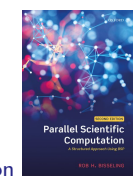
Mondriaan Sparse Matrix Distribution

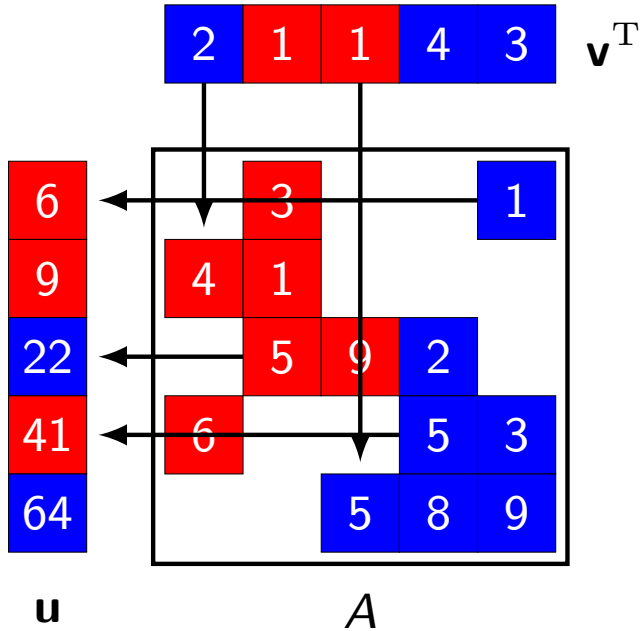
Section 4.5 of Parallel Scientific Computation, 2nd edition

Rob H. Bisseling

Utrecht University

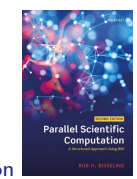


Parallel sparse matrix–vector multiplication

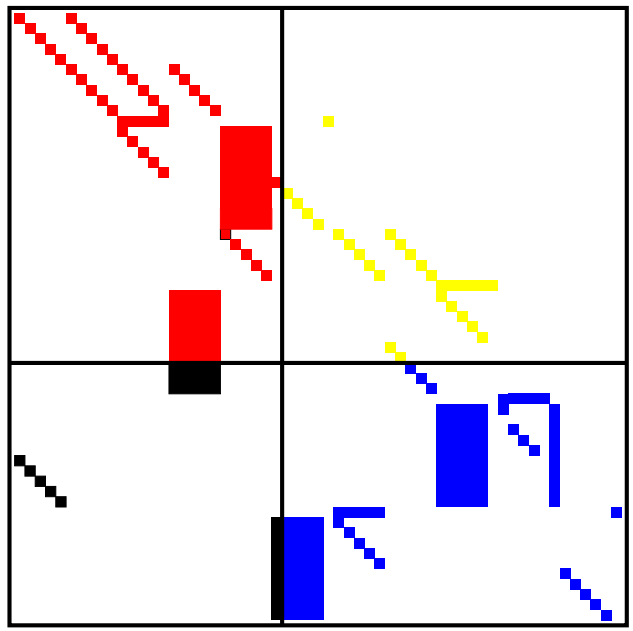


- 4 supersteps: communicate, compute, communicate, compute.
- Parallel sparse matrix–vector multiplication $\mathbf{u} := A\mathbf{v}$, where A sparse $n \times n$ matrix, $\mathbf{u}, \mathbf{v}$ dense vectors of length n.
- Computation of

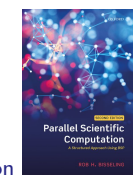
$$u_i := \sum_{j=0}^{n-1} a_{ij} v_j, \quad \text{for } 0 \leq i < n.$$



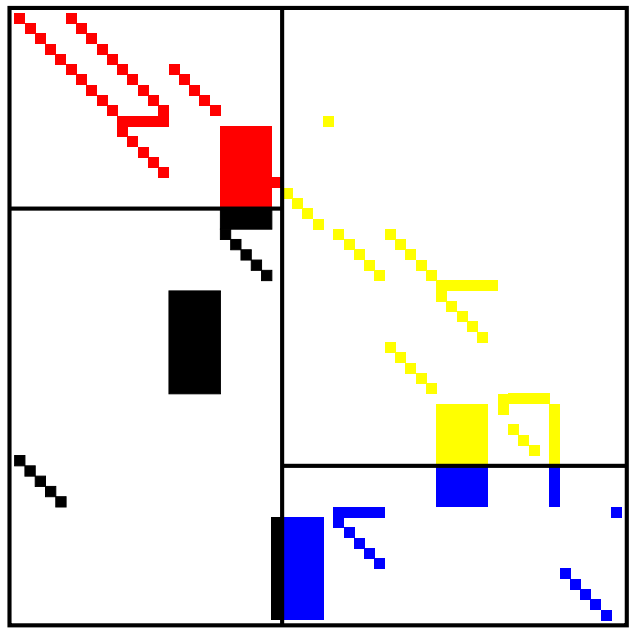
Cartesian matrix partitioning



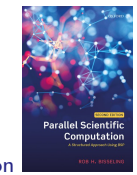
- Block distribution of 59×59 matrix `impcol_b` from Harwell–Boeing collection with 312 nonzeros, for $p = 4$
- #nonzeros per processor: 126, 28, 128, 30
- Each split has optimal balance (for blocks)



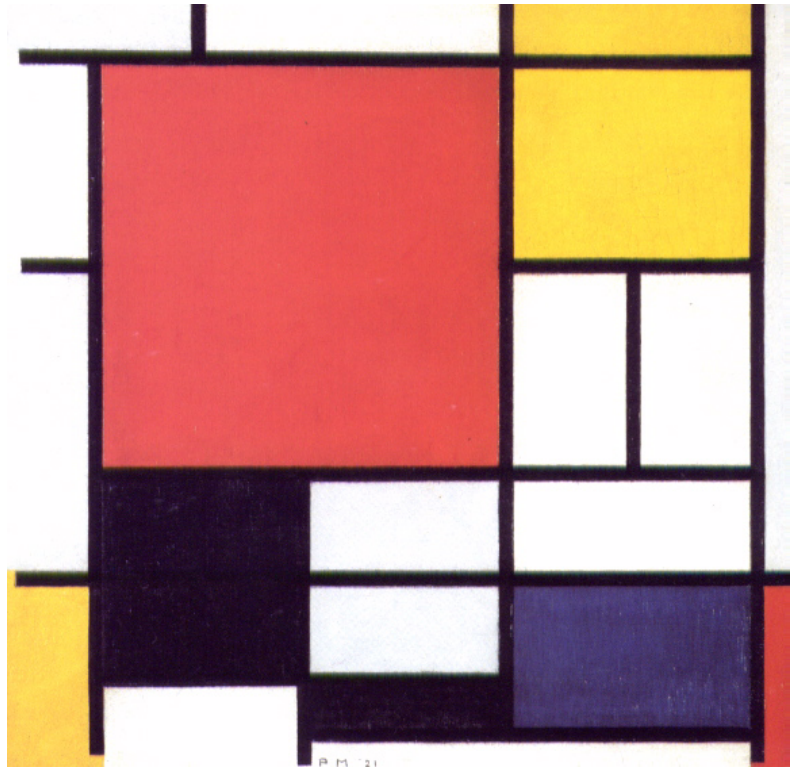
Non-Cartesian matrix partitioning



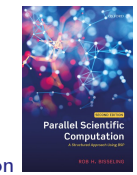
- Block distribution of 59×59 matrix `impcol_b` from Harwell–Boeing collection with 312 nonzeros, for $p = 4$
- #nonzeros per processor: 76, 76, 80, 80
- Each split has optimal balance (for blocks)



Composition with Red, Yellow, Blue and Black



Piet Mondriaan 1921



p-way matrix partitioning

- A p-way partitioning $A_0, \ldots, A_{p-1}$ is defined by

$$\bigcup_{s=0}^{p-1} A_s = A,$$

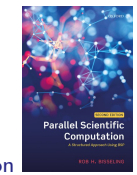
$$A_s \cap A_t = \emptyset, \quad \text{for } s \neq t,$$

$$A_s \neq \emptyset, \quad \text{for } 0 \leq s < p.$$

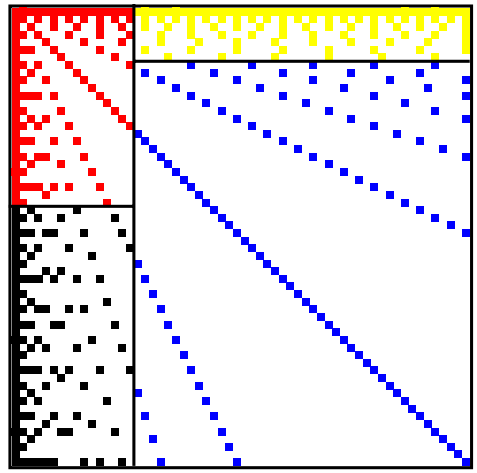
- Here, A_s is the set of index pairs of the nonzeros of processor $P(s)$,

$$A_s = \{(i,j) : 0 \leq i,j < n \wedge \phi(i,j) = s\}, \quad \text{for } 0 \leq s < p.$$

- For the purpose of partitioning, we identify: nonzero $\equiv$ index pair; sparse matrix $\equiv$ set of index pairs.
- If all $nz(A_s) > 0$, then $A_0, \ldots, A_{p-1}$ forms a p-way partitioning of $A = \{(i,j) : 0 \leq i,j < n \wedge a_{ij} \neq 0\}$.
- We use the notation $V(A_0, \ldots, A_{p-1}) = V_\phi$.

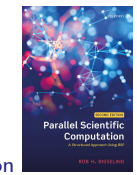


Communication volume for partitioned matrix



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

- $V(A_0, A_1, A_2, A_3)$ 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 $A_2 \cup A_3$.

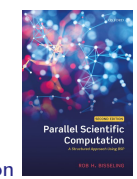


Motivation of the Mondriaan splitting

Theorem 4.6 Given a 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: the split that creates $k+1$ parts from k parts can be done locally and independently, looking at just the part that is to be split.
- This greedily minimizes the total communication volume.



Proof of the theorem

- For a given partitioning $A_0, \ldots, A_{k-1}$, let the number of processors that need a vector component v_j be $\mu_j = \mu_j(A_0, \ldots, A_{k-1})$; this is 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 $\lambda_i = \lambda_i(A_0, \ldots, A_{k-1})$.
- Let $\lambda'_i = \max(\lambda_i - 1, 0)$ and $\mu'_j = \max(\mu_j - 1, 0)$, so that the total communication volume is

$$V = \sum_{i=0}^{n-1} \lambda'_i + \sum_{j=0}^{n-1} \mu'_j.$$

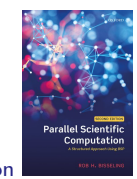
- Instead of proving

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

it is sufficient to prove for all i that

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

and similar for the μ'_j.



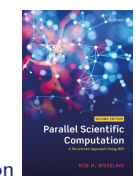
Proof of the theorem (cont'd)

- $\lambda_i = \#$ 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 $\lambda'_i = \max(\lambda_i - 1, 0) = \lambda_i - 1$ in all three terms. Thus,

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

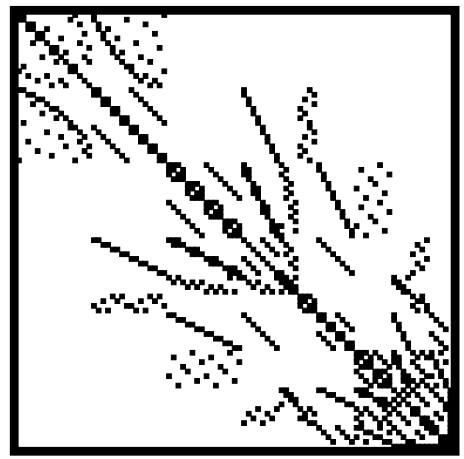
- If row i has no nonzero in $A_{k-1} \cup A_k$, then both A_{k-1} and A_k are empty, so that

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



Computational load balance

- Paint all nonzeros black:

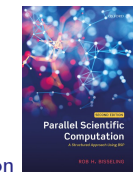


No communication, but no parallelism. No pain, no gain!

- A load balance criterion must therefore be satisfied:

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

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



BSP cost determines ϵ

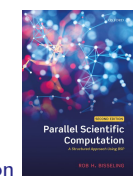
- Best choice of ϵ is machine-dependent and can be found by using the BSP model.
- Communication cost is $\frac{Vg}{p}$, assuming that the communication is balanced by the vector partitioning that follows the matrix partitioning.
- Total BSP cost is

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

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

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

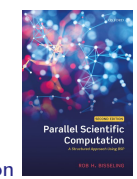
- If necessary, we adjust ϵ and run the partitioner again.



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 saves 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 rows (or by columns) restricts the search space to $2^{n-1} = 2^4 = 16$ possibilities.



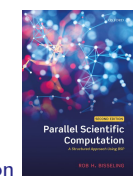
Vertical and horizontal bipartitioning

- Assume $\epsilon = 0.1$, so $nz(A_s) \leq 7$, for $s = 0, 1$.
- An optimal column split has $V = 4$:

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

- An optimal row split has $V = 3$:

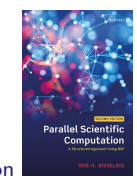
$$A = \begin{bmatrix} \cdot & 3 & \cdot & \cdot & 1 \\ 4 & 1 & \cdot & \cdot & \cdot \\ \cdot & 5 & 9 & 2 & \cdot \\ 6 & \cdot & \cdot & 5 & 3 \\ \cdot & \cdot & 5 & 8 & 9 \end{bmatrix}.$$



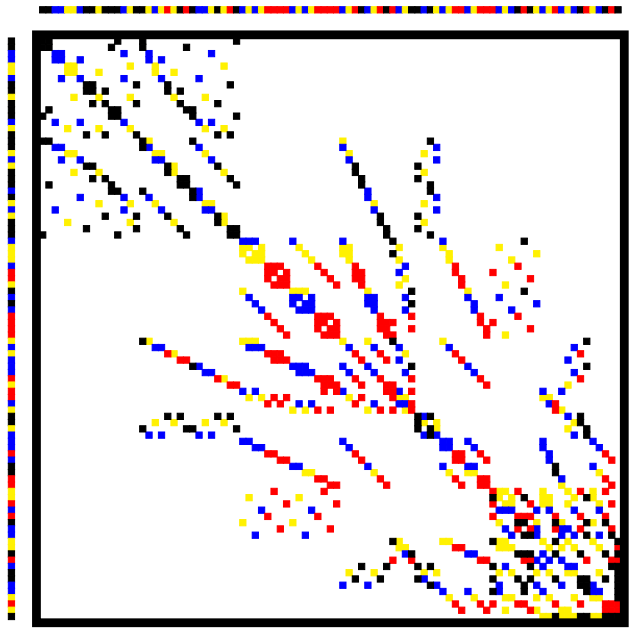
Repeated splits

- 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 submatrices are mutually disjoint.
- Removing empty rows and columns from $\bar{I}_s \times \bar{J}_s$ gives $I_s \times J_s$. Thus

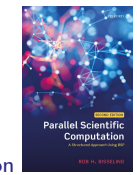
$$A_s \subseteq I_s \times J_s \subseteq \bar{I}_s \times \bar{J}_s.$$



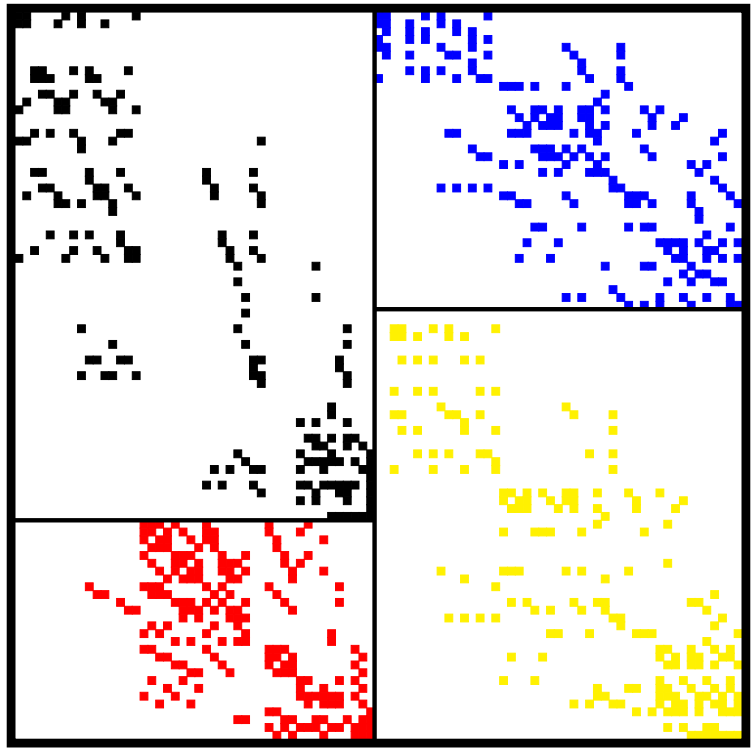
Global view of matrix cage6



- Distribution of 93×93 matrix cage6 with 785 nonzeros, for $p = 4$, obtained by Mondriaan partitioning with $\epsilon = 3\%$.
- Maximum number of nonzeros per processor is 202; average is 785/4=196.25. Achieved imbalance is $\epsilon' \approx 2.93\%$.
- Communication volume: fanout 58; fanin 55; $V = 113$.



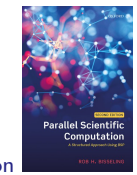
Local view of matrix cage6



- The local submatrix $\bar{I}_s \times \bar{J}_s$ of processor $P(s)$ has size:

  65×45 for $P(3)$; 38×48 for $P(1)$;
  28×45 for $P(2)$; 55×48 for $P(0)$.

- $\bar{I}_0 \times \bar{J}_0$ has 17 empty rows and 6 empty columns, giving a size of 38×42 for $I_0 \times J_0$.

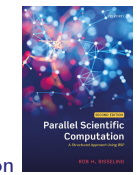


Growth of load imbalance by splitting

- 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.,

$$\delta = \frac{\epsilon}{q}.$$

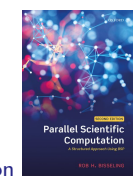
- For the deepest splits, with $q = 1$, we have $\delta = \epsilon$ and the above first-order approximation is exact.
- After the first split, one part has at least half the nonzeros, and the other part at most half. We recompute the ϵ-values for both halves based on the new situation.
- Here, the less-loaded processor can increase the allowed load imbalance to reduce communication in further splits.



Recursive matrix partitioning

input: A: sparse $m \times n$ matrix, $p = 2^q$, $\epsilon =$ allowed imbalance.
output: $(A_0, \ldots, A_{p-1})$: ϵ-balanced p-way partitioning of A.

function MatrixPartition(A, p, ϵ)
  if $p > 1$ **then**
    $maxnz := (1 + \epsilon)\frac{nz(A)}{p}$;
    $(B_0^{\text{row}}, B_1^{\text{row}}) := Bipartition(A, \text{row}, \frac{\epsilon}{q})$;
    $(B_0^{\text{col}}, B_1^{\text{col}}) := Bipartition(A, \text{col}, \frac{\epsilon}{q})$;
    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 $(B_0, B_1) := (B_0^{\text{col}}, B_1^{\text{col}})$;

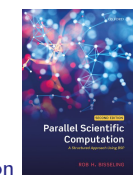


Recursive matrix partitioning

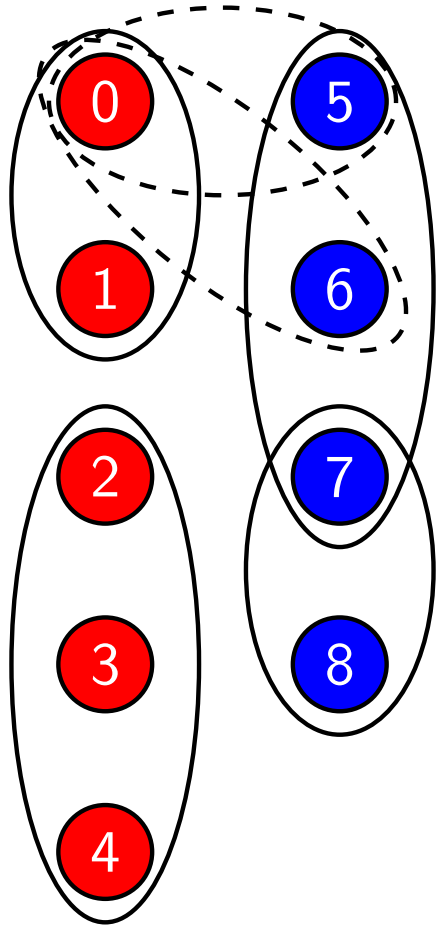
input: A: sparse $m \times n$ matrix, $p = 2^q$, $\epsilon =$ allowed imbalance.
output: $(A_0, \ldots, A_{p-1})$: ϵ-balanced p-way partitioning of A.

function MatrixPartition(A, p, ϵ)
  if $p > 1$ **then**
    $maxnz := (1+\epsilon)\frac{nz(A)}{p}$;
    $(B_0^{\text{row}}, B_1^{\text{row}}) := Bipartition(A, \text{row}, \frac{\epsilon}{q})$;
    $(B_0^{\text{col}}, B_1^{\text{col}}) := Bipartition(A, \text{col}, \frac{\epsilon}{q})$;
    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 $(B_0, B_1) := (B_0^{\text{col}}, B_1^{\text{col}})$;

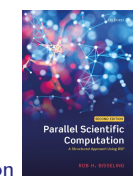
    $\epsilon_0 := \frac{maxnz}{nz(B_0)} \cdot \frac{p}{2} - 1$; $\epsilon_1 := \frac{maxnz}{nz(B_1)} \cdot \frac{p}{2} - 1$;
    $(A_0, \ldots, A_{p/2-1}) :=$ MatrixPartition$(B_0, \frac{p}{2}, \epsilon_0)$;
    $(A_{p/2}, \ldots, A_{p-1}) :=$ MatrixPartition$(B_1, \frac{p}{2}, \epsilon_1)$;
  else
    $A_0 := A$;



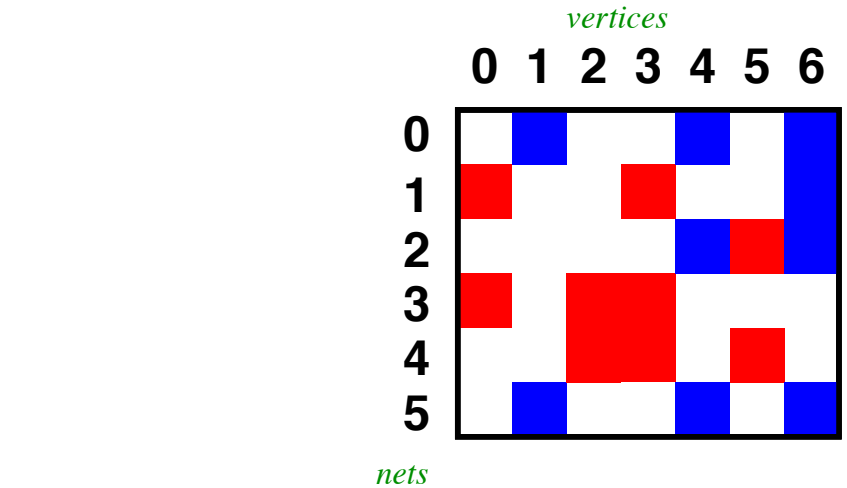
Hypergraph



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



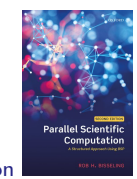
The row-net model



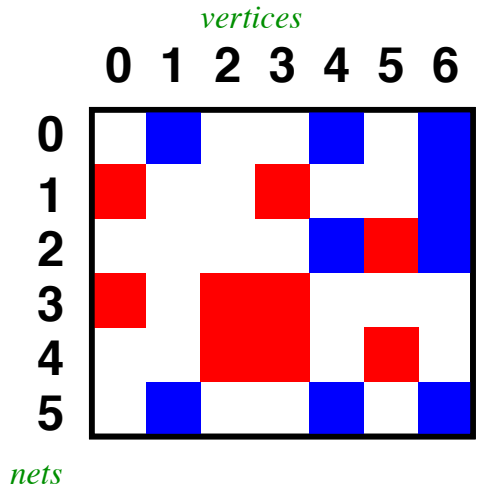
- Hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ gives exact communication volume.
- Columns $\equiv$ Vertices: $0, 1, 2, 3, 4, 5, 6$.
  Rows $\equiv$ Hyperedges (nets), $n_i = \{j : 0 \leq j < n \wedge a_{ij} \neq 0\}$:

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

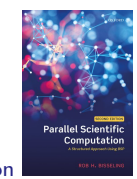
Ü. V. Çatalyürek and C. Aykanat, *IEEE Transactions on Parallel and Distributed Systems* **10** (7) (1999) pp. 673–693.



Minimizing communication volume

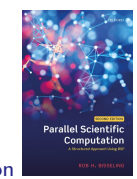


- Cut nets n_1, n_2 each cause one communication: $V = 2$.
- Use Kernighan–Lin algorithm for hypergraph bipartitioning: start with an initial random partitioning.
- Try to improve it by moving vertices (columns) with the largest gain in communication to the other part.
- If this increases the communication, the move is still accepted.
- Several passes are carried out. Vertices are never moved twice in a pass. The best solution encountered is kept.

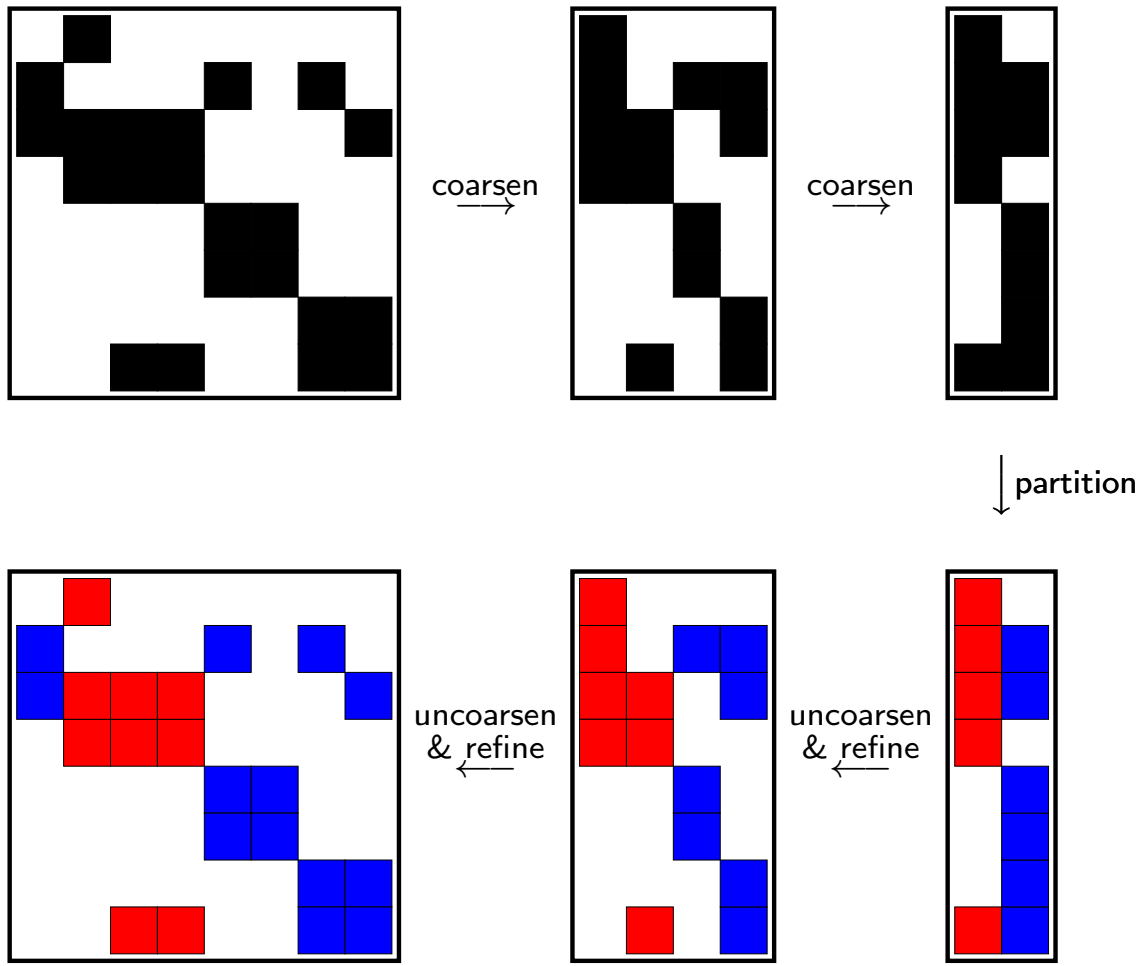


The multilevel scheme

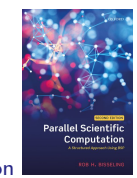
1. Merge similar columns in pairs to reduce the problem size, and repeat this until the problem is small.
2. Bipartition the smaller problem using Kernighan–Lin with an improved implementation by Fiduccia and Mattheyses.
3. Refine the bipartitioning using a simplified KLFM scheme.



Multilevel bipartitioning of an 8×8 matrix



- Here, the coarsening of the matrix is by merging pairs of adjacent columns.



Summary

- We have derived a recursive algorithm that yields a p-way sparse matrix partitioning $A_0, \ldots, A_{p-1}$ with

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

- It is greedy, i.e., minimizes for the splits separately without looking ahead, and it adapts the allowed load imbalance to the current partitioning.
- A hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ is a generalization of a graph. It consists of a set $\mathcal{V}$ of vertices and a set $\mathcal{N}$ of hyperedges, which are subsets of $\mathcal{V}$.
- Multilevel methods for hypergraph partitioning find good splits of a sparse matrix in reasonable time.