# Cache-oblivious sparse matrix–vector multiplication

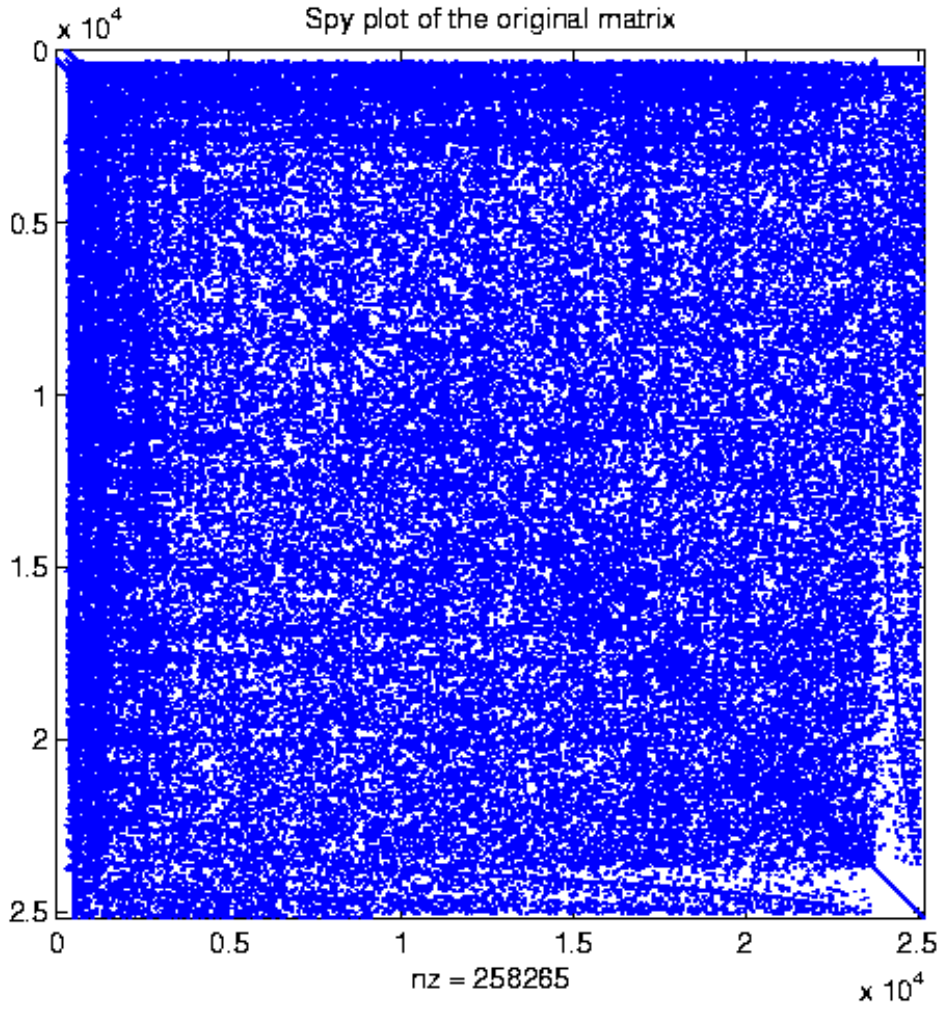
Albert-Jan Yzelman & Rob H. Bisseling

May 2011

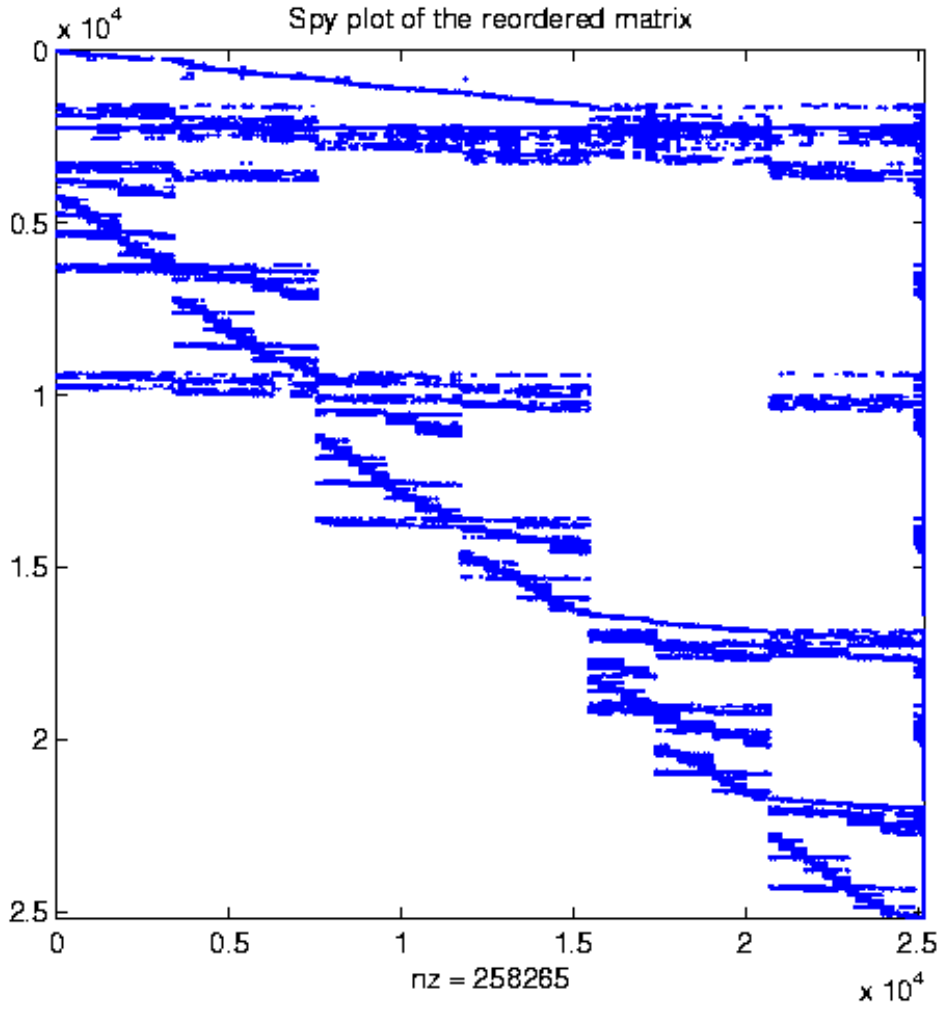
# Sparse matrix reordering



# Chip industry

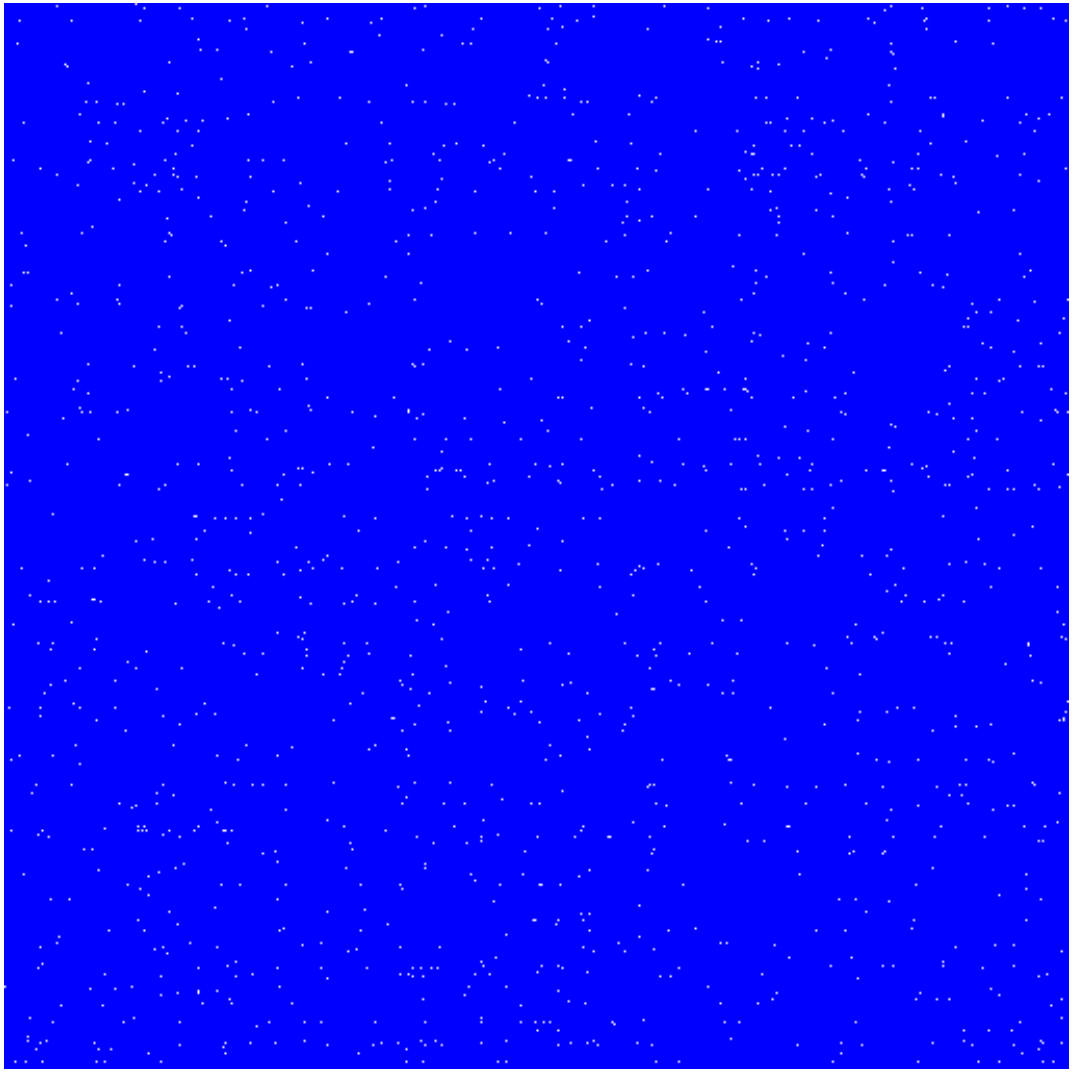


# Chip industry – 1D reordering

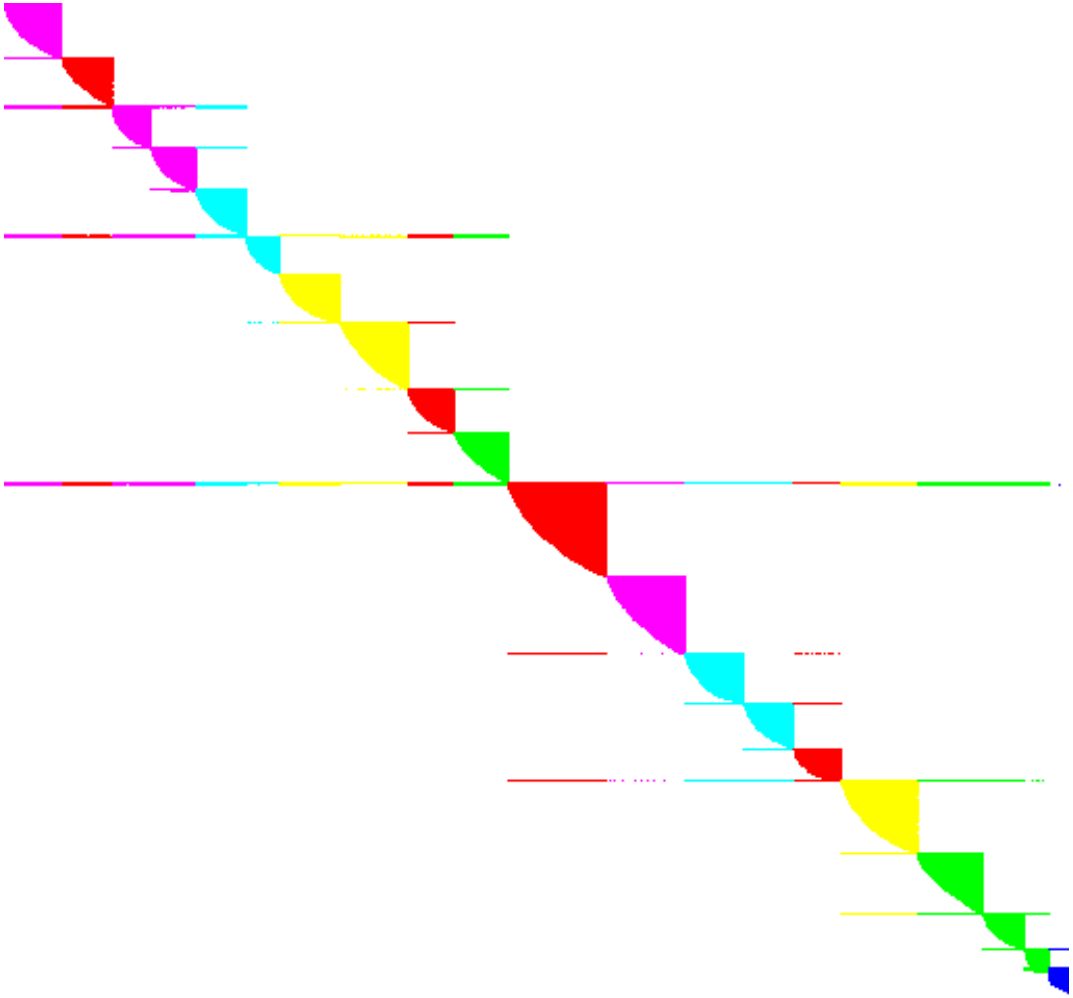



$p = 100, \quad \epsilon = 0.1$

# Link matrix

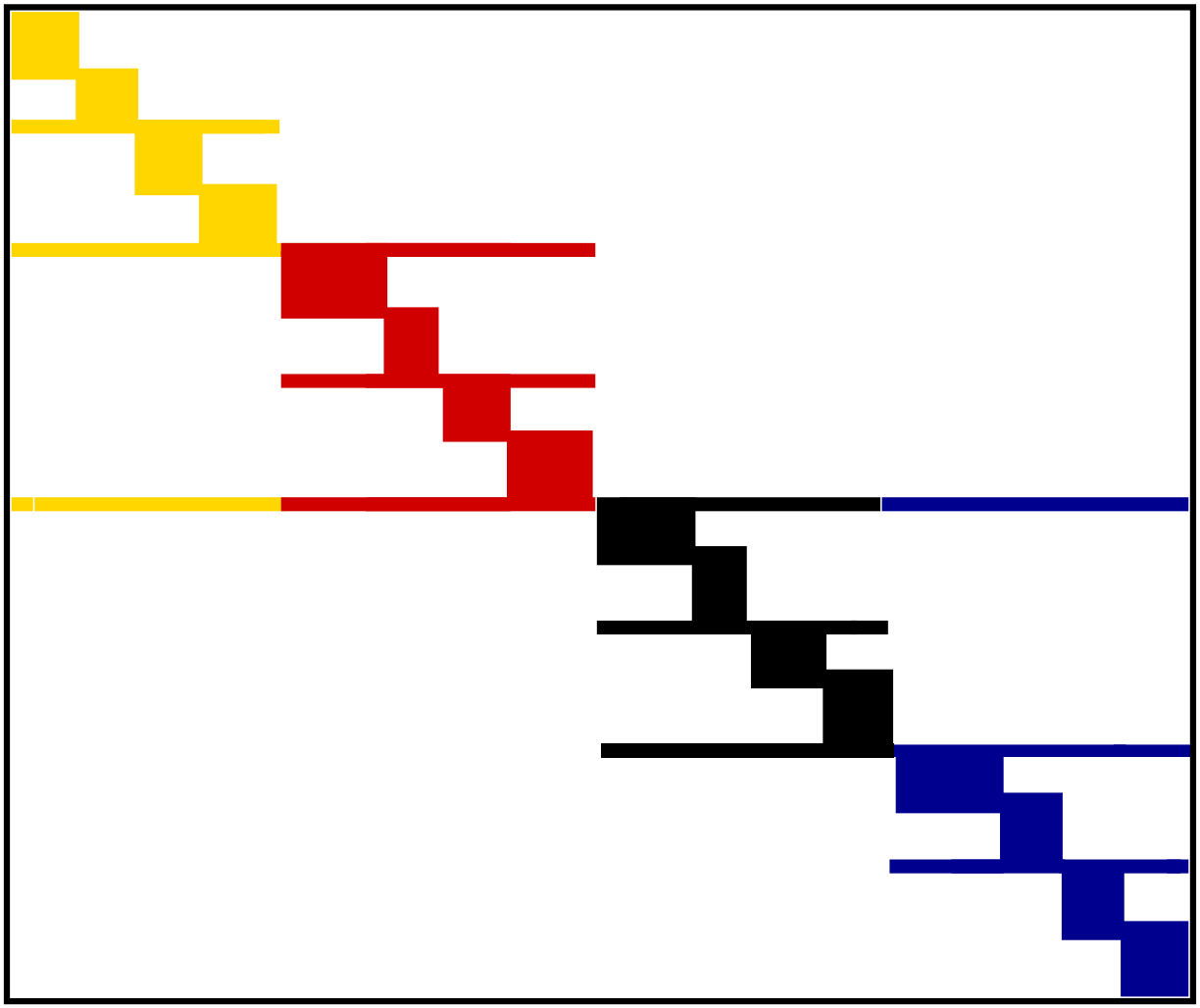


# Link matrix – 1D reordering

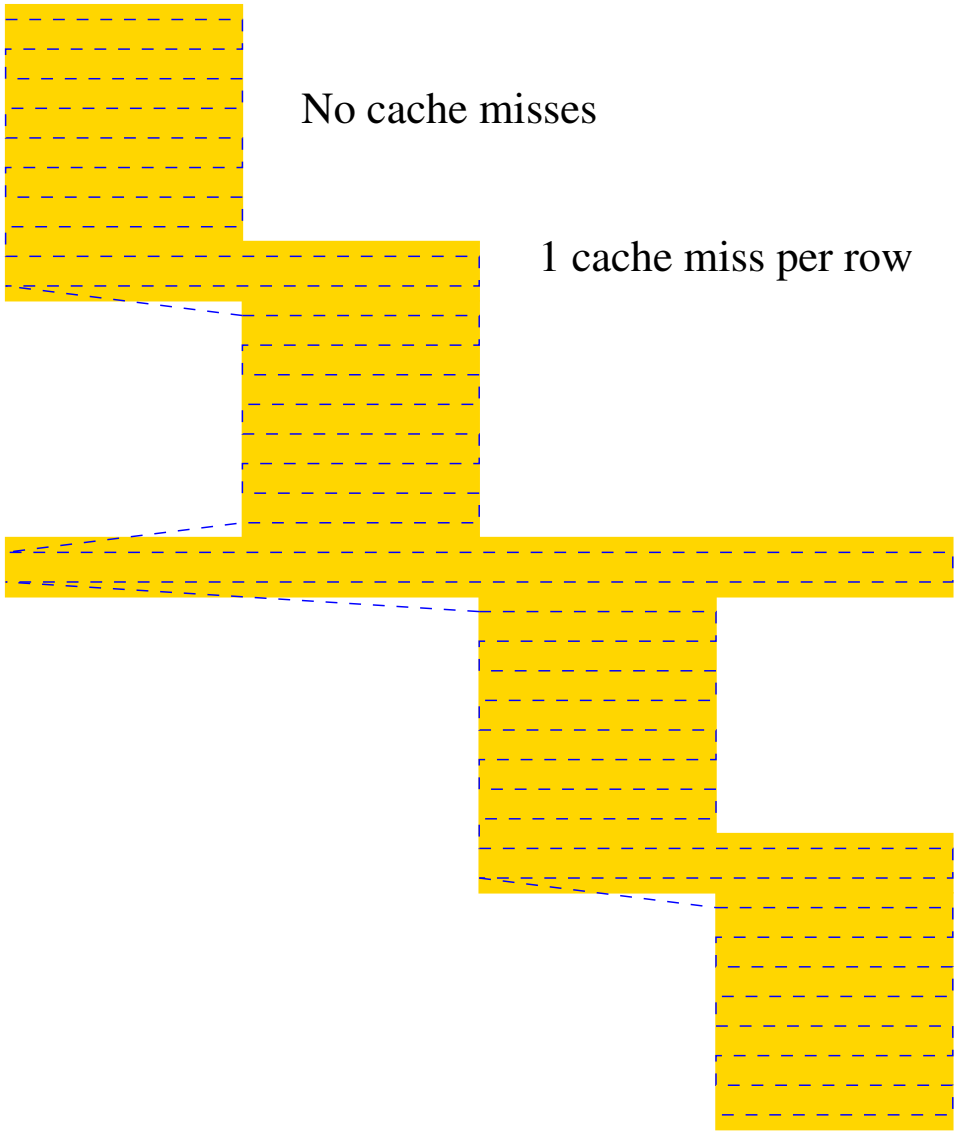


$p = 20, \quad \epsilon = 0.1$

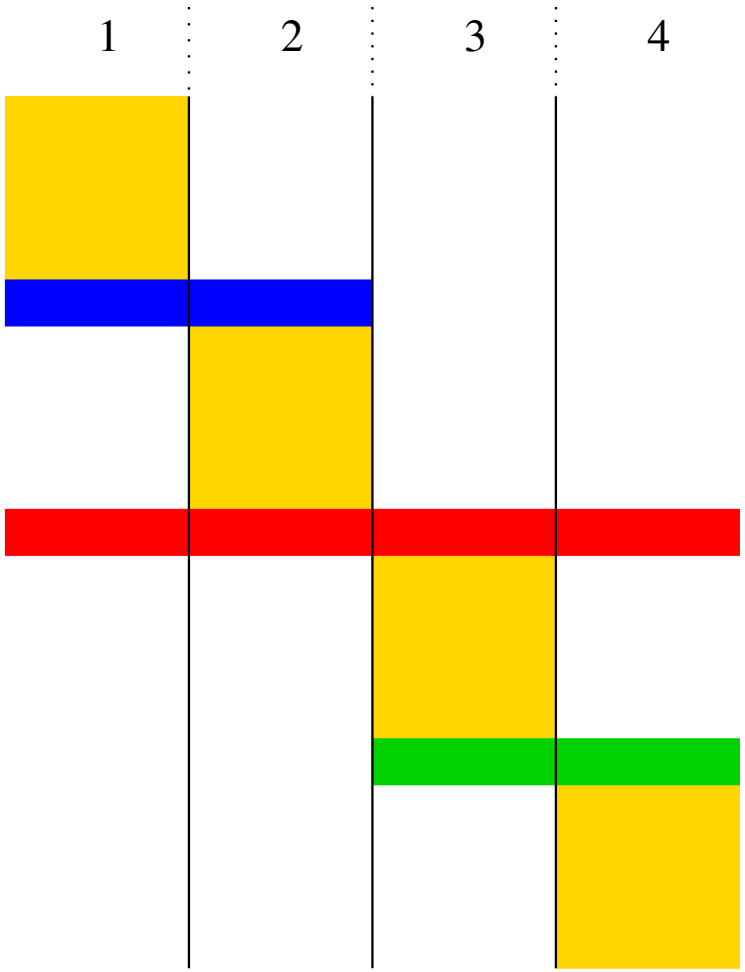
# Separated Block Diagonal form

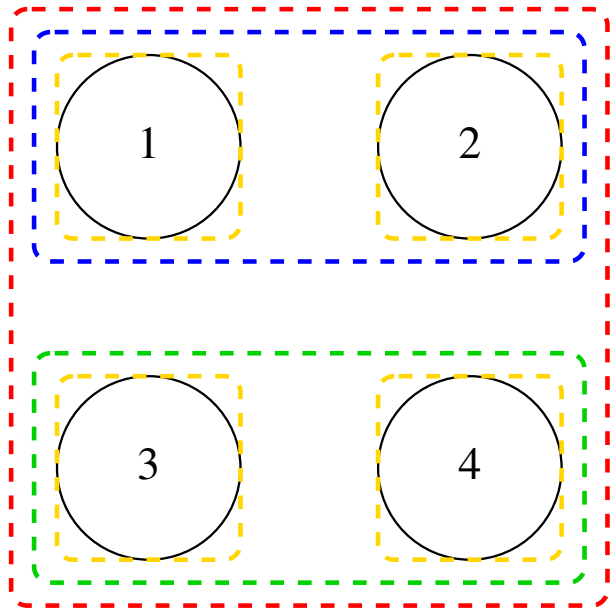


# Separated Block Diagonal form




# Separated Block Diagonal form




(Upper bound on) the number of cache misses: $\sum_i (\lambda_i - 1)$

# Separated Block Diagonal form

In 1D, row and column permutations bring the original matrix $A$ in *Separated Block Diagonal* (SBD) form as follows.

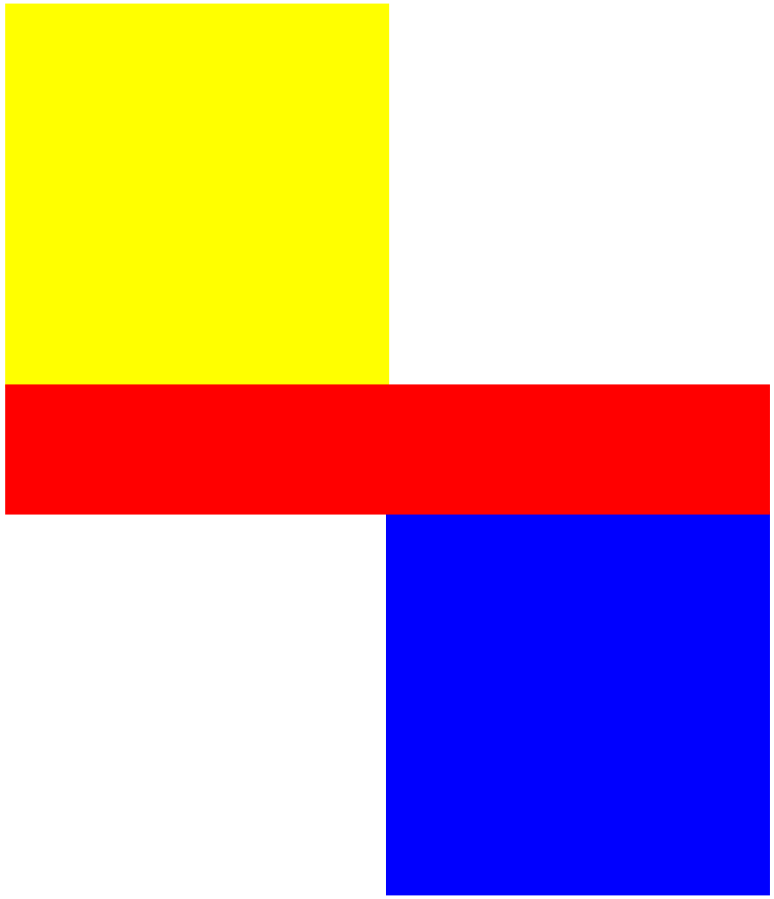
$A$ is modelled as a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N})$, with

- $\mathcal{V}$ the set of columns of $A$,
- $\mathcal{N}$ the set of *hyperedges*, each element is a subset of $\mathcal{V}$ and corresponds to a row of $A$.

A partitioning $\mathcal{V}_1, \mathcal{V}_2$ of $\mathcal{V}$ can be constructed; and from these, three hyperedge categories can be constructed:

- $\mathcal{N}_-^{\text{row}}$ as the set of hyperedges with vertices only in $\mathcal{V}_1$,
- $\mathcal{N}_c^{\text{row}}$ as the set of hyperedges with vertices both in $\mathcal{V}_1$ and $\mathcal{V}_2$,
- $\mathcal{N}_+^{\text{row}}$ the set of remaining hyperedges.

# Separated Block Diagonal form



$\mathcal{N}_-^{\text{row}}$

$\mathcal{N}_c^{\text{row}}$

$\mathcal{N}_+^{\text{row}}$

$\mathcal{V}_1$ $\mathcal{V}_2$

# Reordering parameters

$$\text{Taking } p = \frac{n}{S},$$

the number of cache misses is strictly bounded by

$$\sum_{i:\ n_i \in \mathcal{N}} (\lambda_i - 1);$$

taking $p \to \infty$ yields a *cache-oblivious* method with the same bound.

**References:**


- Yzelman and Bisseling, *Cache-oblivious sparse matrix-vector multiplication by using sparse matrix partitioning methods*, SIAM Journal on Scientific Computing, 2009


# Reordering parameters

The $(\lambda - 1)$ *metric* is already used extensively in parallel computing; in particular during *parallel* SpMV multiplication. Partitioners designed to that end, also take into account a *load-imbalance* $\epsilon$.

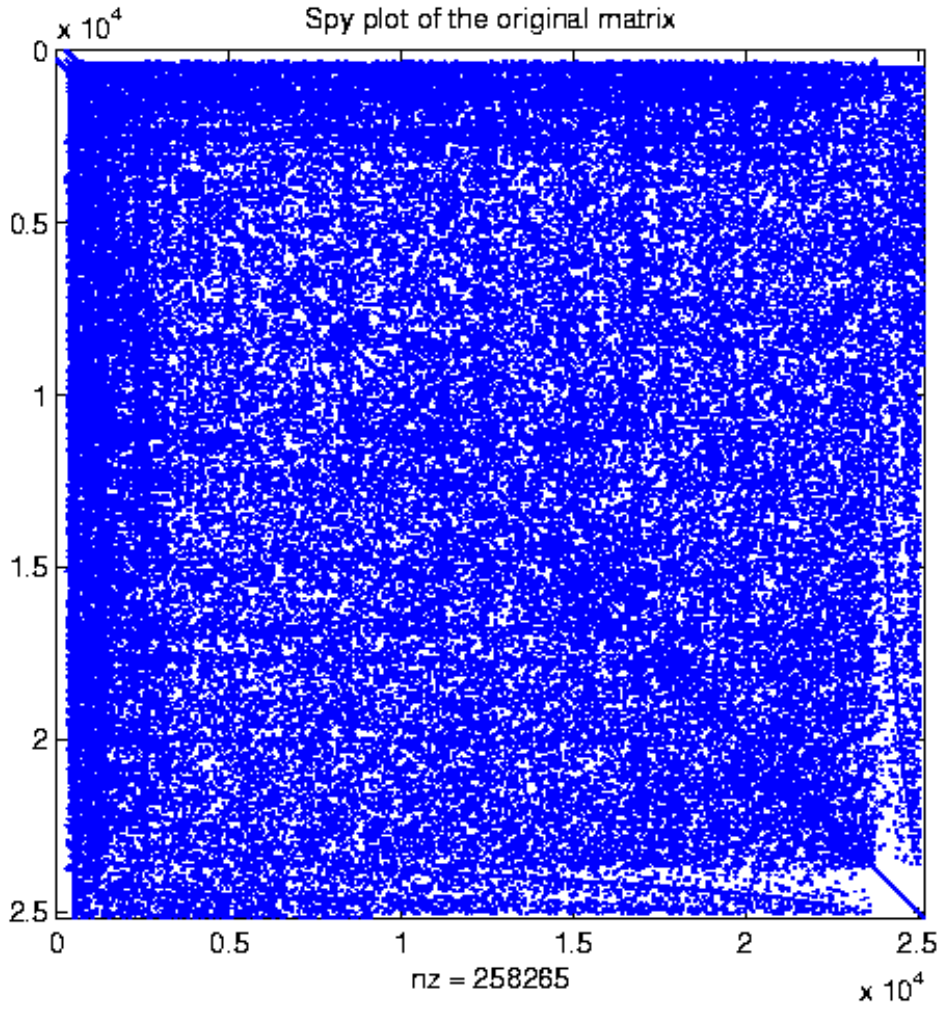
**References:**


- Çatalyürek and Aykanat, *Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication*, 1999
- Vastenhouw and Bisseling, *A two-dimensional data distribution method for parallel sparse matrix-vector multiplication*, 2005

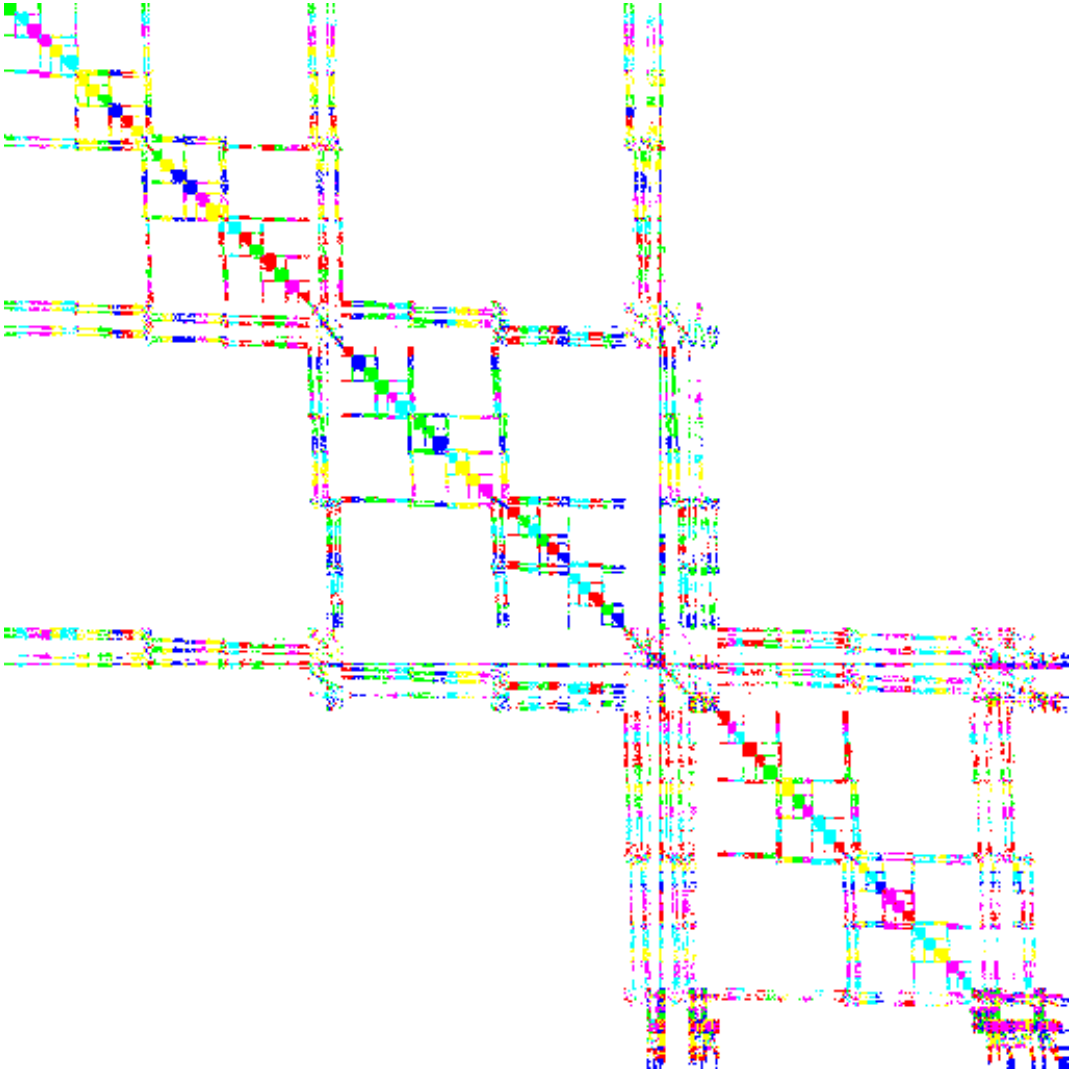

# Moving to two dimensions



# Chip industry

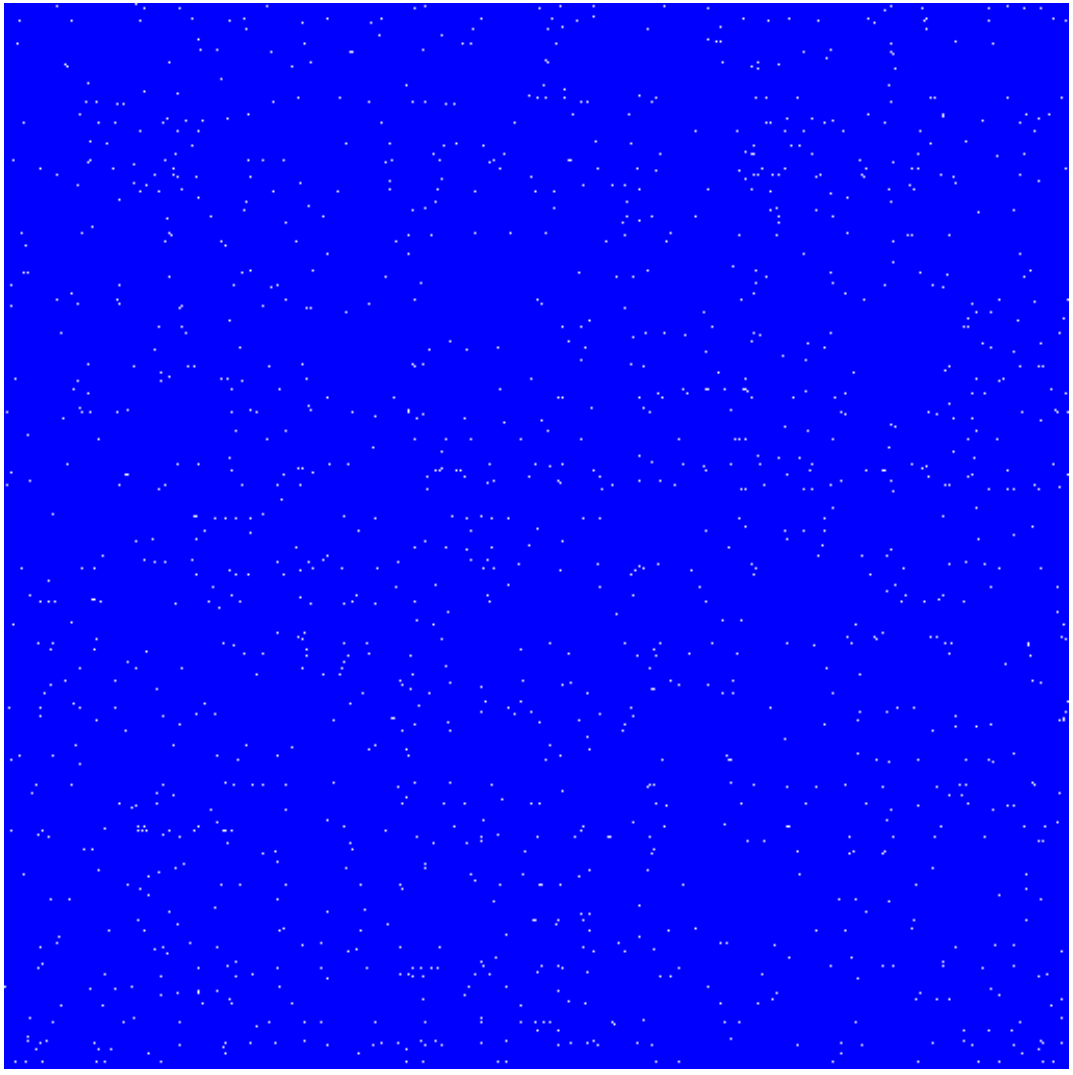


# Chip industry – 2D reordering

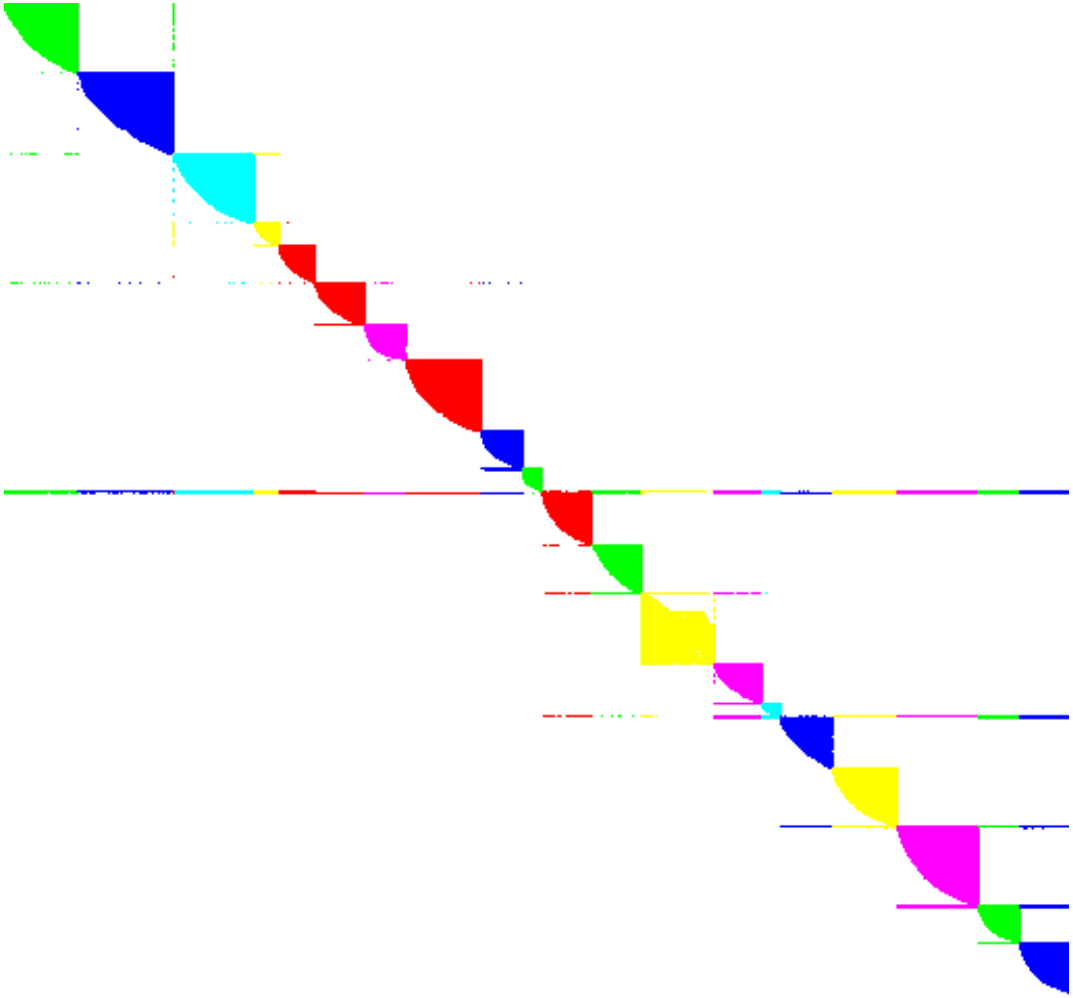


$p = 100, \quad \epsilon = 0.1$

# Link matrix

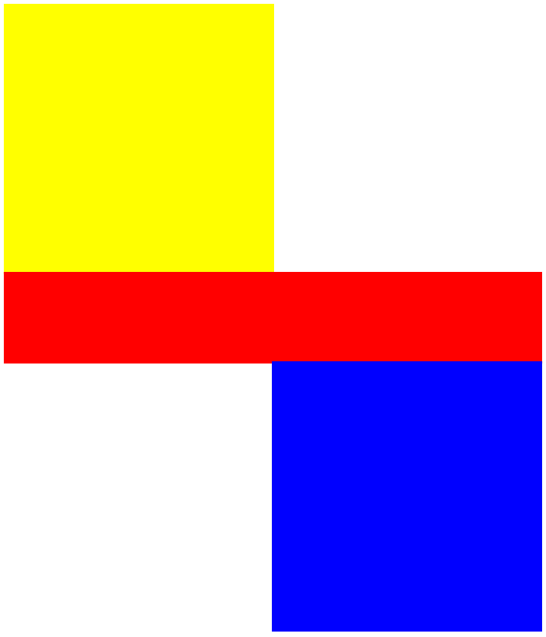


# Link matrix – 2D reordering

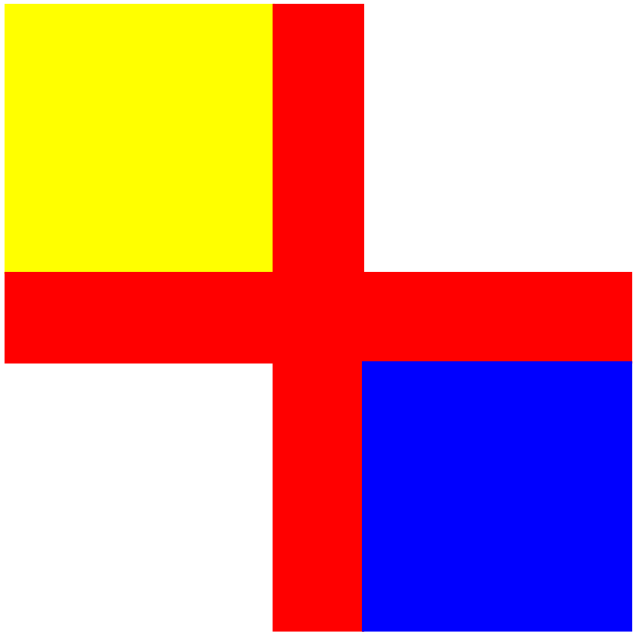


$p = 20, \quad \epsilon = 0.1$

# Two-dimensional SBD (doubly separated block diagonal)



**1D**

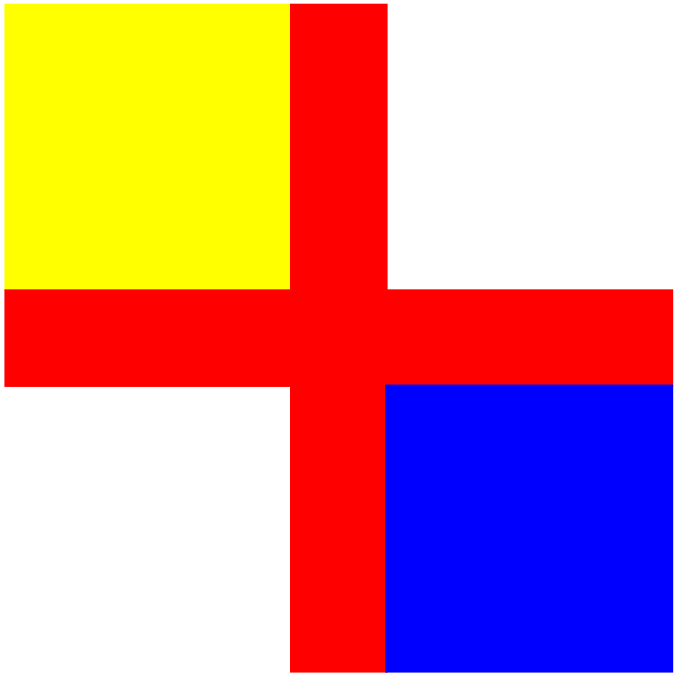


**2D**

Yzelman and Bisseling, *Two-dimensional cache-oblivious sparse matrix–vector multiplication*, April 2011 (Revised pre-print); `http://www.math.uu.nl/people/yzelman/publications/#pp`

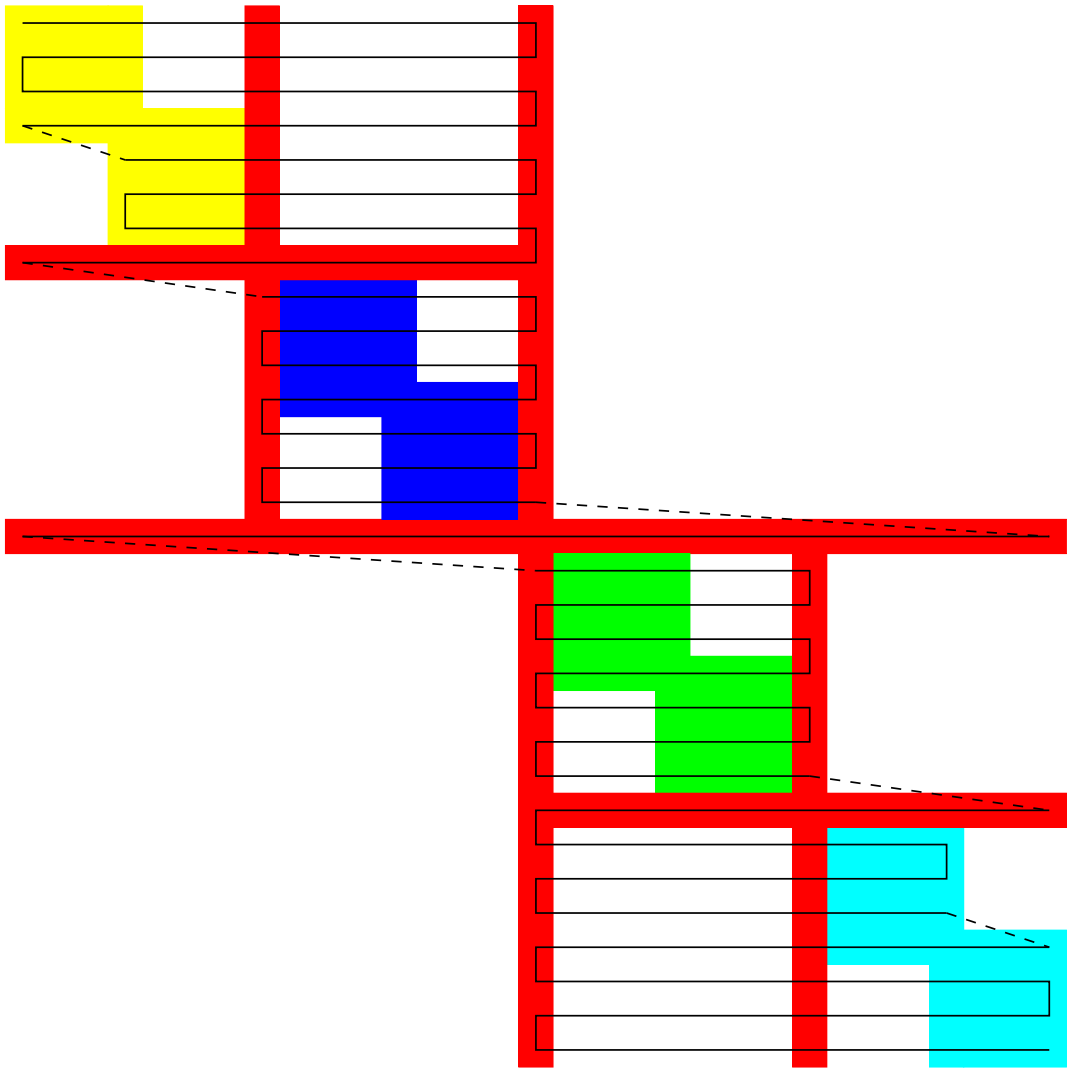
# Two-dimensional SBD (doubly separated block diagonal)

Using a fine-grain model of the input sparse matrix, individual nonzeros each correspond to a vertex;
*each row and column has a corresponding net.*



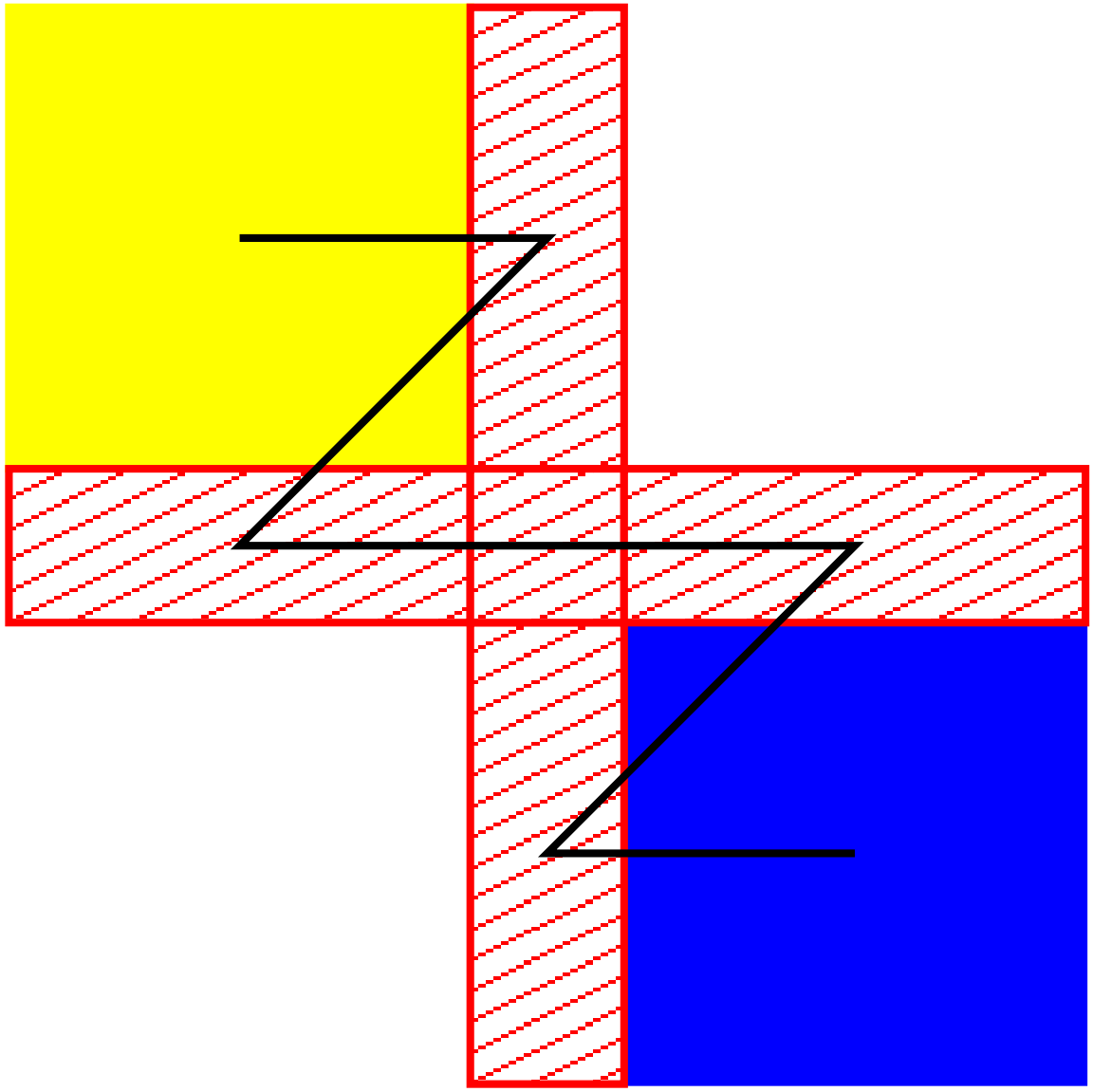

The quantity minimised remains $\sum_i(\lambda_i - 1)$.

# Two-dimensional SBD (doubly separated block diagonal)

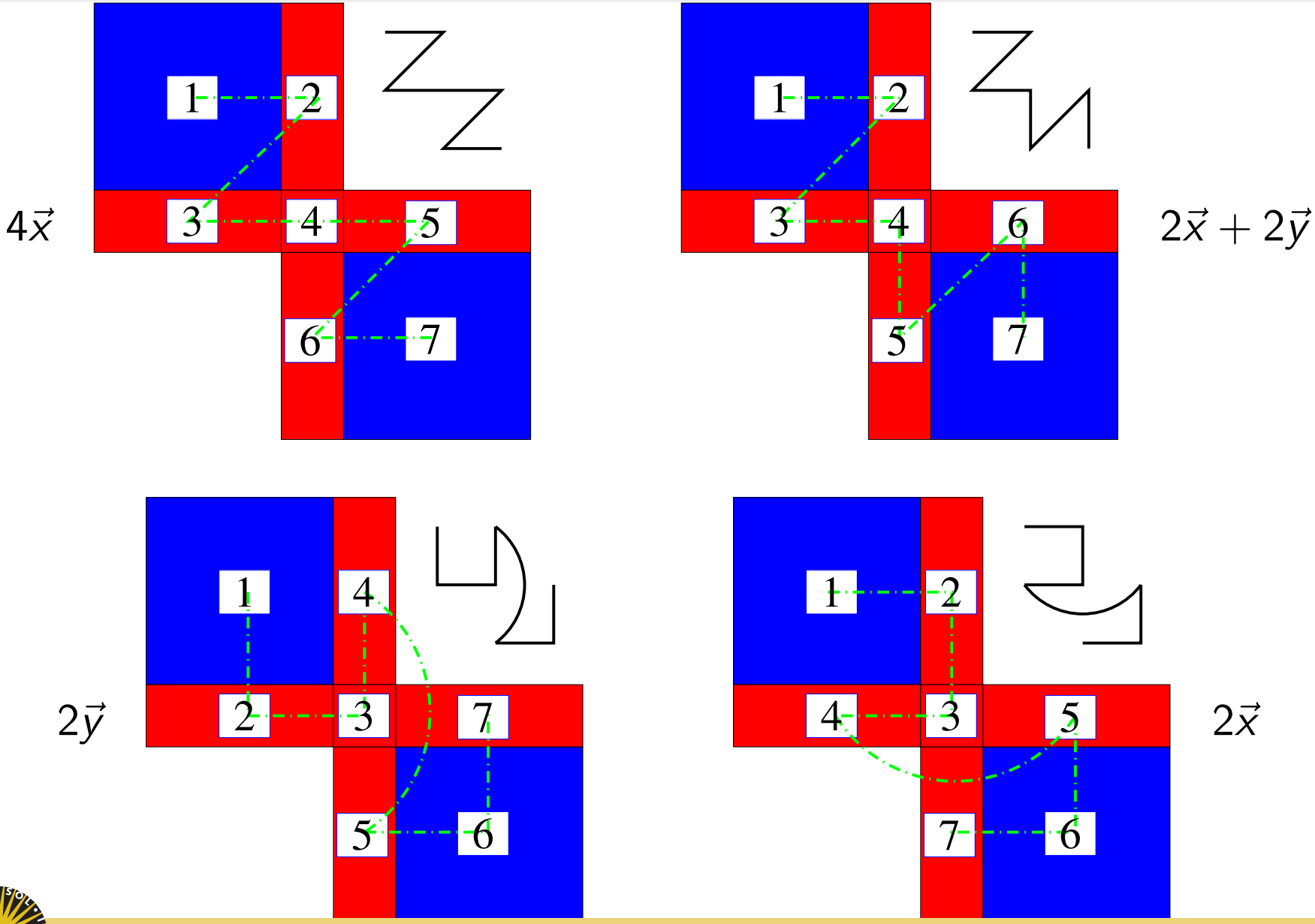


Zig-zag CRS is not suitable for handling 2D SBD!

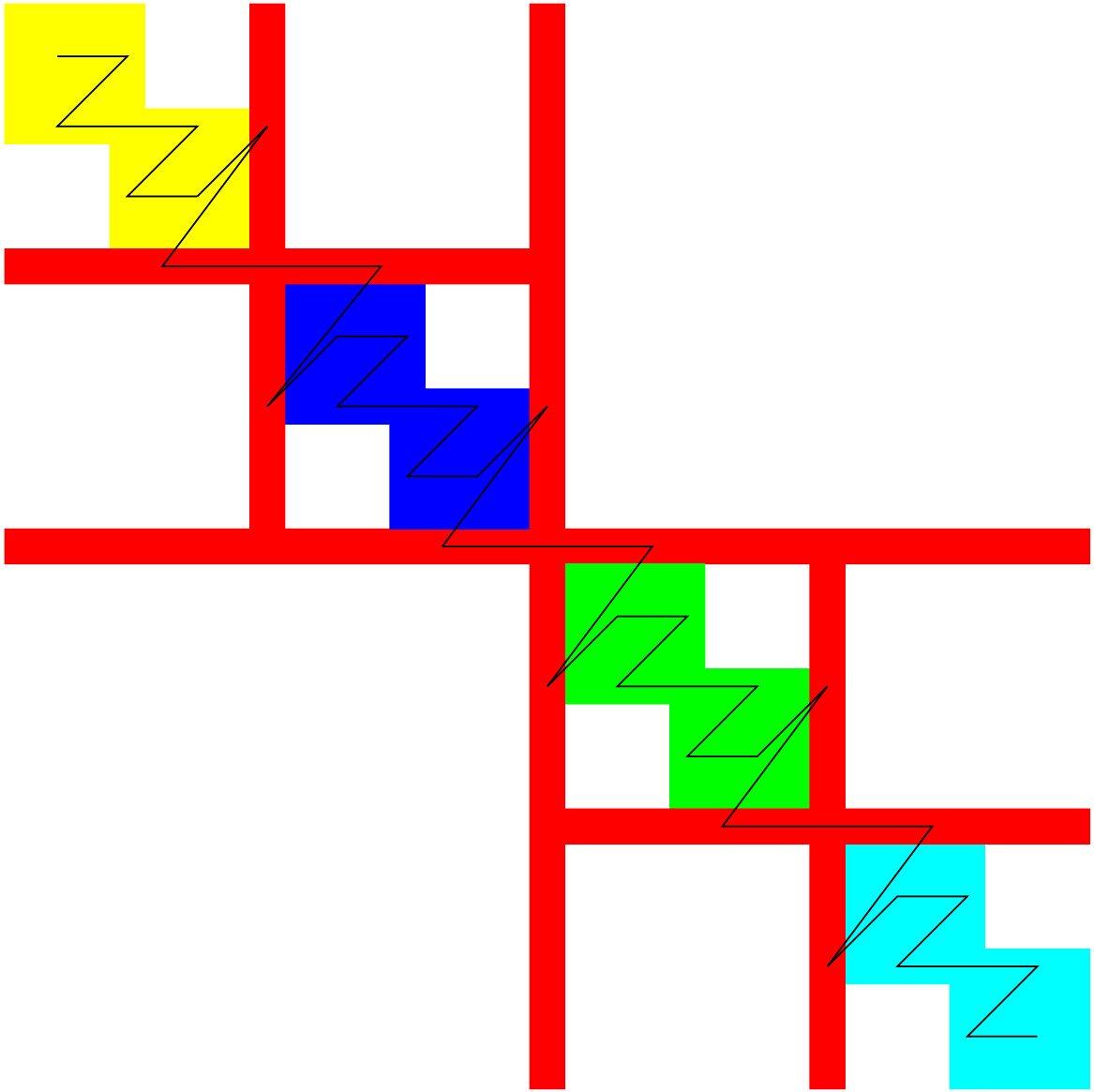
# Two-dimensional SBD; block ordering



# Two-dimensional SBD; block ordering



# Two-dimensional SBD; block ordering



# Pre-processing and SpMV times

| Matrix | Reordering time | SpMV time (old/1D/2D) |
|---|---|---|
| memplus, $p = 50$: | 4 seconds | (0.4 / 0.3 / 0.3 ms.) |
| rhpentium, $p = 50$: | 1 minute | (0.9 / 0.7 / 0.9 ms.) |
| cage14, $p = 10$: | 30 minutes | (111.6 / 130.4 / 130.4 ms.) |
| wiki2005, $p = 10$: | 2 hours | (347.4 / 212.5 / 136.7 ms.) |
| GL7d18, $p = 10$: | 2 hours | (780.3 / 552.5 / 549.5 ms.) |

*Old*: SpMV on the original matrix $A$
*1D*: SpMV on the 1D reordered matrix $PAQ$
*2D*: SpMV on the 2D reordered matrix $PAQ$

Black indicates use of a regular data structure, green the use of block ordering, blue the use of the OSKI auto-tuning library.

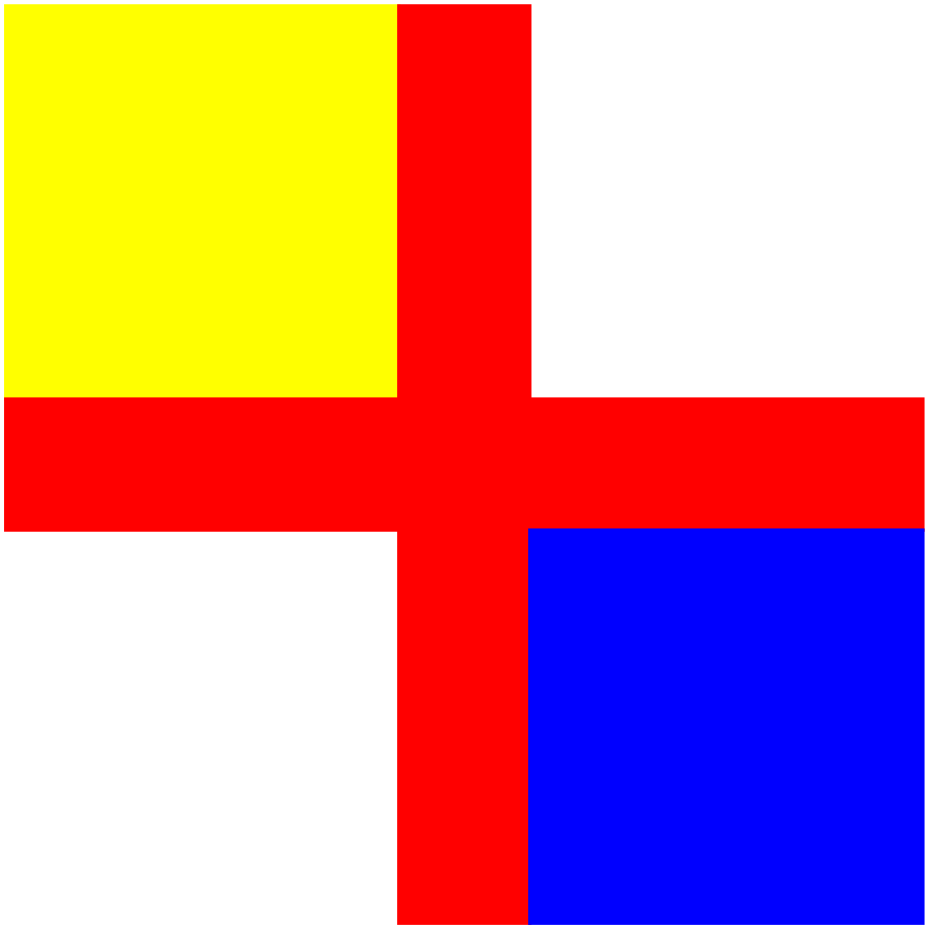
Results from 2011: reordering on an AMD Opteron 2378,
SpMV on an Intel Q6600

# Parallel cache-friendly SpMV



# On distributed-memory architectures

Directly use partitioner output:



# On distributed-memory architectures

Directly use partitioner output:

| Matrix | $p = 1$ | $p = 4$ | $p = 16$ | $p = 64$ |
|---|---|---|---|---|
| cage13 | 372.2 | 120.7 | 37.1 | *16.1* |
| stanford_berkeley | 552.6 | 169.3 | 71.2 | *21.4* |

Using the BSPOnMPI library with the parallel SpMV kernel from BSPedupack; three superstep algorithm with full synchronisations.

Bisseling, van Leeuwen, Çatalyürek, Fagginger Auer, Yzelman, *Two-dimensional approach to sparse matrix partitioning* in *Combinatorial Scientific Computing* by Olaf Schenk and Uwe Naumann (eds.)

# On shared-memory architectures

Directly use partitioner output:

| Matrix | sequential unordered | $p = 2$ | $p = 3$ | $p = 4$ |
|---|---|---|---|---|
| cage14 | 232.8 | 272.5 | *249.7* | 297.1 |
| wiki2005 | 564.2 | 285.3 | *244.5* | 255.0 |

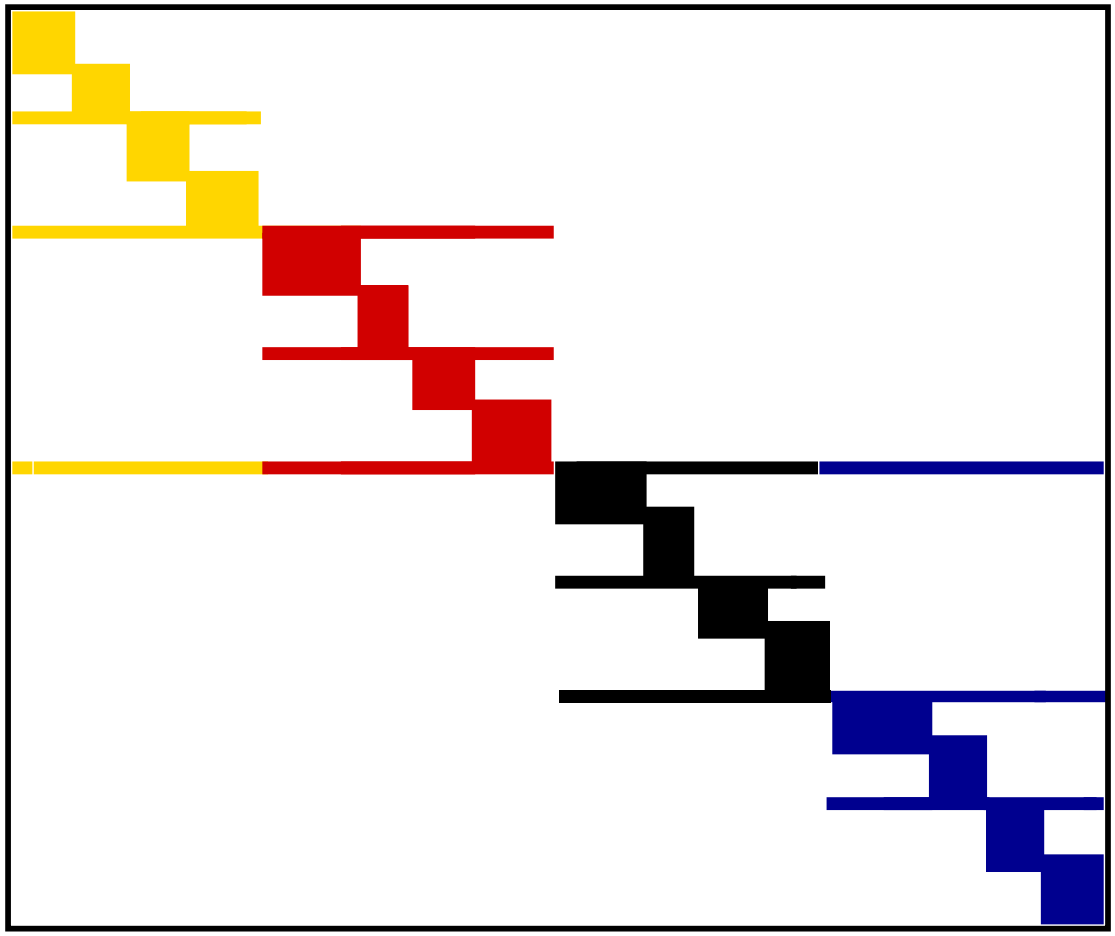
Using the Java MulticoreBSP library; two superstep algorithm with full synchronisation.

Yzelman and Bisseling, *An Object-Oriented BSP Library for Multicore Programming*, 2011 (Pre-print);
`http://www.math.uu.nl/people/yzelman/publications/#pp`

`http://www.multicorebsp.com`

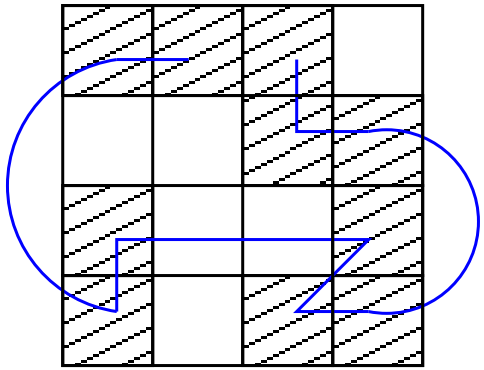
# On distributed-memory architectures

Use both partitioner and reordering output: partition for $p \to \infty$, but distribute only over the actual number of processors:



# Bi-directional Incremental CRS (BICRS)

$$A = \begin{pmatrix} 4 & 1 & 3 & 0 \\ 0 & 0 & 2 & 3 \\ 1 & 0 & 0 & 2 \\ 7 & 0 & 1 & 1 \end{pmatrix}$$



Stored as:

| | |
|---|---|
| nzs: | [3 2 3 1 1 2 1 7 4 1] |
| col_increment: | [2 4 1 4 -1 5 -3 4 4 1] |
| row_increment: | [0 1 2 -1 1 -3] |

, $2nnz + (row\_jumps + 1)$ accesses

Yzelman and Bisseling, *A cache-oblivious sparse matrix–vector multiplication scheme based on the Hilbert curve*, 2010 (Pre-print); `http://www.math.uu.nl/people/yzelman/publications/#pub`

# On shared-memory architectures

Multiple threads can work simultaneously on a BICRS representation of reordered matrices; smart synchronisation is required at the row-wise separators to prevent concurrent writes.

# Software

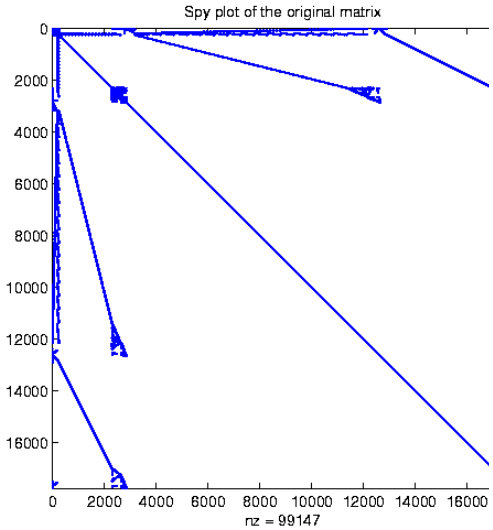
The latest version (3.11) of the sparse matrix partitioner software *Mondriaan* natively supports both the 1D and 2D reordering methods described here. This version has been released in December 2010, and can be found at:

`http://www.math.uu.nl/people/bisseling/Mondriaan`

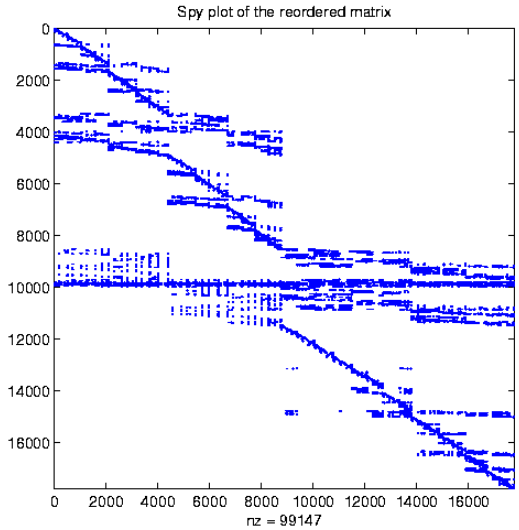
The plain SpMV kernels and block SpMV kernels used are freely available as well, and can be found at:

`http://www.math.uu.nl/people/yzelman/software`

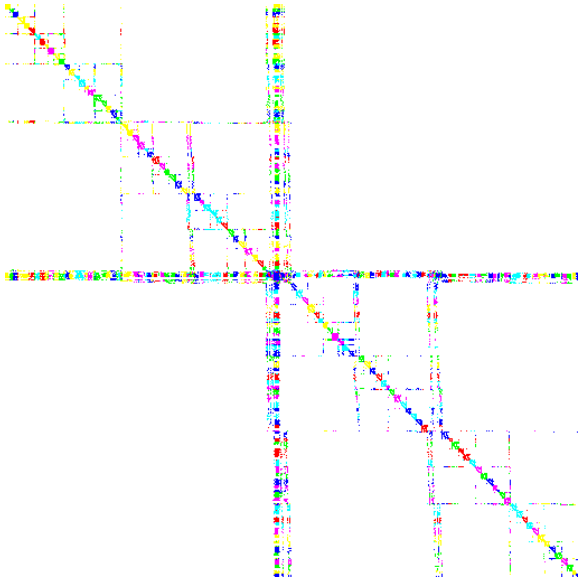
# The memplus matrix




Original

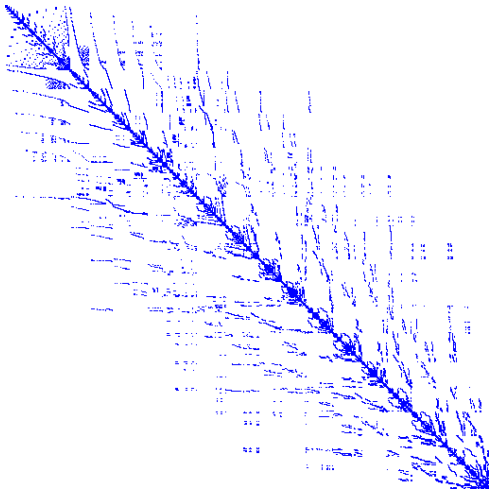



1D, $p = 100$, $\epsilon = 0.1$

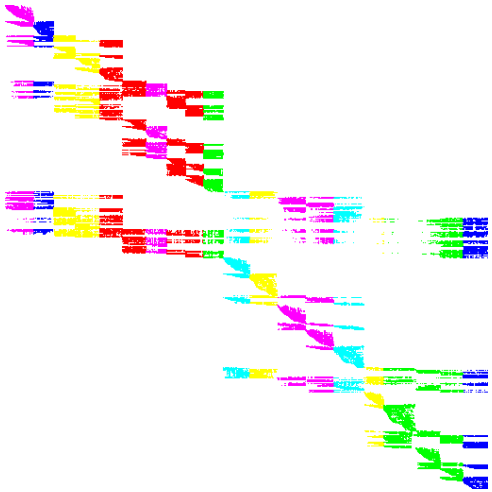


2D, $p = 100$, $\epsilon = 0.1$

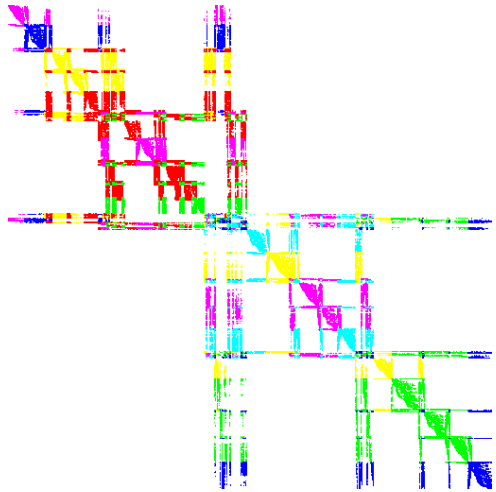
# The cage14 matrix



Original



1D ($p = 20$, $\epsilon = 0.1$)



Finegrain ($p = 20$, $\epsilon = 0.1$)