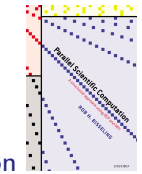
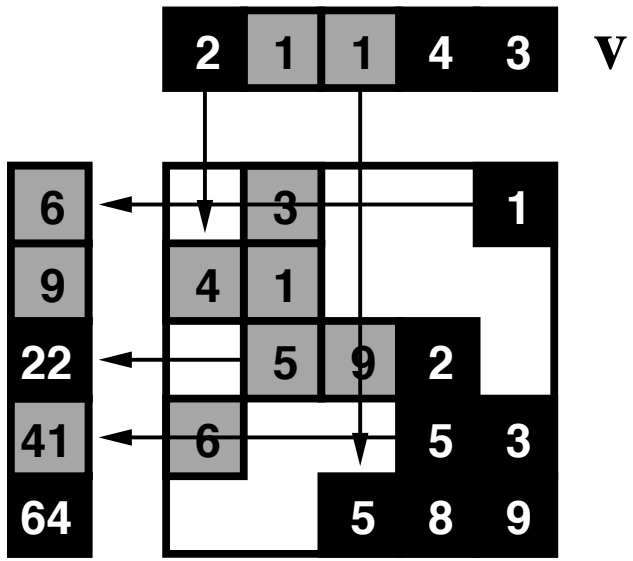
Mondriaan Sparse Matrix Distribution
(PSC §4.5)

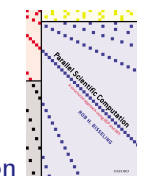


Sparse matrix–vector multiplication

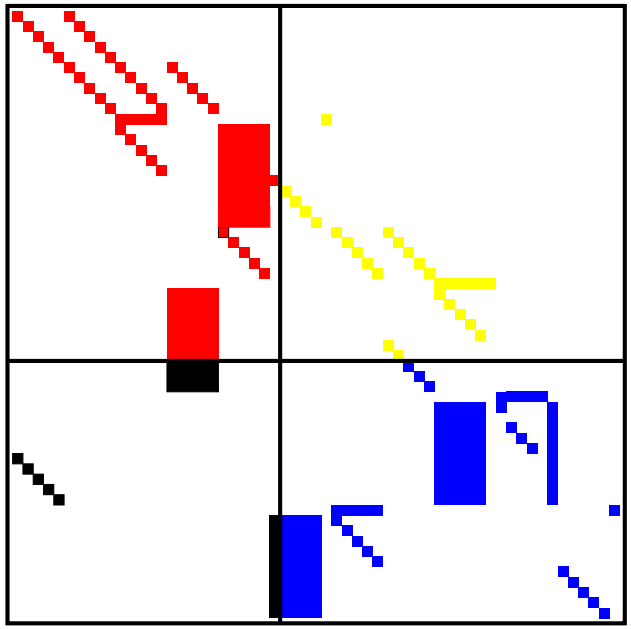
Parallel sparse matrix–vector multiplication $\mathbf{u} := A\mathbf{v}$.
A sparse $m \times n$ matrix, $\mathbf{u}$ dense m-vector, $\mathbf{v}$ dense n-vector.
Sequential computation $u_i := \sum_{j=0}^{n-1} a_{ij} v_j$.



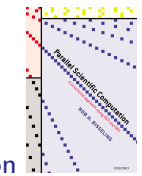
4 supersteps: communicate, compute, communicate, compute



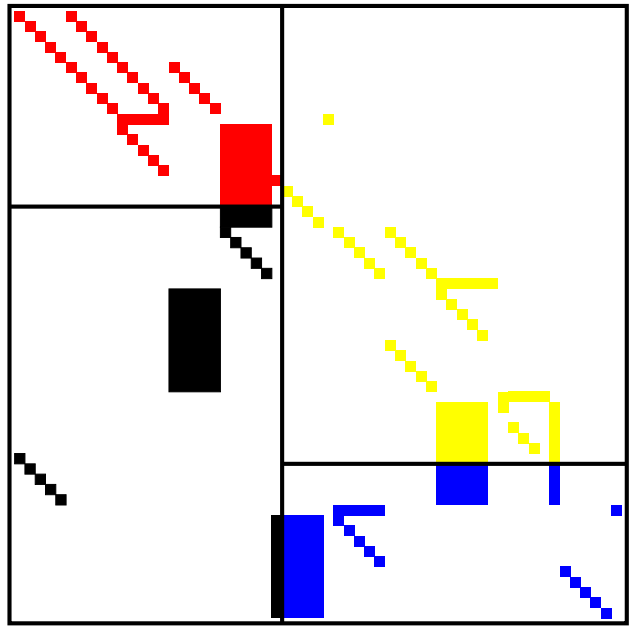
Cartesian matrix partitioning



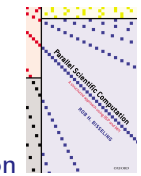
- Block distribution of 59×59 matrix `impcol_b` from Harwell–Boeing collection with 312 nonzeros, for $p = 4$
- #nonzeros per processor: 126, 28, 128, 30
- Each separate split has optimal balance (for blocks)



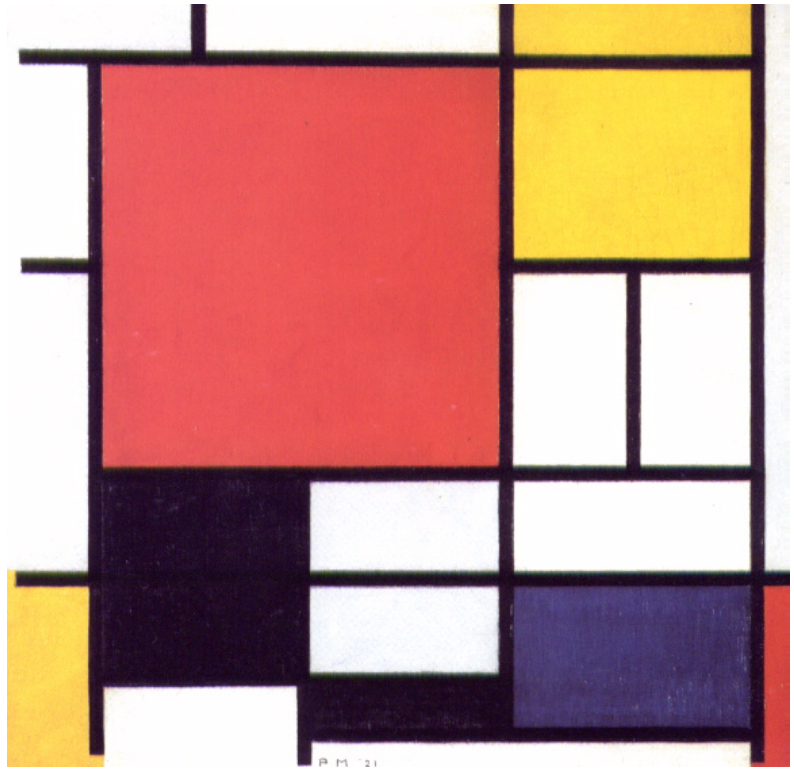
Non-Cartesian matrix partitioning



- Block distribution of 59×59 matrix `impcol_b` from Harwell–Boeing collection with 312 nonzeros, for $p = 4$
- #nonzeros per processor: 76, 76, 80, 80
- Each separate split has optimal balance (for blocks)

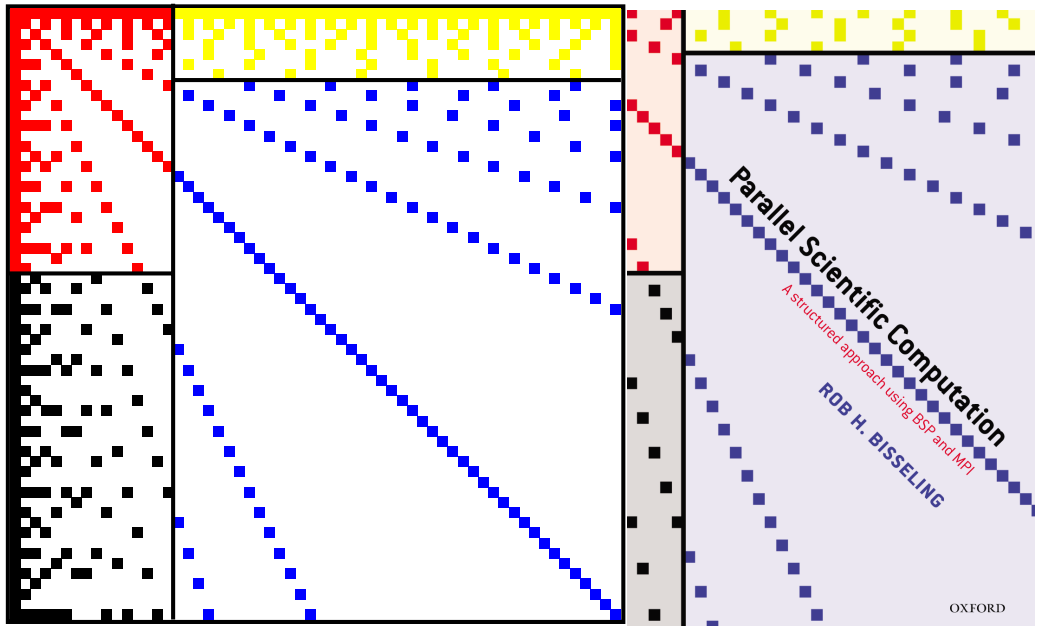


Composition with Red, Yellow, Blue and Black

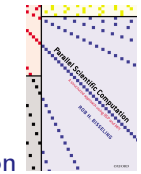


Piet Mondriaan 1921

Matrix prime60



- Non-Cartesian block distribution of 60×60 matrix prime60 with 462 nonzeros, for $p = 4$
- $a_{ij} \neq 0 \iff i|j \text{ or } j|i \qquad (1 \leq i, j \leq 60)$
  Exceptional numbering, starting at 1!



p-way matrix partitioning

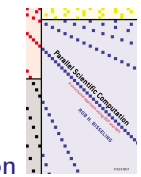
- Define

$$A_s = \{(i,j) : 0 \leq i,j < n \wedge \phi(i,j) = s\}$$

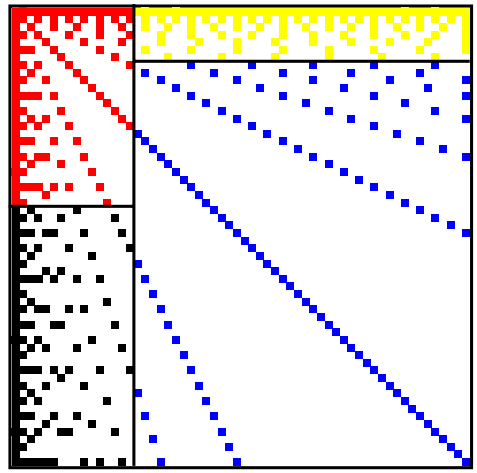
  as the set of index pairs corresponding to the nonzeros of processor $P(s)$, for $0 \leq s < p$.
- For the purpose of partitioning, we identify:
  - nonzero $\equiv$ index pair;
  - sparse matrix $\equiv$ set of index pairs.
- $A_0, \ldots, A_{p-1}$ forms a p-way partitioning of

$$A = \{(i,j) : 0 \leq i,j < n \wedge a_{ij} \neq 0\}.$$

- We use the notation $V(A_0, \ldots, A_{p-1}) = V_\phi$.

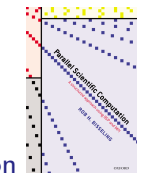


Communication volume for partitioned matrix



$$V(A_0, A_1, A_2, A_3) = V(A_0, A_1, A_2 \cup A_3) + V(A_2, A_3)$$

- $V(A_0, A_1, A_2, A_3)$ is the total matrix–vector communication volume corresponding to the partitioning A_0, A_1, A_2, A_3.
- $V(A_2, A_3)$ is the volume corresponding to the partitioning A_2, A_3 of the matrix $A_2 \cup A_3$.

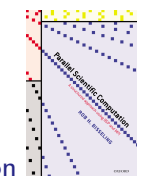


Motivation of the Mondriaan splitting

Theorem. Given an $m \times n$ sparse matrix A, and mutually disjoint subsets $A_0, \ldots, A_k$ of A, where $k \geq 1$, it holds that

$$V(A_0, \ldots, A_k) = V(A_0, \ldots, A_{k-2}, A_{k-1} \cup A_k) + V(A_{k-1}, A_k).$$

Meaning: k parts $\Rightarrow$ $k+1$ parts can be done locally, independently, by looking at just one split. This greedily minimises the total communication volume.



Proof of theorem

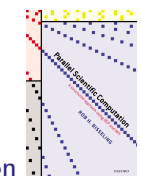
- For a given partitioning $A_0, \ldots, A_{k-1}$, let the number of processors that need a vector component v_j be $q_j = q_j(A_0, \ldots, A_{k-1})$. This equals the number of sets A_s that have a nonzero in matrix column j.
- Let the number of processors that contribute to a vector component u_i be $p_i = p_i(A_0, \ldots, A_{k-1})$.
- Let $p_i' = \max(p_i - 1, 0)$ and $q_j' = \max(q_j - 1, 0)$.
- Instead of proving

$$V(A_0, \ldots, A_k) = V(A_0, \ldots, A_{k-2}, A_{k-1} \cup A_k) + V(A_{k-1}, A_k),$$

it is sufficient to prove for all i that

$$p_i'(A_0, \ldots, A_k) = p_i'(A_0, \ldots, A_{k-2}, A_{k-1} \cup A_k) + p_i'(A_{k-1}, A_k).$$

Similar for q_j'. Result follows from $V = \sum_i p_i' + \sum_j q_j'$.



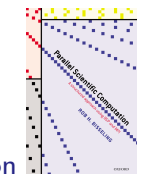
Proof of theorem (cont'd)

- $p_i = \#$ sets A_s that have a nonzero in matrix row i.
- If row i has a nonzero in $A_{k-1} \cup A_k$, then $p'_i = p_i - 1$ in all three terms. Thus,

$$\begin{aligned} & p'_i(A_0, \ldots, A_{k-2}, A_{k-1} \cup A_k) + p'_i(A_{k-1}, A_k) \\ &= p_i(A_0, \ldots, A_{k-2}, A_{k-1} \cup A_k) - 1 + p_i(A_{k-1}, A_k) - 1 \\ &= p_i(A_0, \ldots, A_{k-2}) + 1 - 1 + p_i(A_{k-1}, A_k) - 1 \\ &= p_i(A_0, \ldots, A_{k-2}) + p_i(A_{k-1}, A_k) - 1 \\ &= p_i(A_0, \ldots, A_k) - 1 = p'_i(A_0, \ldots, A_k). \end{aligned}$$

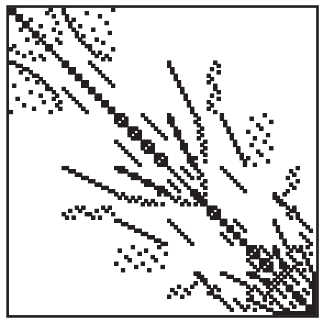
- If row i has no nonzero in $A_{k-1} \cup A_k$, then both A_{k-1} and A_k are empty, so that

$$\begin{aligned} & p'_i(A_0, \ldots, A_{k-2}, A_{k-1} \cup A_k) + p'_i(A_{k-1}, A_k) \\ &= p'_i(A_0, \ldots, A_{k-2}) + 0 = p'_i(A_0, \ldots, A_k). \quad \square \end{aligned}$$



Computational load balance

- Paint all nonzeros black:

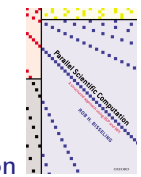


No communication, but no parallelism. No pain, no gain!

- A load balance criterion must therefore be satisfied:

$$\max_{0 \leq s < p} nz(A_s) \leq (1 + \epsilon) \frac{nz(A)}{p}.$$

- ϵ is specified allowable imbalance;
  ϵ' is imbalance achieved by partitioning.



BSP cost determines ϵ

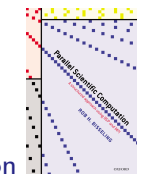
- Best choice of ϵ is machine-dependent and can be found by using the BSP model.
- Communication cost is $\frac{Vg}{p}$, assuming communication is balanced by subsequent vector partitioning.
- Total BSP cost is

$$2(1+\epsilon')\frac{nz(A)}{p} + \frac{Vg}{p} + 4l.$$

- To get a good trade-off between computation imbalance and communication, we require

$$2\epsilon'\frac{nz(A)}{p} \approx \frac{Vg}{p}, \quad \text{i.e.,} \quad \epsilon' \approx \frac{Vg}{2nz(A)}.$$

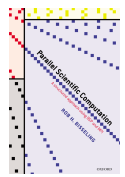
- If necessary, we adjust ϵ and run the partitioner again.



Bipartitioning: splitting into 2 parts

$$A = \begin{bmatrix} 0 & 3 & 0 & 0 & 1 \\ 4 & 1 & 0 & 0 & 0 \\ 0 & 5 & 9 & 2 & 0 \\ 6 & 0 & 0 & 5 & 3 \\ 0 & 0 & 5 & 8 & 9 \end{bmatrix}.$$

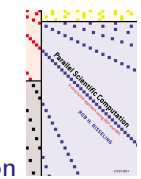
- The number of possible 2-way partitionings is $2^{nz(A)-1} = 2^{12} = 4096$. (Symmetry saved a factor of 2.)
- Finding the best solution by enumeration, trying all possibilities and choosing the best, works only for small problems. Thus, we need heuristic methods.
- Splitting by columns restricts the search space to $2^{n-1} = 2^4 = 16$ possibilities. An optimal column split for $\epsilon = 0.1$ is $\{0, 1, 2\} \mid \{3, 4\}$, with $V = 4$.



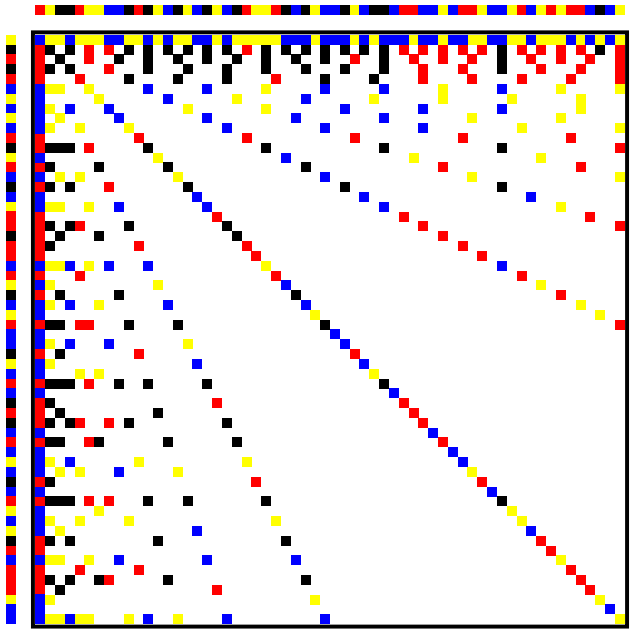
Repeated splits

- The partitioning starts with a complete matrix, splits it into 2 submatrices, splits each submatrix, giving 4 submatrices, and so on. The method can be formulated recursively. For simplicity, we assume that $p = 2^q$.
- Rows and columns in the submatrix need not be consecutive.
- The recursion level of a submatrix is the number of times the original matrix must be split to reach the submatrix. The level of the original matrix is 0.
- The final result for processor $P(s)$ is a submatrix defined by an index set $\bar{I}_s \times \bar{J}_s$. The sets are mutually disjoint.
- Removing empty rows and columns from $\bar{I}_s \times \bar{J}_s$ gives $I_s \times J_s$. Thus

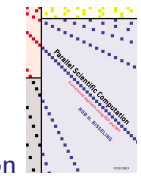
$$A_s \subset I_s \times J_s \subset \bar{I}_s \times \bar{J}_s.$$



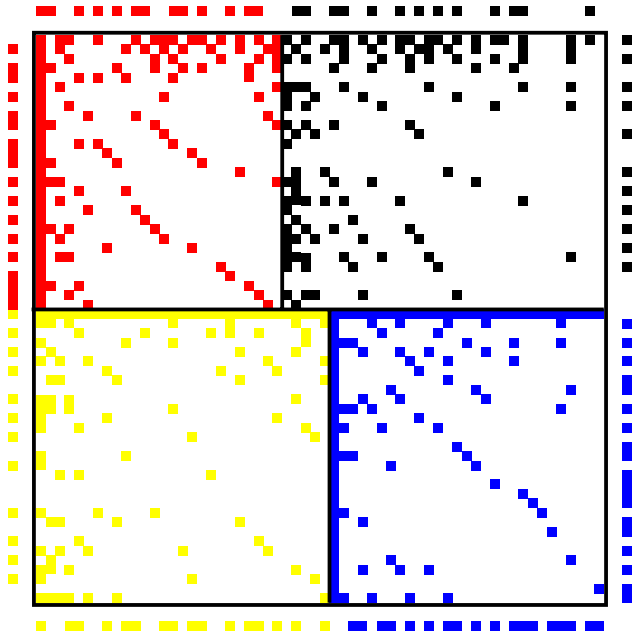
Global view of matrix prime60



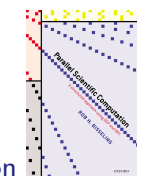
- Distribution of 60×60 matrix prime60 with 462 nonzeros, for $p = 4$, obtained by Mondriaan partitioning with $\epsilon = 3\%$.
- Maximum number of nonzeros per processor is 117; average is 462/4=115.5. Achieved imbalance is $\epsilon' \approx 1.3\%$.
- Communication volume is: fanout 51; fanin 47; $V = 98$.



Local view of matrix prime60

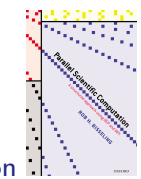


- The local submatrix $\bar{I}_s \times \bar{J}_s$ of processor $P(s)$ has size:
  - 29×26 for $P(0)$; 29×34 for $P(1)$
  - 31×31 for $P(2)$; 31×29 for $P(3)$
- Note that $\bar{I}_1 \times \bar{J}_1$ has 6 empty rows and 9 empty columns, giving a size of 23×25 for $I_1 \times J_1$.



Growth of load imbalance by splitting

- If the growth factor at each recursion level is $1 + \delta$, the overall growth factor is $(1 + \delta)^q \approx 1 + q\delta$. Here, $p = 2^q$. This motivates starting with $q\delta = \epsilon$, i.e., $\delta = \epsilon/q$.
- After the first split, one part has at least half the nonzeros, and the other part at most half. We recompute the ϵ values for both halves based on the new situation.
- The less-loaded processor can increase the allowed load imbalance to reduce communication.



Recursive, adaptive bipartitioning algorithm

MatrixPartition(A, p, ϵ)

input: $p = 2^q$, $\epsilon =$ allowed load imbalance, $\epsilon > 0$.

output: p-way partitioning of A with imbalance $\leq \epsilon$.

if $p > 1$ **then**

$maxnz := (1 + \epsilon)\frac{nz(A)}{p}$;

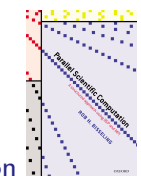
$(B_0^{\text{row}}, B_1^{\text{row}}) := split(A, \text{row}, \frac{\epsilon}{q})$;

$(B_0^{\text{col}}, B_1^{\text{col}}) := split(A, \text{col}, \frac{\epsilon}{q})$;

if $V(B_0^{\text{row}}, B_1^{\text{row}}) \leq V(B_0^{\text{col}}, B_1^{\text{col}})$ **then**

$(B_0, B_1) := (B_0^{\text{row}}, B_1^{\text{row}})$;

else $(B_0, B_1) := (B_0^{\text{col}}, B_1^{\text{col}})$;



Recursive, adaptive bipartitioning algorithm

MatrixPartition(A, p, ϵ)

input: $p = 2^q$, $\epsilon =$ allowed load imbalance, $\epsilon > 0$.

output: p-way partitioning of A with imbalance $\leq \epsilon$.

if $p > 1$ **then**

$maxnz := (1 + \epsilon)\frac{nz(A)}{p}$;

$(B_0^{\text{row}}, B_1^{\text{row}}) := split(A, \text{row}, \frac{\epsilon}{q})$;

$(B_0^{\text{col}}, B_1^{\text{col}}) := split(A, \text{col}, \frac{\epsilon}{q})$;

if $V(B_0^{\text{row}}, B_1^{\text{row}}) \leq V(B_0^{\text{col}}, B_1^{\text{col}})$ **then**

$(B_0, B_1) := (B_0^{\text{row}}, B_1^{\text{row}})$;

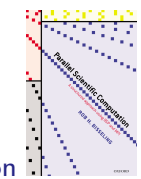
else $(B_0, B_1) := (B_0^{\text{col}}, B_1^{\text{col}})$;

$\epsilon_0 := \frac{maxnz}{nz(B_0)} \cdot \frac{p}{2} - 1$; $\epsilon_1 := \frac{maxnz}{nz(B_1)} \cdot \frac{p}{2} - 1$;

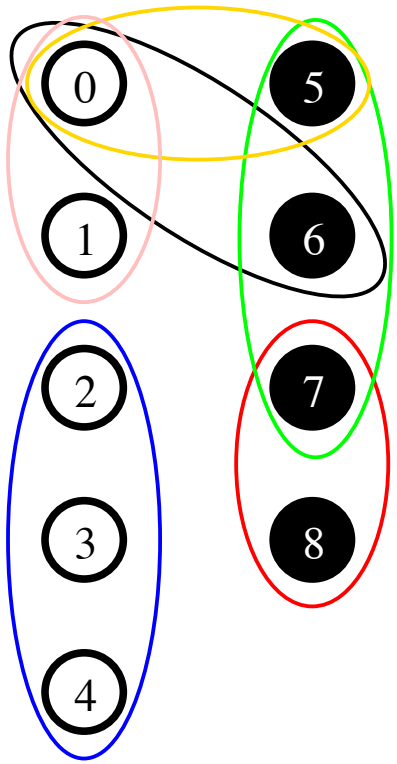
$(A_0, \ldots, A_{p/2-1}) :=$ MatrixPartition$(B_0, \frac{p}{2}, \epsilon_0)$;

$(A_{p/2}, \ldots, A_{p-1}) :=$ MatrixPartition$(B_1, \frac{p}{2}, \epsilon_1)$;

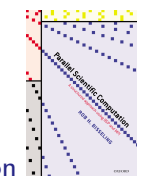
else $A_0 := A$;



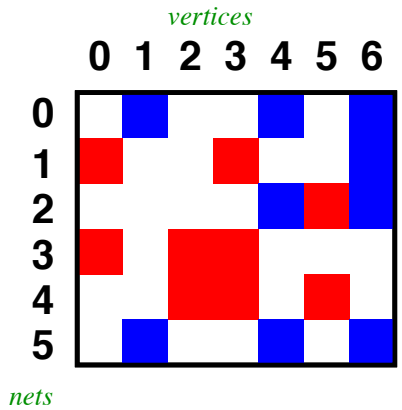
Hypergraph



Hypergraph with 9 vertices and 6 hyperedges (nets), partitioned over 2 processors



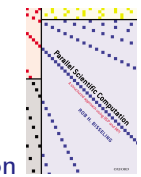
The magic *split* function



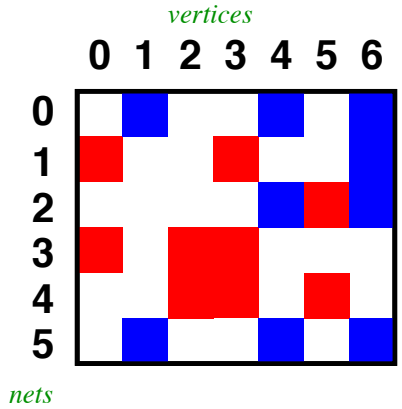
Magic column bipartitioning of $m \times n$ matrix

- Hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N}) \Rightarrow$ exact communication volume.
- Columns $\equiv$ Vertices: $0, 1, 2, 3, 4, 5, 6$.
  Rows $\equiv$ Hyperedges (nets, subsets of $\mathcal{V}$).
  Net $n_i = \{j : 0 \leq j < n \wedge a_{ij} \neq 0\}$:

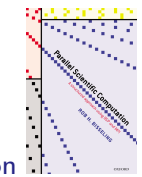
$$n_0 = \{1, 4, 6\}, \quad n_1 = \{0, 3, 6\}, \quad n_2 = \{4, 5, 6\},$$
$$n_3 = \{0, 2, 3\}, \quad n_4 = \{2, 3, 5\}, \quad n_5 = \{1, 4, 6\}.$$



Minimising communication volume



- Broken nets: n_1, n_2 cause one horizontal communication.
- Use Kernighan–Lin algorithm for hypergraph bipartitioning: try to improve initial random partitioning by moving vertices (columns) to the other part.
- The vertex with the largest gain (communication reduction) is moved. If the best possible move increases the communication, it is still accepted.
- Several passes are carried out. Vertices are never moved twice in a pass. Best solution encountered is kept.

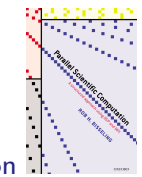


Multilevel scheme

1. Merge similar columns in pairs to reduce the problem size, and repeat this until the problem is small:

$$\begin{bmatrix} \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & \cdot & \cdot & \cdot & 1 & \cdot & 1 & \cdot \\ 1 & 1 & 1 & 1 & \cdot & \cdot & \cdot & 1 \\ \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 \\ \cdot & \cdot & 1 & 1 & \cdot & \cdot & 1 & 1 \end{bmatrix} \xrightarrow{merge} \begin{bmatrix} 1 & \cdot & \cdot & \cdot \\ 1 & \cdot & 1 & 1 \\ 1 & 1 & \cdot & 1 \\ 1 & 1 & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & 1 \\ \cdot & 1 & \cdot & 1 \end{bmatrix}$$

2. Bipartition the smaller problem using Kernighan–Lin with improved implementation by Fiduccia and Mattheyses.
3. Refine the bipartitioning using a simplified KLFM scheme.

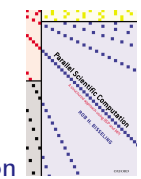


Communication volume and time: 1D vs. 2D

(Vastenhouw and Bisseling, *SIAM Review* **47** (2005) pp.67–95.)

p	Volume (in data words)			Time (in ms)		
	1D row	1D col	2D	1D row	1D col	2D
1	0	0	0	67.55	67.61	74.15
2	15764	24463	15764	36.65	32.26	32.16
4	42652	54262	30444	14.06	12.22	12.14
8	90919	96038	49120	6.49	6.35	6.62
16	177347	155604	75884	5.22	4.22	4.20
32	297658	227368	106563	4.32	4.08	3.23

Term-by-document matrix `tbdlinux`:
112,757 rows; 20,167 columns; 2,157,675 nonzeros.
Timings obtained on an SGI Origin 3800.



Summary

- We have derived a recursive partitioning algorithm for a sparse matrix. It is greedy (minimises splits separately without looking ahead) and adapts the allowed load imbalance to the current partitioning.
- The result is a p-way matrix partitioning $A_0, \ldots, A_{p-1}$ with

$$A_s \subset I_s \times J_s \subset \bar{I}_s \times \bar{J}_s.$$

- A hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ is a generalisation of a graph. It consists of a set of vertices $\mathcal{V}$ and a set of hyperedges, or nets, $\mathcal{N}$, which are subsets of $\mathcal{V}$.
- Multilevel methods for hypergraph partitioning find good splits of a sparse matrix in reasonable time.