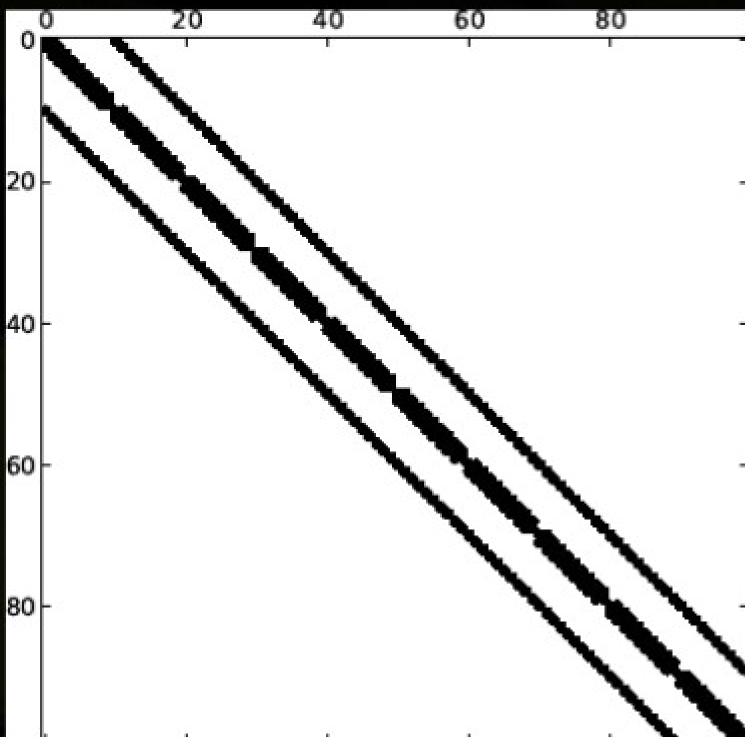
Nathan Bell, NVIDIA

Formatos para matrices ralas



Nathan Bell, NVIDIA

Que formato elegir?



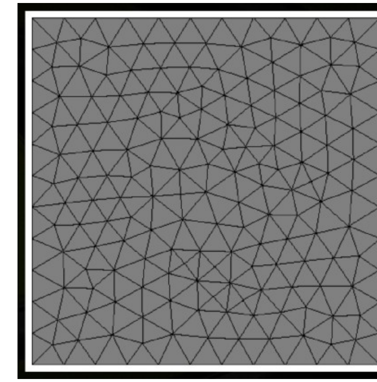
Matrix

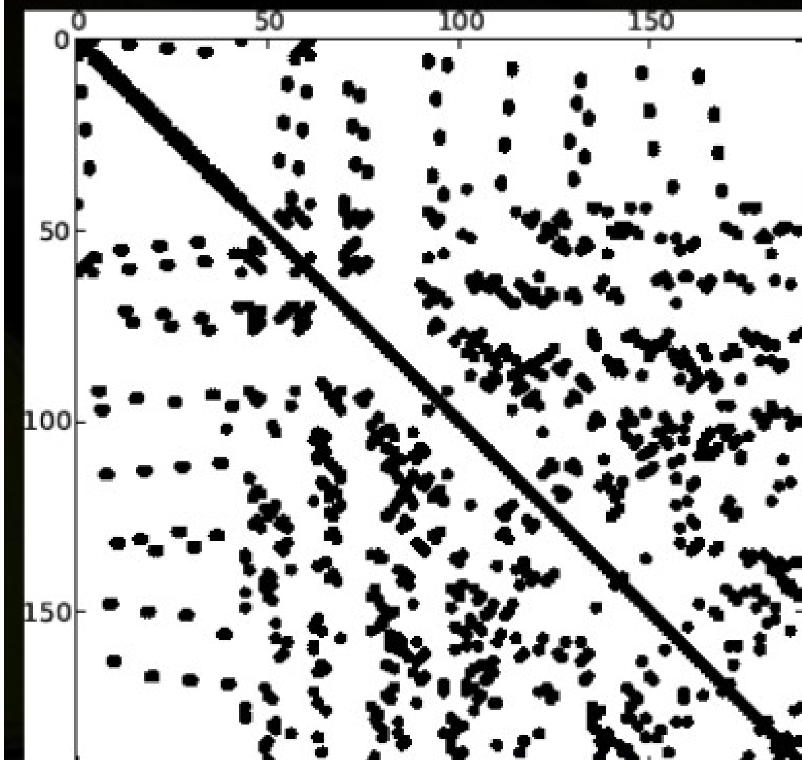
Format	float	double
COO	12.00	16.00
CSR	8.45	12.45
DIA	4.05	8.10
ELL	8.11	12.16
HYB	8.11	12.16

Bytes per Nonzero Entry

Nathan Bell, NVIDIA

Que formato elegir?





Matrix

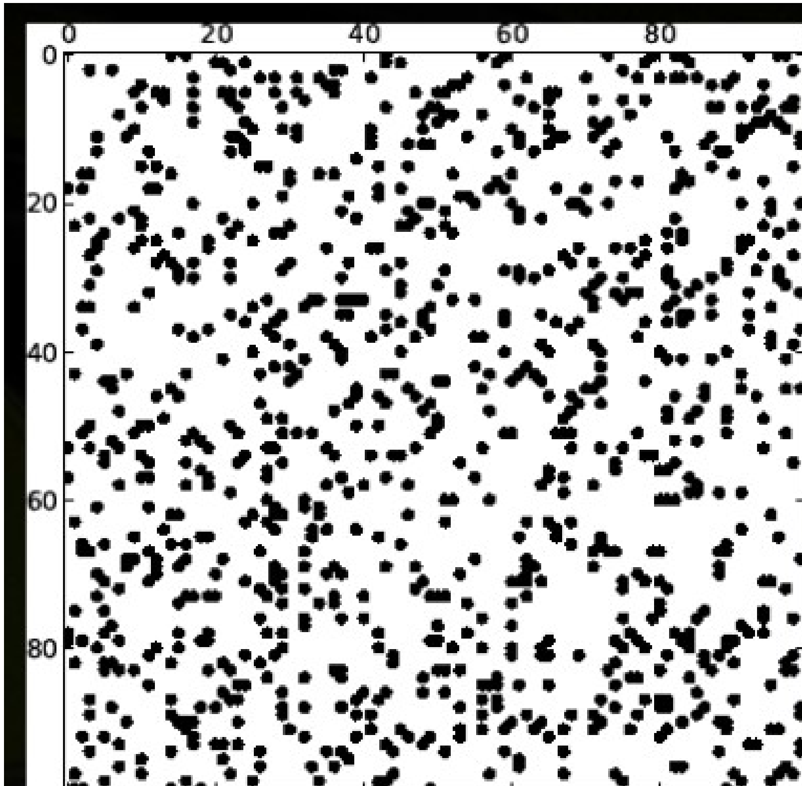
Format	float	double
COO	12.00	16.00
CSR	8.62	12.62
DIA	164.11	328.22
ELL	11.06	16.60
HYB	9.00	13.44

Bytes per Nonzero Entry

Nathan Bell, NVIDIA

Que formato elegir?

Random



Matrix

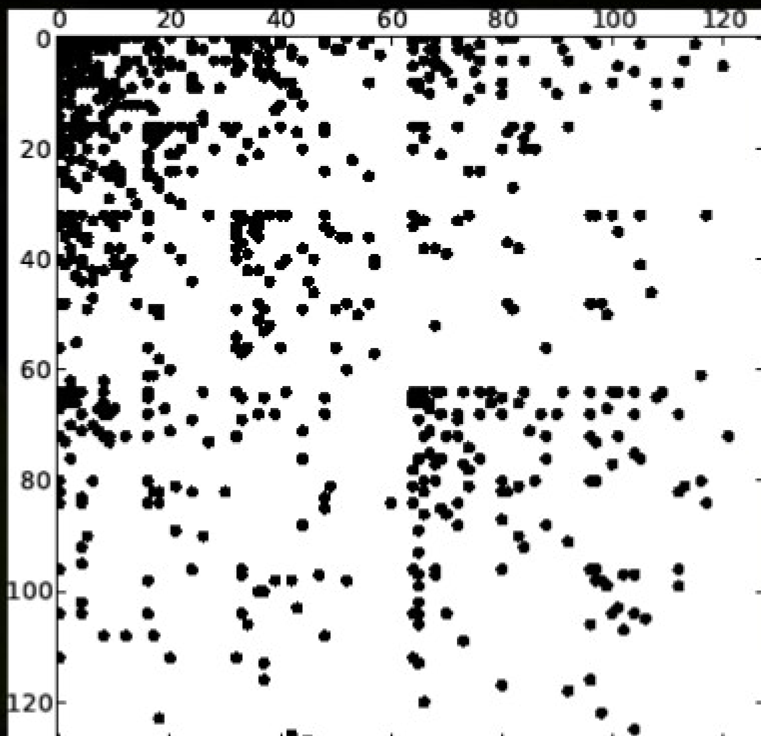
Format	Float	Double
COO	12.00	16.00
CSR	8.42	12.42
DIA	76.83	153.65
ELL	14.20	21.29
HYB	9.60	14.20

Bytes per Nonzero Entry

Nathan Bell, NVIDIA

Que formato elegir?

Power law graph



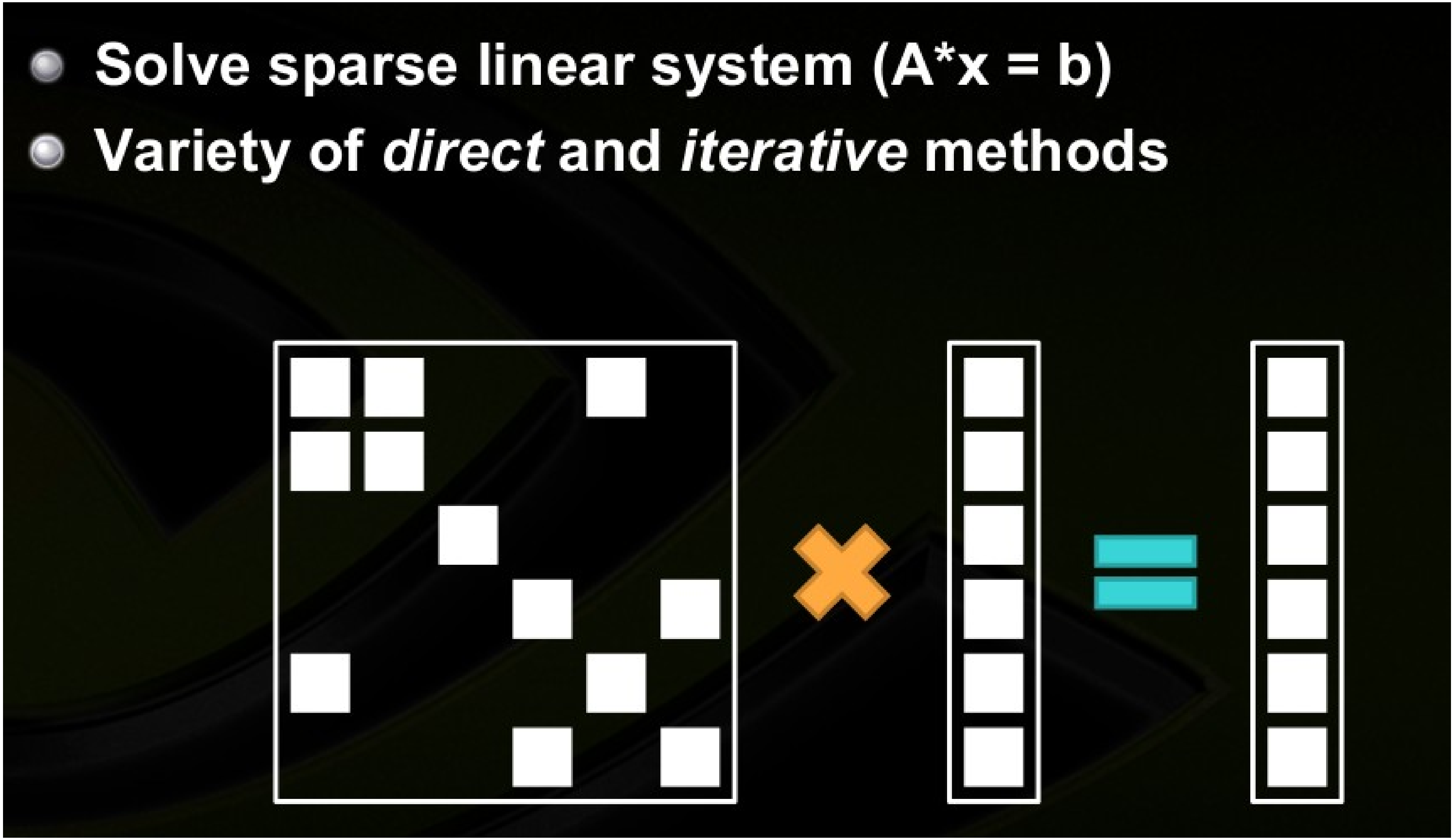
Matrix

Format	Float	Double
COO	12.00	16.00
CSR	8.74	12.73
DIA	118.83	237.66
ELL	49.88	74.82
HYB	13.50	19.46

Bytes per Nonzero Entry

Nathan Bell, NVIDIA

Sistemas lineales ralos



Nathan Bell, NVIDIA

CuSPARSE: Formatos de Matrices Ralas

- Compressed Sparse Row Format (CSR)

The only way the CSR differs from the COO format is that the array containing the row indices is compressed in CSR format. The m×n sparse matrix A is represented in CSR format by the following parameters.

$$\begin{matrix} 1.0 & 4.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 2.0 & 3.0 & 0.0 & 0.0 \\ 5.0 & 0.0 & 0.0 & 7.0 & 8.0 \\ 0.0 & 0.0 & 9.0 & 0.0 & 6.0 \end{matrix}$$

$$\begin{aligned} \mathrm{csrValA} &= 1.0 \quad 4.0 \quad 2.0 \quad 3.0 \quad 5.0 \quad 7.0 \quad 8.0 \quad 9.0 \quad 6.0 \\ \mathrm{csrRowPtrA} &= 0 \quad 2 \quad 4 \quad 7 \quad 9 \\ \mathrm{csrColIndA} &= 0 \quad 1 \quad 1 \quad 2 \quad 0 \quad 3 \quad 4 \quad 2 \quad 4 \end{aligned}$$

`nnz`	(integer)	The number of nonzero elements in the matrix.
`csrValA`	(pointer)	Points to the data array of length `nnz` that holds all nonzero values of A in row-major format.
`csrRowPtrA`	(pointer)	Points to the integer array of length `m+1` that holds indices into the arrays `csrColIndA` and `csrValA`. The first `m` entries of this array contain the indices of the first nonzero element in the `i`th row for `i=i,...,m`, while the last entry contains `nnz+csrRowPtrA(0)`. In general, `csrRowPtrA(0)` is `0` or `1` for zero- and one-based indexing, respectively.
`csrColIndA`	(pointer)	Points to the integer array of length `nnz` that contains the column indices of the corresponding elements in array `csrValA`.

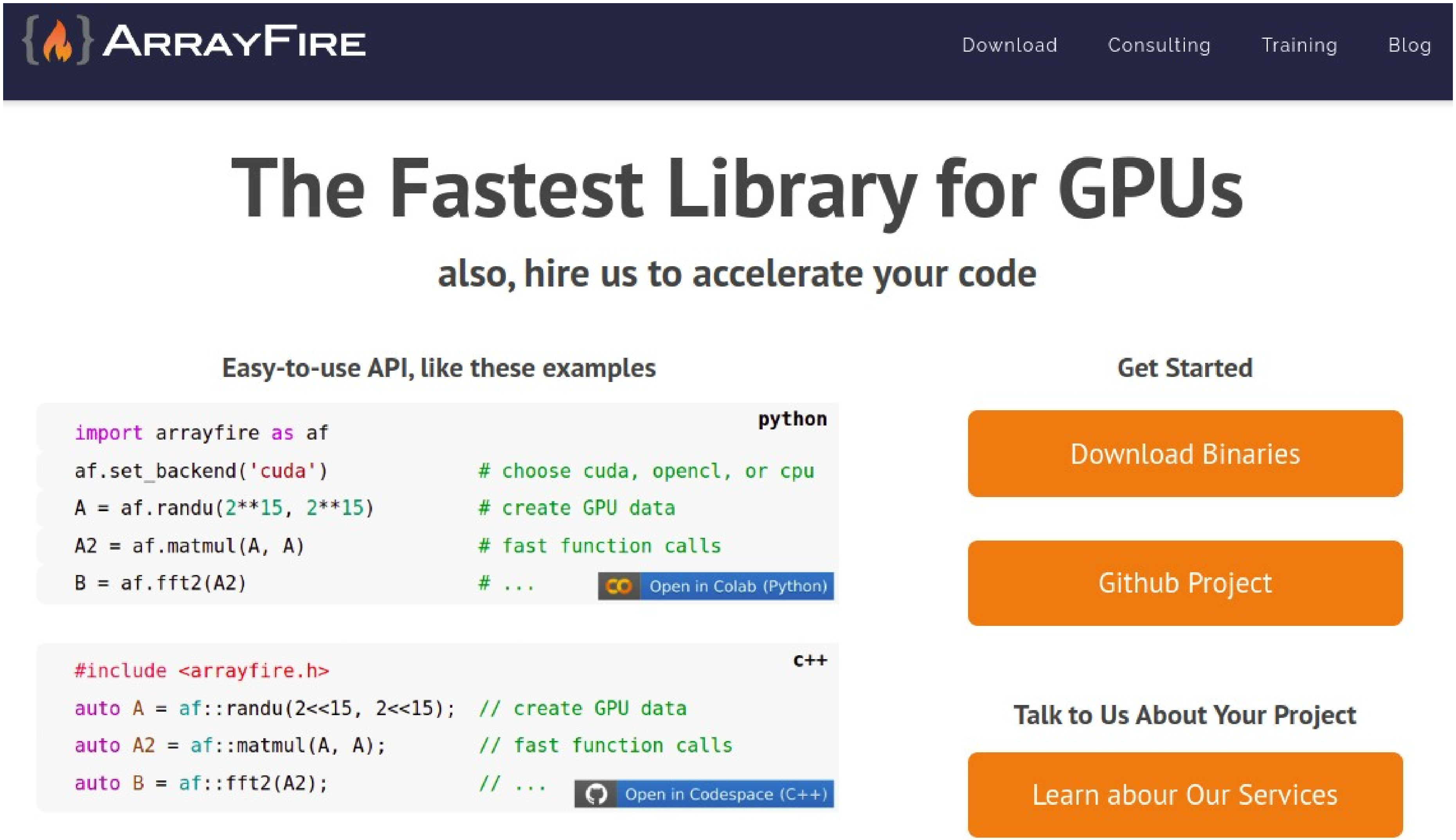
CuSPARSE: Formatos de Matrices Ralas

- 3.2.2. Coordinate Format (COO)
- 3.2.3. Compressed Sparse Row Format (CSR)
- 3.2.4. Compressed Sparse Column Format (CSC)
- 3.2.5. Block Compressed Sparse Row Format (BSR)
- 3.2.6. Extended BSR Format (BSRX)

Ejericicios CuSPARSE

- **/share/apps/icnpg/clases/Clases_cusparse/**
  - // Create the cuSPARSE handle
  - // Allocate device memory for vectors and the dense form of the matrix A
  - // Construct a descriptor of the matrix A
  - // Transfer the input vectors and dense matrix A to the device
  - // Compute the number of non-zero elements in A
  - // Allocate device memory to store the sparse CSR representation of A
  - // Convert A from a dense formatting to a CSR formatting, using the GPU
  - // Perform matrix-matrix multiplication with the CSR-formatted matrix A
  - // Copy the result vector back to the host

ArrayFire

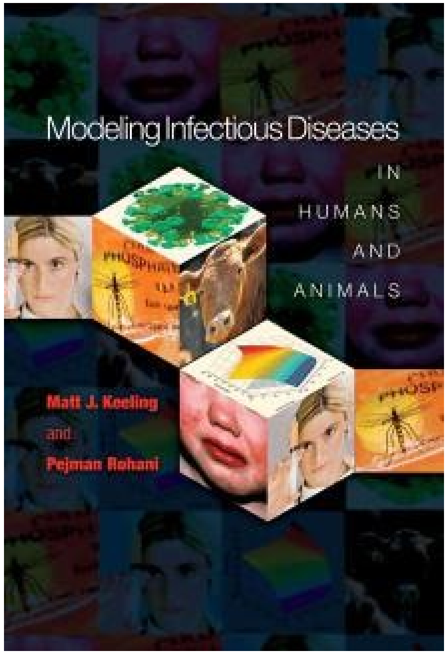


Mejor performance que cupy, y además corre en CUDA y OPENCL, y se puede usar en C++

(usa cuSOLVER para gpu de nvidia)

Ejericicios SPARSE con cupy y ArrayFire

- **ICNPG_try-arrayfire-on-colab**



$$\frac{dX_i}{dt} = \nu_i N_i - \lambda_i X_i - \mu_i X_i,$$

$$\frac{dY_i}{dt} = \lambda_i X_i - \gamma_i Y_i - \mu_i Y_i,$$

$$\lambda_i = \beta_i \sum_j \rho_{ij} \frac{X_j}{N_i},$$

Elementos de matriz
Típicamente rala

CuSOLVER

$$Ax = b \qquad A_i x_i = b_i$$

$$Ax = \lambda x \qquad Ax = \lambda Bx$$

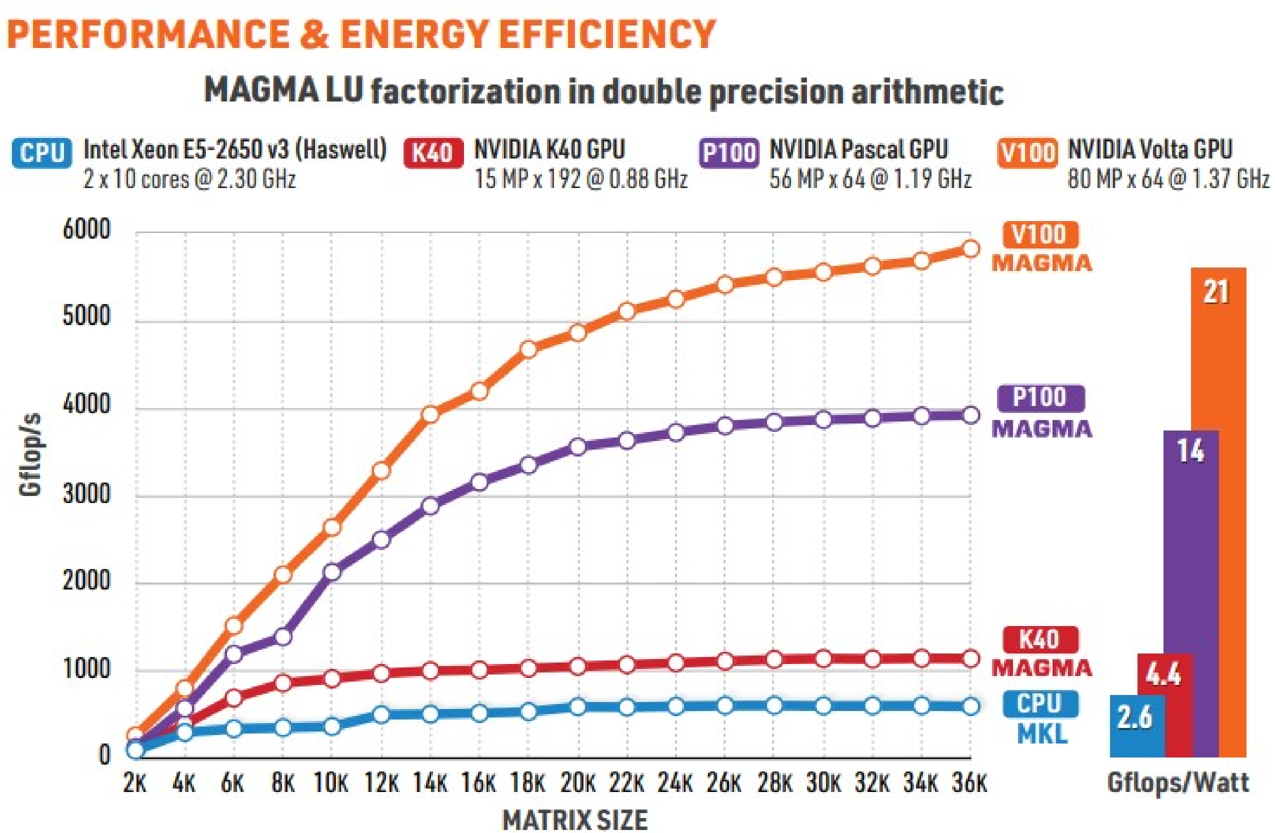
- **¿Qué es?**: A high-level package based on the cuBLAS and cuSPARSE libraries… para resolver sistemas de ecuaciones lineales.
- **Operaciones:** matrix factorization and triangular solve routines for dense matrices, a sparse least-squares solver and an eigenvalue solver + a new refactorization library useful for solving sequences of matrices with a shared sparsity pattern.
- **Tres sabores**:
  - CuSolverDN: matrices densas generales.
  - CuSolverSP: formato CSR, matrices generales o especiales (simétricas/hermíticas, etc).
  - CuSolverRF: refactorización, útil para sistemas “batcheados”

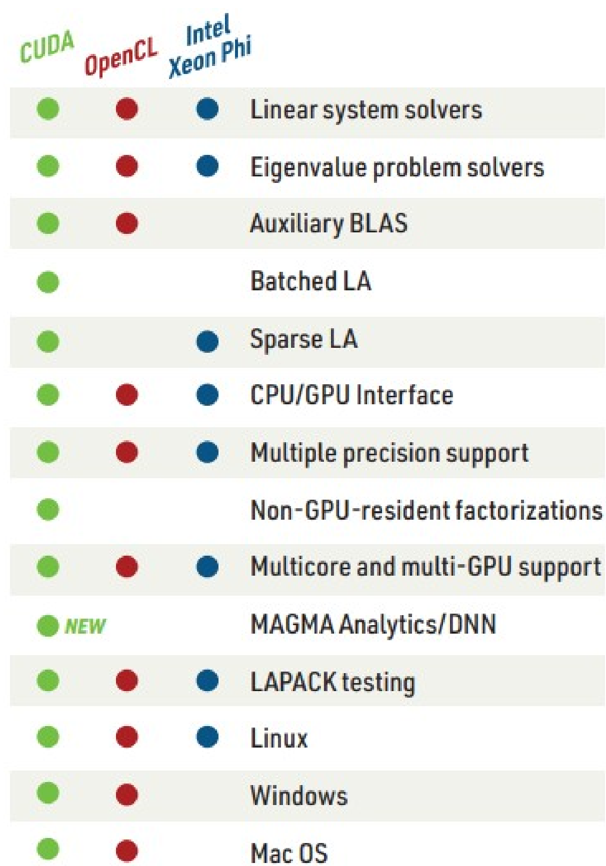
cuSOLVER

https://docs.nvidia.com/cuda/cusolver/index.html

MAGMA

- Algebra lineal numérica para sistemas híbridos/heterogéneos: CPU-multicore + GPUs + ?





https://icl.utk.edu/magma/overview/index.html

http://www.icl.utk.edu/files/print/2017/magma-sc17.pdf

(usa cuSOLVER para gpu de nvidia)

Ejercicios CuSolver

- /share/apps/icnpg/clases/Clases_cusolver/

Linear algebra (`cupy.linalg`)

Hint

NumPy API Reference: Linear algebra (numpy.linalg)

See also

Linear algebra (cupyx.scipy.linalg)

Linear algebra (`cupyx.scipy.linalg`)

Hint

SciPy API Reference: Linear algebra (scipy.linalg)

Linear algebra (`cupyx.scipy.linalg`)

- cupyx.scipy.linalg.solve_triangular
- cupyx.scipy.linalg.tril
- cupyx.scipy.linalg.triu
- cupyx.scipy.linalg.lu
- cupyx.scipy.linalg.lu_factor
- cupyx.scipy.linalg.lu_solve
- cupyx.scipy.linalg.block_diag
- cupyx.scipy.linalg.circulant
- cupyx.scipy.linalg.companion
- cupyx.scipy.linalg.convolution_matrix
- cupyx.scipy.linalg.dft
- cupyx.scipy.linalg.fiedler
- cupyx.scipy.linalg.fiedler_companion
- cupyx.scipy.linalg.hadamard
- cupyx.scipy.linalg.hankel
- cupyx.scipy.linalg.helmert
- cupyx.scipy.linalg.hilbert
- cupyx.scipy.linalg.kron
- cupyx.scipy.linalg.leslie
- cupyx.scipy.linalg.toeplitz
- cupyx.scipy.linalg.tri

Sparse matrices (`cupyx.scipy.sparse`

- cupyx.scipy.sparse.coo_matrix
- cupyx.scipy.sparse.csc_matrix
- cupyx.scipy.sparse.csr_matrix
- cupyx.scipy.sparse.dia_matrix
- cupyx.scipy.sparse.spmatrix
- cupyx.scipy.sparse.eye
- cupyx.scipy.sparse.identity
- cupyx.scipy.sparse.kron
- cupyx.scipy.sparse.diags
- cupyx.scipy.sparse.spdiags
- cupyx.scipy.sparse.tril
- cupyx.scipy.sparse.triu
- cupyx.scipy.sparse.bmat
- cupyx.scipy.sparse.hstack
- cupyx.scipy.sparse.vstack
- cupyx.scipy.sparse.rand
- cupyx.scipy.sparse.random
- cupyx.scipy.sparse.find
- cupyx.scipy.sparse.issparse
- cupyx.scipy.sparse.isspmatrix
- cupyx.scipy.sparse.isspmatrix_csc
- cupyx.scipy.sparse.isspmatrix_csr
- cupyx.scipy.sparse.isspmatrix_coo
- cupyx.scipy.sparse.isspmatrix_dia

Linear algebra (`cupy.linalg`)

- cupy.dot
- cupy.vdot
- cupy.inner
- cupy.outer
- cupy.matmul
- cupy.tensordot
- cupy.einsum
- cupy.linalg.matrix_power
- cupy.kron
- cupy.linalg.cholesky
- cupy.linalg.qr
- cupy.linalg.svd
- cupy.linalg.eigh
- cupy.linalg.eigvalsh
- cupy.linalg.norm
- cupy.linalg.det
- cupy.linalg.matrix_rank
- cupy.linalg.slogdet
- cupy.trace
- cupy.linalg.solve
- cupy.linalg.tensorsolve
- cupy.linalg.lstsq
- cupy.linalg.inv
- cupy.linalg.pinv

CUPY

CUPY

Linear Algebra

Matrix and vector products

`cupy.cross`	Returns the cross product of two vectors.
`cupy.dot`	Returns a dot product of two arrays.
`cupy.vdot`	Returns the dot product of two vectors.
`cupy.inner`	Returns the inner product of two arrays.
`cupy.outer`	Returns the outer product of two vectors.
`cupy.matmul`	Returns the matrix product of two arrays and is the implementation
`cupy.tensordot`	Returns the tensor dot product of two arrays along specified axes.
`cupy.einsum`	Evaluates the Einstein summation convention on the operands.
`cupy.linalg.matrix_power`	Raise a square matrix to the (integer) power *n*.
`cupy.kron`	Returns the kronecker product of two arrays.

Matrix computations on the GPU

CUBLAS, CUSOLVER and MAGMA by example

Andrzej Chrzęszczyk

Jan Kochanowski University, Kielce, Poland

Jacob Anders

CSIRO, Canberra, Australia

Version 2017

Matrix Computations on GPU with ArrayFire - Python and ArrayFire - C/C++

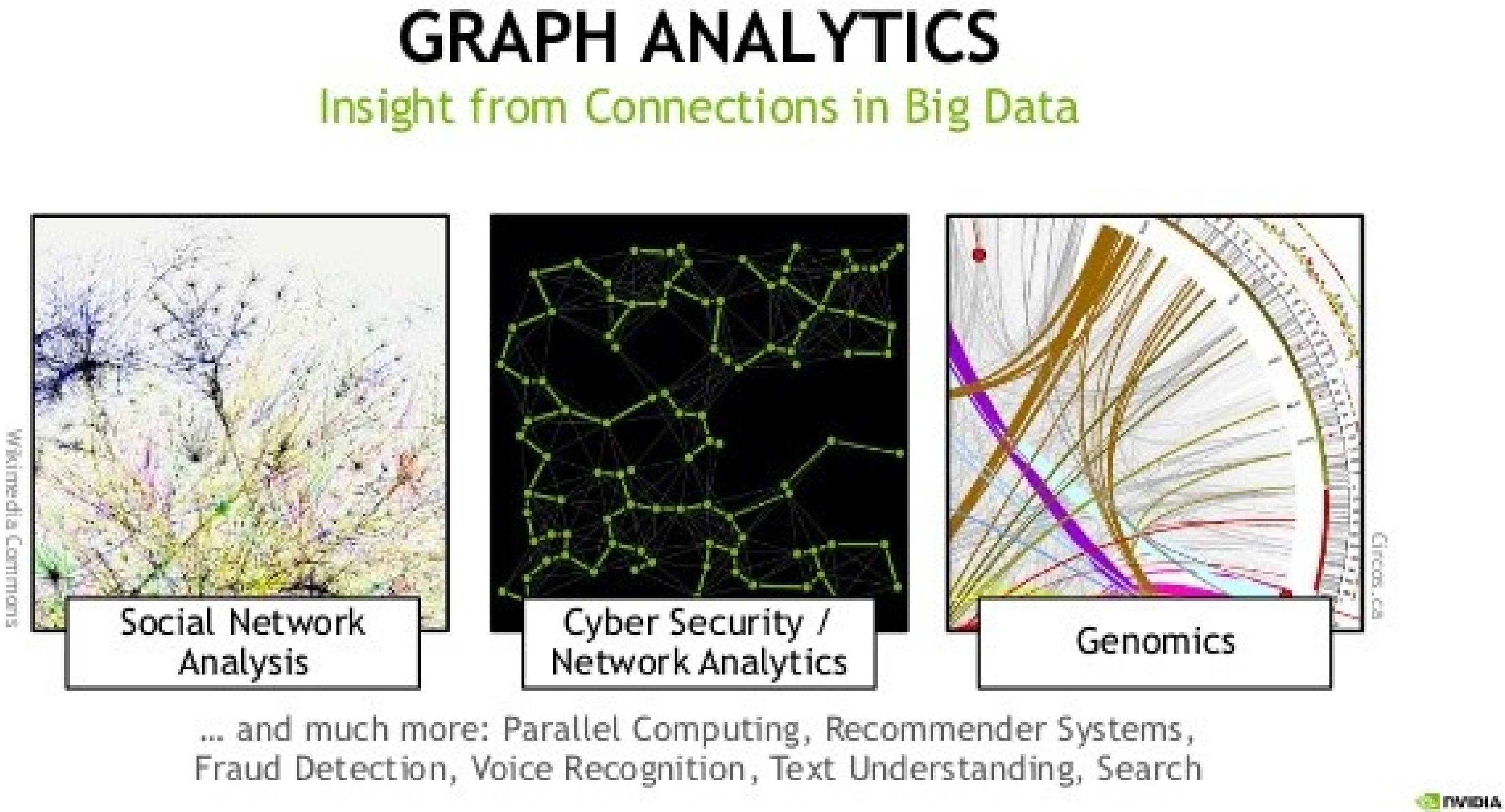
Andrzej Chrzęszczyk

Jan Kochanowski University

Version 2017

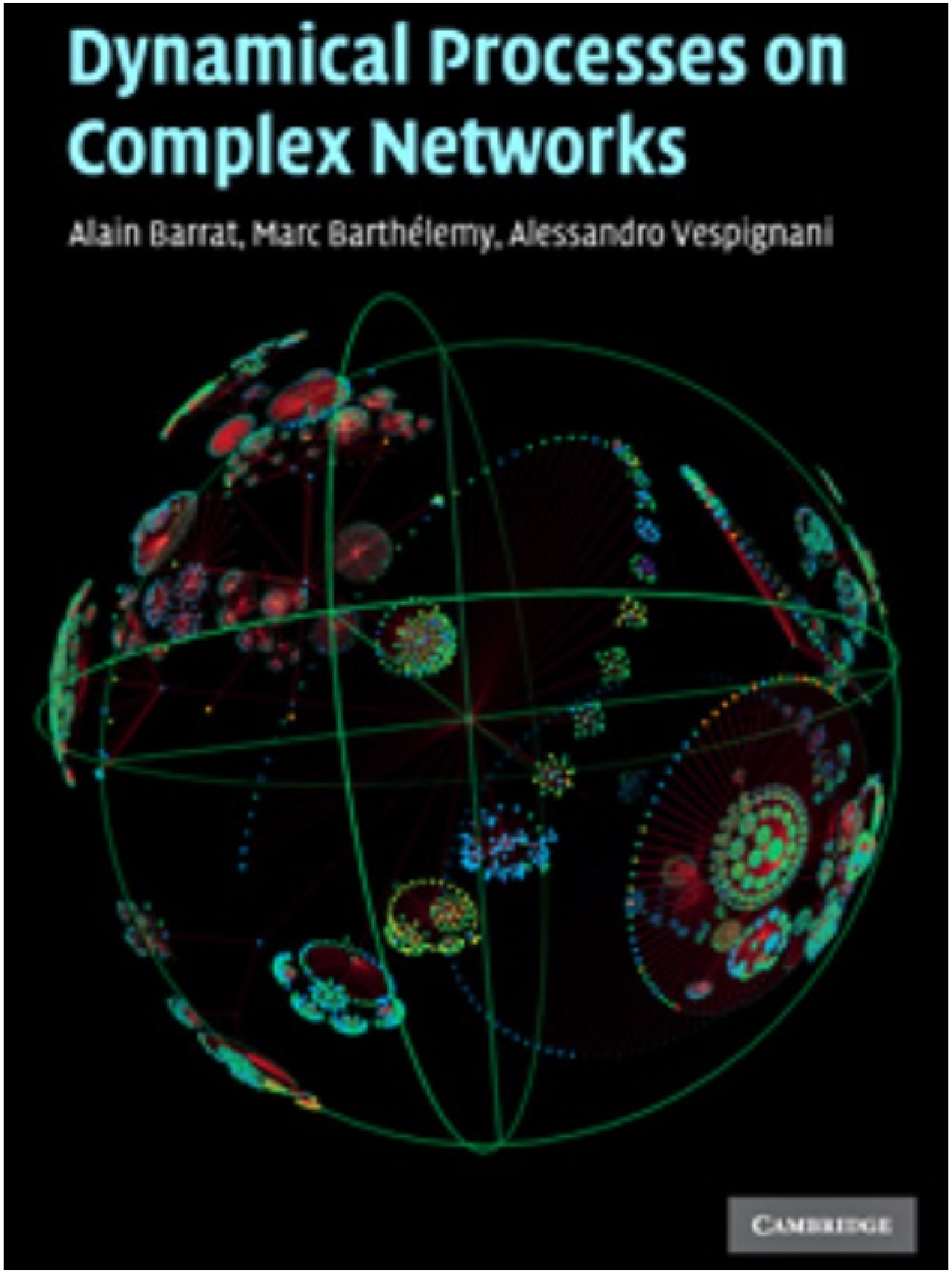
NVGRAPH (C) / CUGRAPH (python)

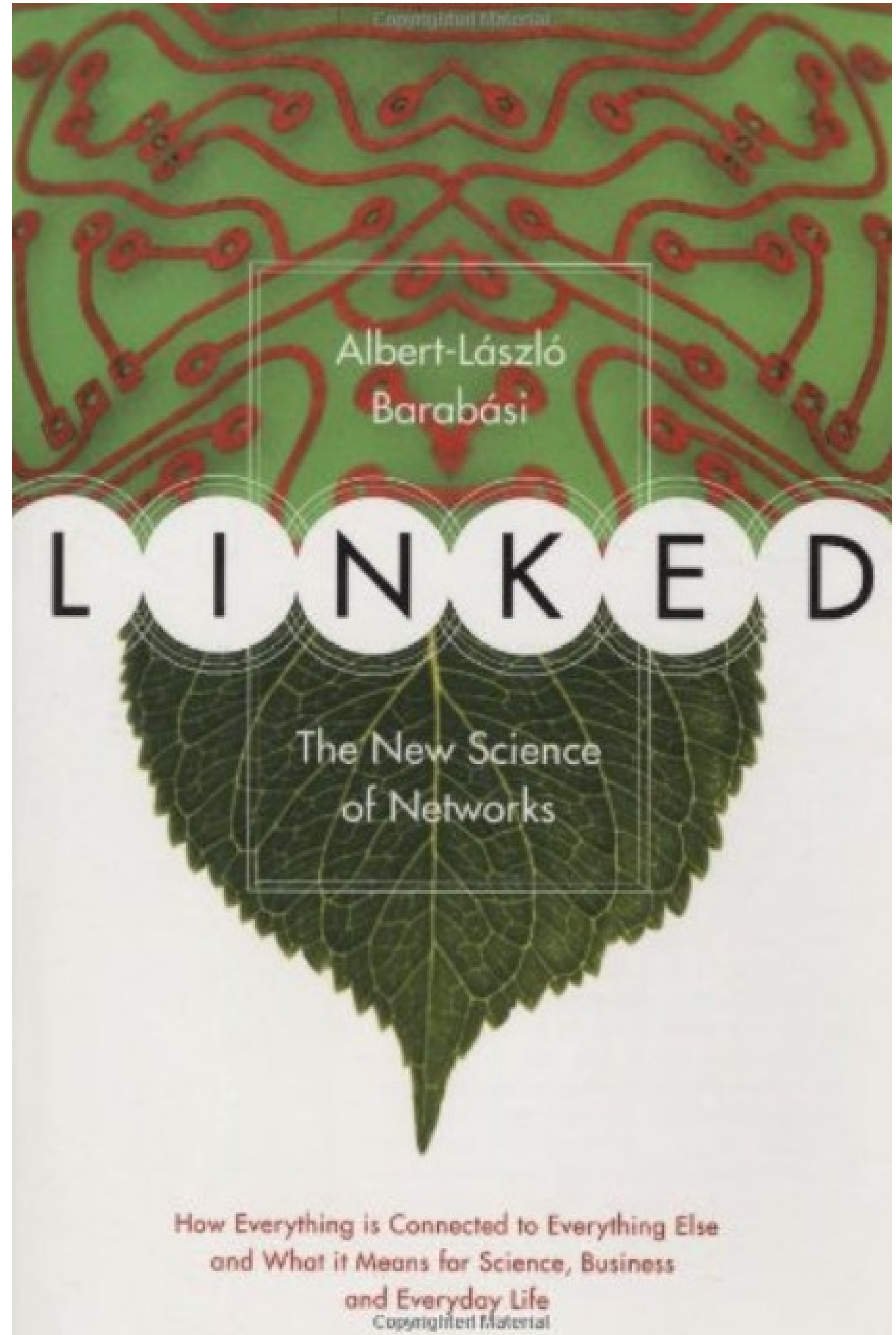
- graph construction and manipulation primitives, and a set of useful graph algorithms optimized for the GPU.



https://github.com/rapidsai/cugraph

https://developer.nvidia.com/nvgraph





http://networkrepository.com/graphvis.php

RAPIDS (python o C++)

DATA SCIENCE AT THE SPEED OF THOUGHT

WHAT IS RAPIDS

RAPIDS provides unmatched speed with familiar APIs that match the most popular PyData libraries. Built on the shoulders of giants including **NVIDIA CUDA** and **Apache Arrow**, it unlocks the speed of GPUs with code you already know.

Learn more on the About Section ↗

WHY USE RAPIDS

RAPIDS allows fluid, creative interaction with data for everyone from BI users to AI researchers on the cutting edge. GPU acceleration means less time and less cost moving data and training models.

Find out more from RAPIDS Use Cases ↗

OPEN SOURCE ECOSYSTEM

RAPIDS is Open Source and available on **GitHub**. Our mission is to empower and advance the open-source GPU data science data engineering ecosystem.

Jump to RAPIDS on GitHub ↗

FASTER PANDAS WITH CUDF

cuDF is a near drop in replacement to pandas for most use cases and has greatly improved performance.

Run this benchmark yourself ↗

FASTER SCIKIT-LEARN WITH CUML

cuML brings huge speedups to ML modeling with an API that matches scikit-learn.

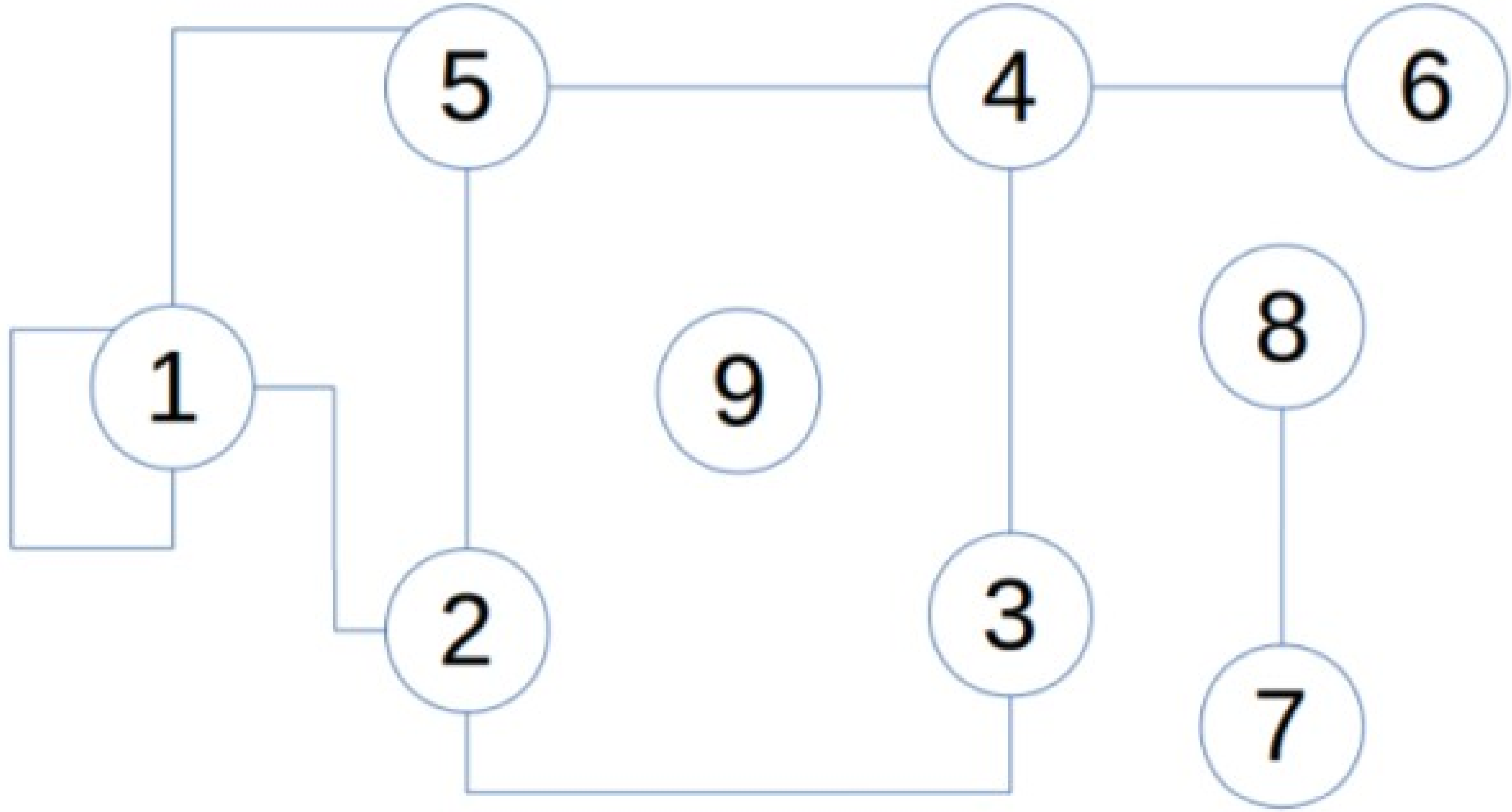
Run this benchmark yourself ↗

FASTER NETWORKX WITH CUGRAPH

cuGraph makes migration from networkX easy, accelerates graph analytics, and allows scaling far beyond existing tools.

Run this benchmark yourself ↗

Ejercicio



Usando CUGRAPH

- Leer matriz de adyacencias csv.
- Número de caminos de exactamente 5 pasos que unen un vértice con cualquier otro.
- Número de componentes del grafo, y sus vértices miembros.
- Colorear, como si fuera un mapa, los vértices usando la mínima cantidad de colores.

ICNPG_ejemplocugraph.ipynb