Data Distribution

Albert-Jan Yzelman

7th of November, 2014

Derived from the book slides by Prof. dr. Rob H. Bisseling, found at

`www.math.uu.nl/people/bisseling/Education/PA/pa.html`

Sparse matrix–vector multiplication

Parallel sparse matrix–vector multiplication $\mathbf{u} := A\mathbf{v}$, with

- A sparse $m \times n$ matrix,
- $\mathbf{u}$ dense m-vector,
- $\mathbf{v}$ dense n-vector;

computes

$$u_i := \sum_{j=0}^{n-1} a_{ij} v_j$$

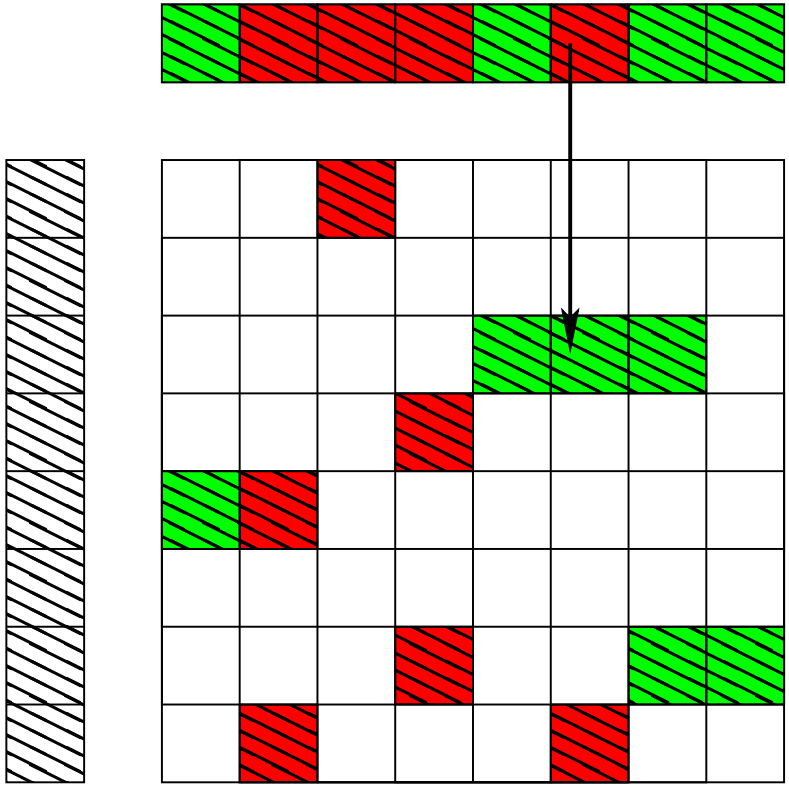
Four supersteps:

1. communicate (fan-out),
2. compute (local SpMV),
3. communicate (fan-in),
4. compute (handle remote contributions).

Sparse matrix–vector multiplication

Four supersteps:

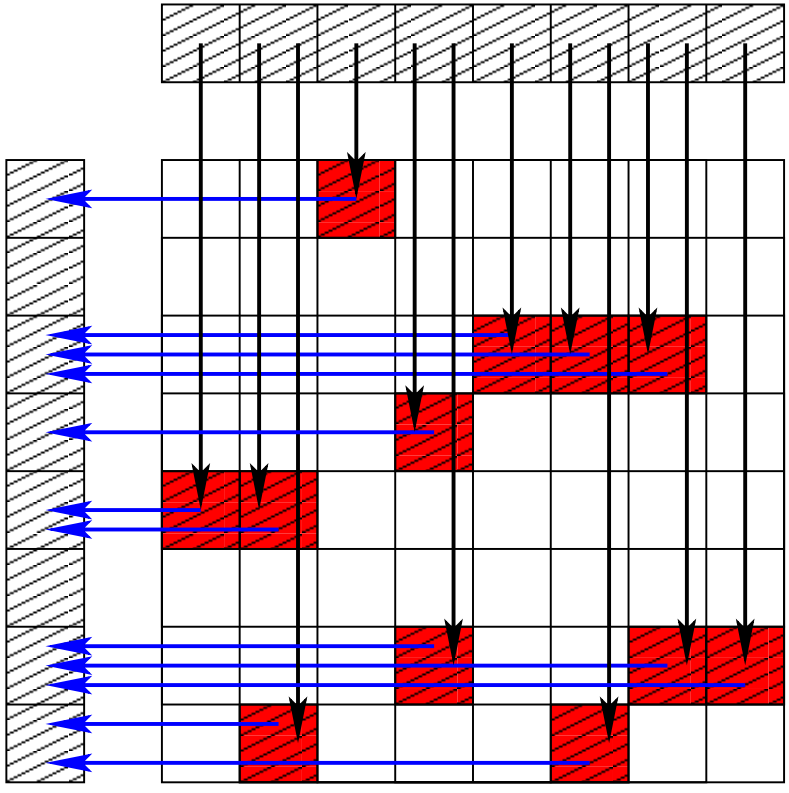
1. fan-out,
2. local SpMV,
3. fan-in,
4. handle remote contributions.



Sparse matrix–vector multiplication

Four supersteps:

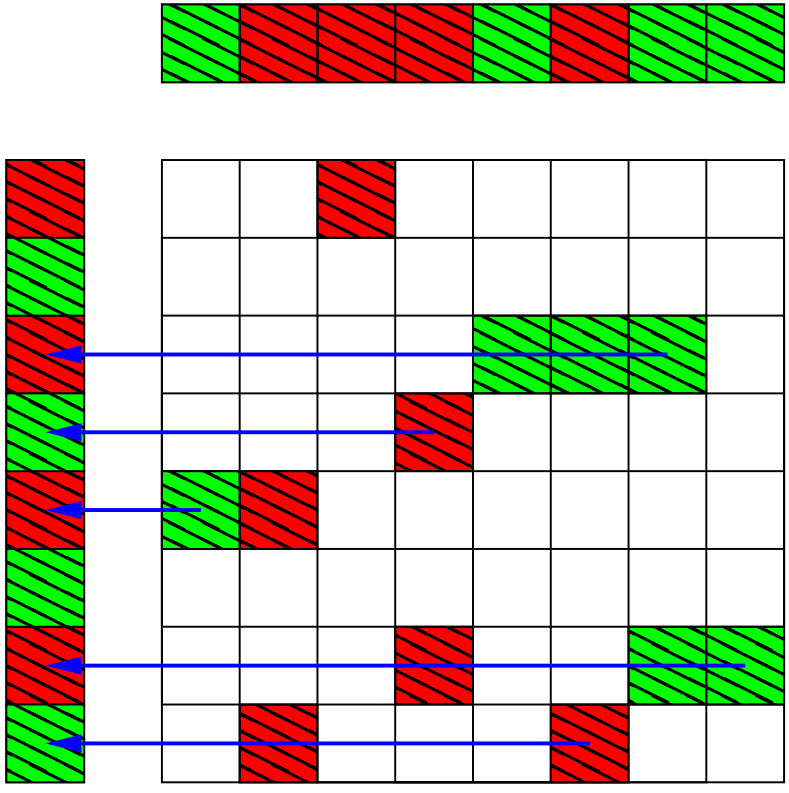
1. fan-out,
2. local SpMV,
3. fan-in,
4. handle remote contributions.



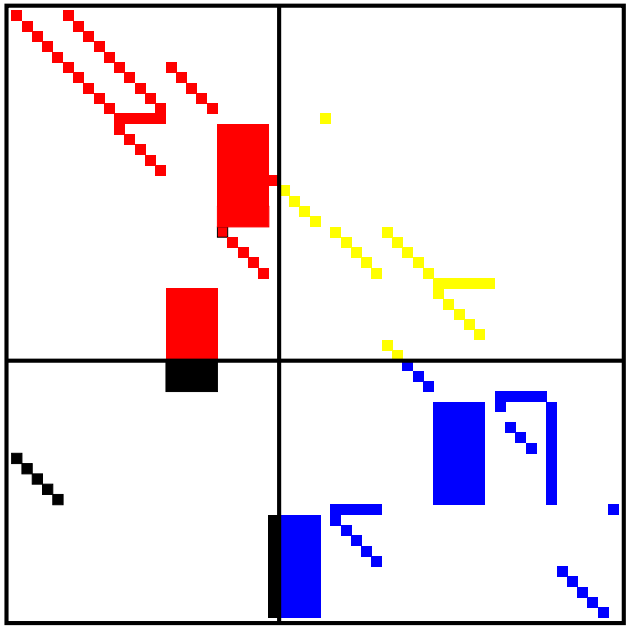
Sparse matrix–vector multiplication

Four supersteps:

1. fan-out,
2. local SpMV,
3. fan-in,
4. handle remote contributions.

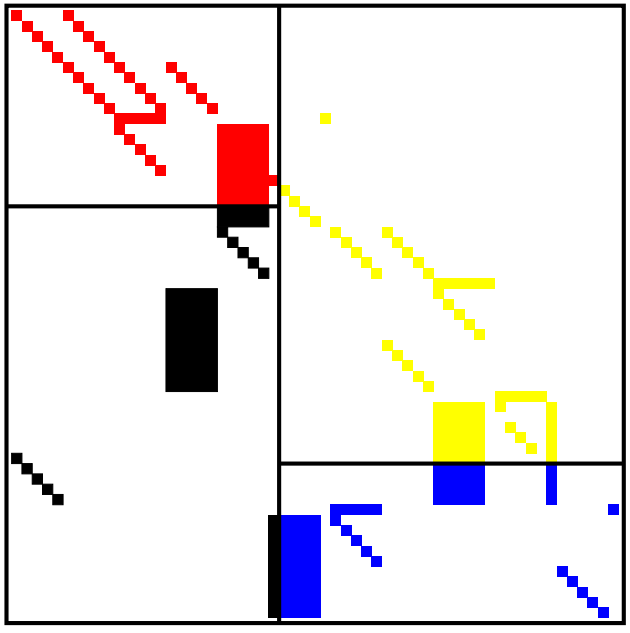


Cartesian matrix partitioning



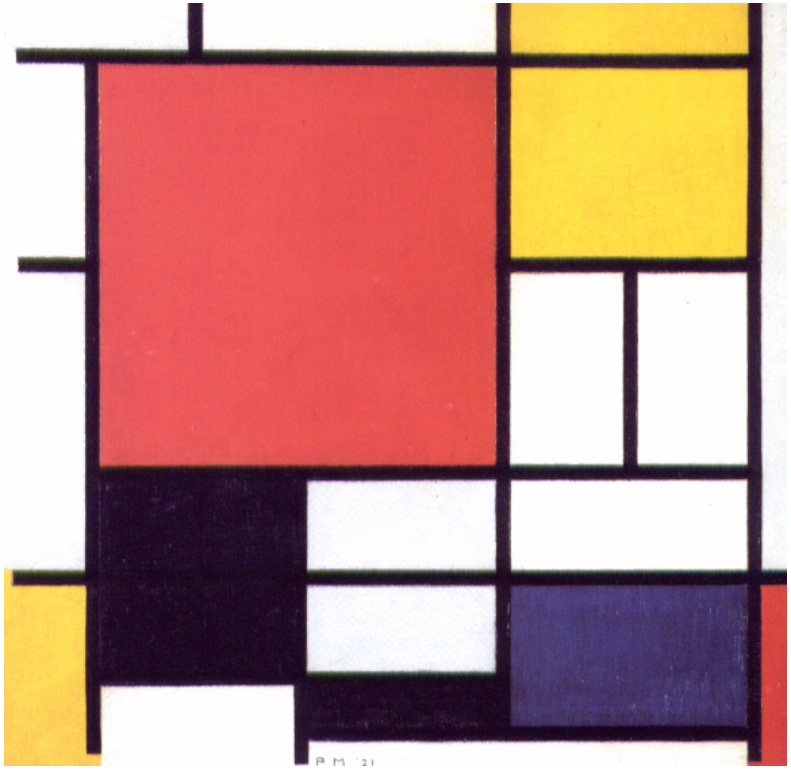
- Block distribution of 59×59 matrix `impcol_b` from Harwell–Boeing collection with 312 nonzeros, for $p = 4$
- #nonzeros per processor: 126, 28, 128, 30
- Each separate split has optimal balance (for blocks)

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

Sparse matrix partitioning

Matrix distributions

Definition (Matrix distribution)

Let A an $m \times n$ sparse matrix, $I = \{0, 1, \ldots, m-1\}$, and $J = \{0, 1, \ldots, n-1\}$. A distribution of this matrix over p processes is a function

$$\phi : I \times J \to \{0, 1, \ldots, p-1\}.$$

Definition (Matrix distribution over a process grid)

Let A, I, J as before. Let M, N be integers. If the p processes are organised in an $M \times N$ grid such that $p = M \cdot N$, then a matrix distribution over this grid is a function

$$\phi : I \times J \to \{0, 1, \ldots, M-1\} \times \{0, 1, \ldots, N-1\}.$$

Matrix distributions

You have seen matrix distributions before (dense LU decomposition):

Definition (2D cyclic distribution over a process grid)

Let $p = MN, A, I, J$ as before. A **2D cyclic distribution** is given by

$$\phi(i,j) = (i \bmod M, j \bmod N).$$

Any distribution on a process grid can be reduced to a distribution unrelated to a process grid, e.g., by mapping (s_i, s_j) to $s = s_i N + s_j M$.

Definition (2D cyclic distribution)

Let p, A, I, J as before. Assume additionally that $p = M \cdot N$ for integer M, N. Then, a 2D cyclic distribution is given by

$$\phi(i,j) = (i \bmod M) \cdot N + j \bmod N.$$

Matrix distributions

Definition (Cartesian distribution over a process grid)

Let $p = MN, A, I, J$ as before. Let $\phi_i : I \to \{0, 1, \ldots, M-1\}$ and $\phi_j : J \to \{0, 1, \ldots, N-1\}$. Then, a 2D Cartesian distribution over an $M \times N$ process grid has the following form:

$$\phi(i, j) = (\phi_i(i), \phi_j(i)).$$

Again, there is no difference with the following definition:

Definition (Cartesian distribution)

Let $p = MN, A, I, J, \phi_i, \phi_j$ as before. A 2D Cartesian distribution has the form

$$\phi(i, j) = \phi_i(i)N + \phi_j(j).$$

p-way sparse matrix partitioning

For sparse matrix partitioning, we identify:

- **nonzero** $\equiv$ index pair;
- **sparse matrix** $\equiv$ set of index pairs.

Instead of thinking about functions, we can also think about sets:

- Define the **process-local sparse matrix** A_s

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

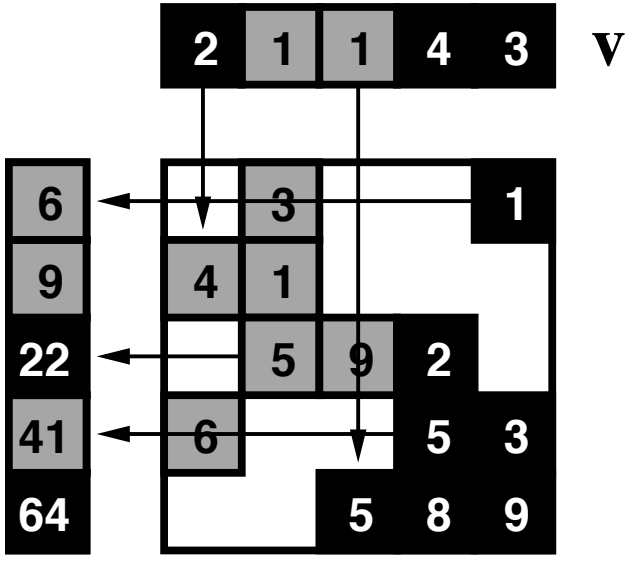
  as the set of nonzeroes local to process s, $0 \leq s < p$.
- The sets $A_0, \ldots, A_{p-1}$ form a p-**way partitioning** of

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

  if all parts are mutually disjoint and include all nonzeroes:
  - $\cup_{k=0}^{p-1} A_k = A$, and
  - $\forall 0 \leq q, r < p$ we have that $A_q \cap A_r = \emptyset$.

Aims of sparse matrix partitioning

Parallel sparse matrix–vector multiplication $\mathbf{u} := A\mathbf{v}$

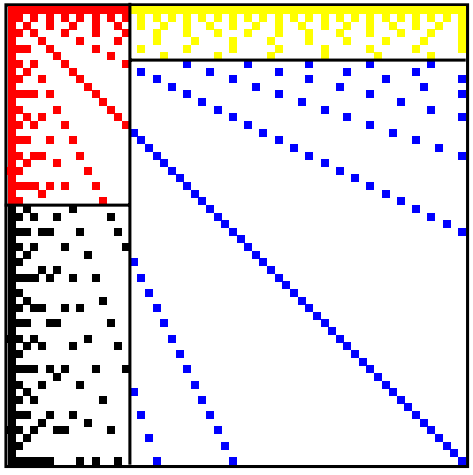


4 supersteps: communicate, compute, communicate, compute. Aims:

1. balance main computation step, and
2. minimise communication volume.

Here, the total communication volume V equals 5.

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

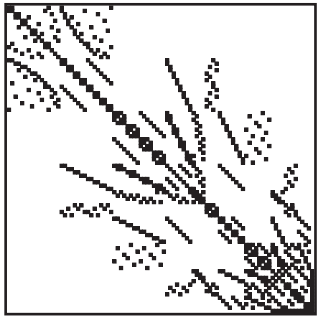
Theorem. Given an $m \times n$ sparse matrix A, and mutually disjoint subsets $A_0, \ldots, A_k$ of A, where $k \geq 1$, it holds that

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

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.

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

BSP cost determines ϵ

We now have a parameter ϵ. What is its best value?

- Communication cost is $\frac{Vg}{p}$, **assuming balanced communication**.
- 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 $\{0, 1, 2\}$ — $\{3, 4\}$, with $V = 4$.

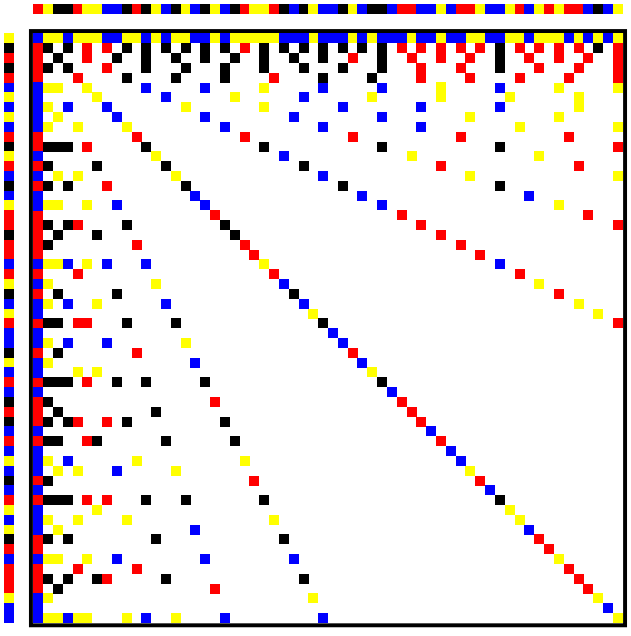
Repeated splits

Recursive bipartitioning: starts with a complete matrix, splits it into 2 submatrices, and recurses on each submatrix (until p parts are created). The maximum number of nonzeroes in one part is at most $(1+\epsilon)\frac{nz}{2}$. The 1:1 load balance ratio might shift if $p \neq 2^q$!

- Rows and columns in the submatrix need not be consecutive.
- A split in the column in direction can cause empty rows to appear in the submatrix (and vice versa).
- The final result for processor $P(s)$ is a local matrix A_s. This matrix is a submatrix of A that corresponds to the rows and columns of $\bar{I}_s \times \bar{J}_s$.
- Removing empty rows and columns from $\bar{I}_s \times \bar{J}_s$ gives $I_s \times J_s$. Thus

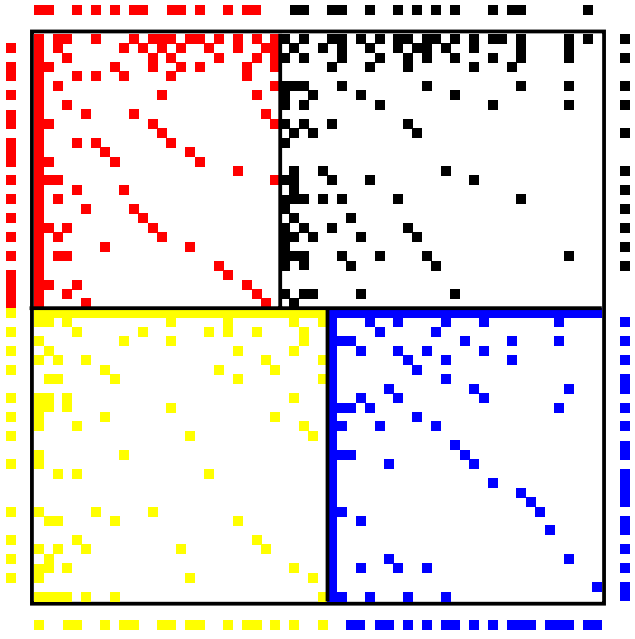
$$A_s \subset I_s \times J_s \subset \bar{I}_s \times \bar{J}_s.$$

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 part can increase the allowed load imbalance as its farther from its maximum load. This results in more freedom for the partitioner 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**

$\ldots$

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

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

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

else

$A_0 := A$;

The magic *split* function

This clarifies the limitations of what the split can do;

- either rowwise or columnwise splits, and
- cannot return a bipartitioning that deviates more than ϵ from
- an ideal 1:1 split in terms of load-balance.

But how does it work, and how does it minimise communication?

Graphs and hypergraphs

To solve this multi-constraint optimisation problem, we use **hypergraphs**. We first introduce graphs:

Definition (Graph)

Let V be a set of **vertices** and $E = \{ \{v_i, v_j\} \mid v_{i,j} \in V \}$ a set of **edges**. Then $G = (V, E)$ is an **undirected graph**.

Graphs and hypergraphs

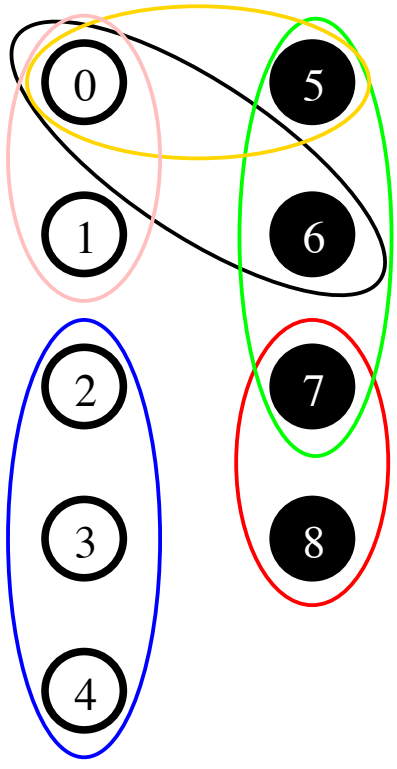
Each edge $e \in E$ of a graph connects but two vertices. Hypergraphs allow for **larger connectivities**.

Definition (Hypergraph)

Let $\mathcal{V}$ be a set of vertices and $\mathcal{N} \subseteq \mathcal{P}(\mathcal{V})$ a set of **hyperedges** (also called **nets**). Then $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ is a hypergraph.

Note that $\mathcal{N} \ni n \subseteq \mathcal{V}$, so $|n| > 2$ is possible.

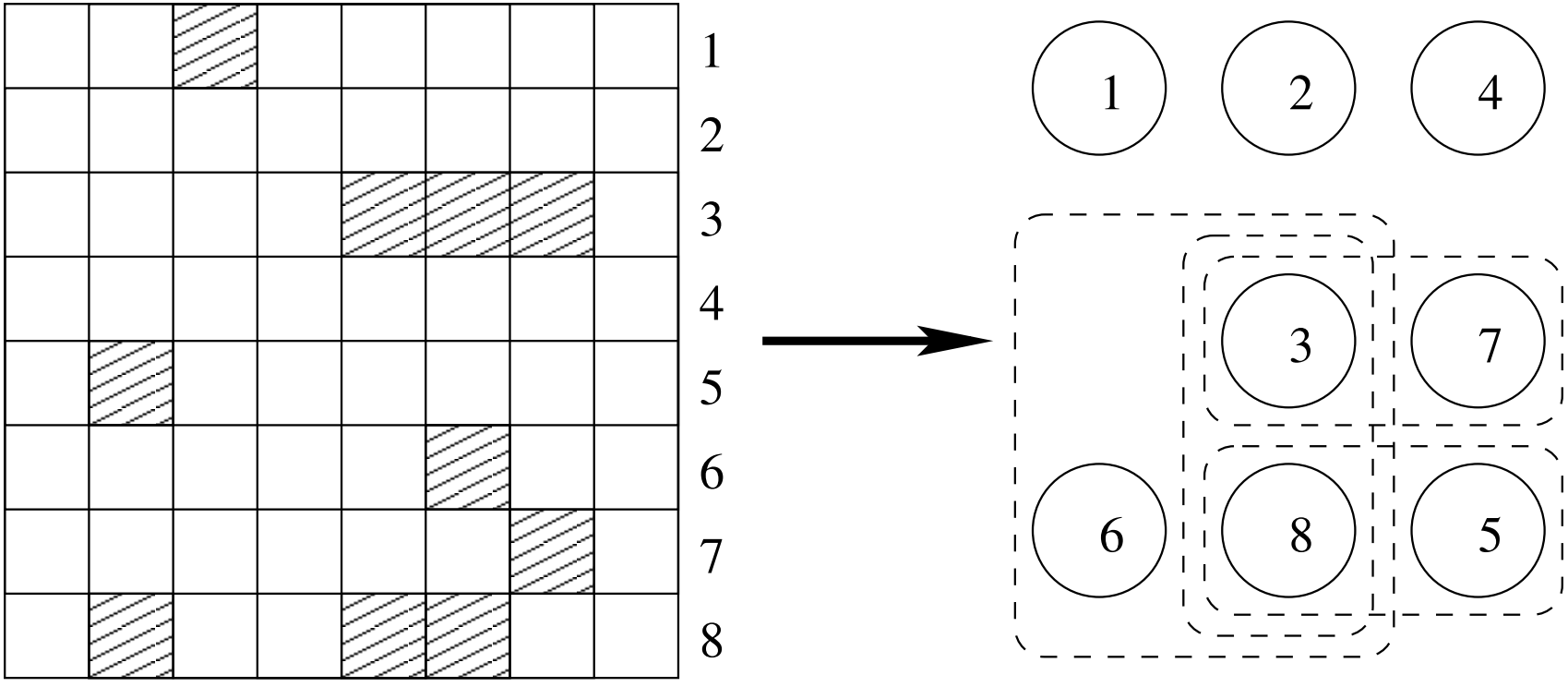
Example hypergraph



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

From matrix to hypergraph

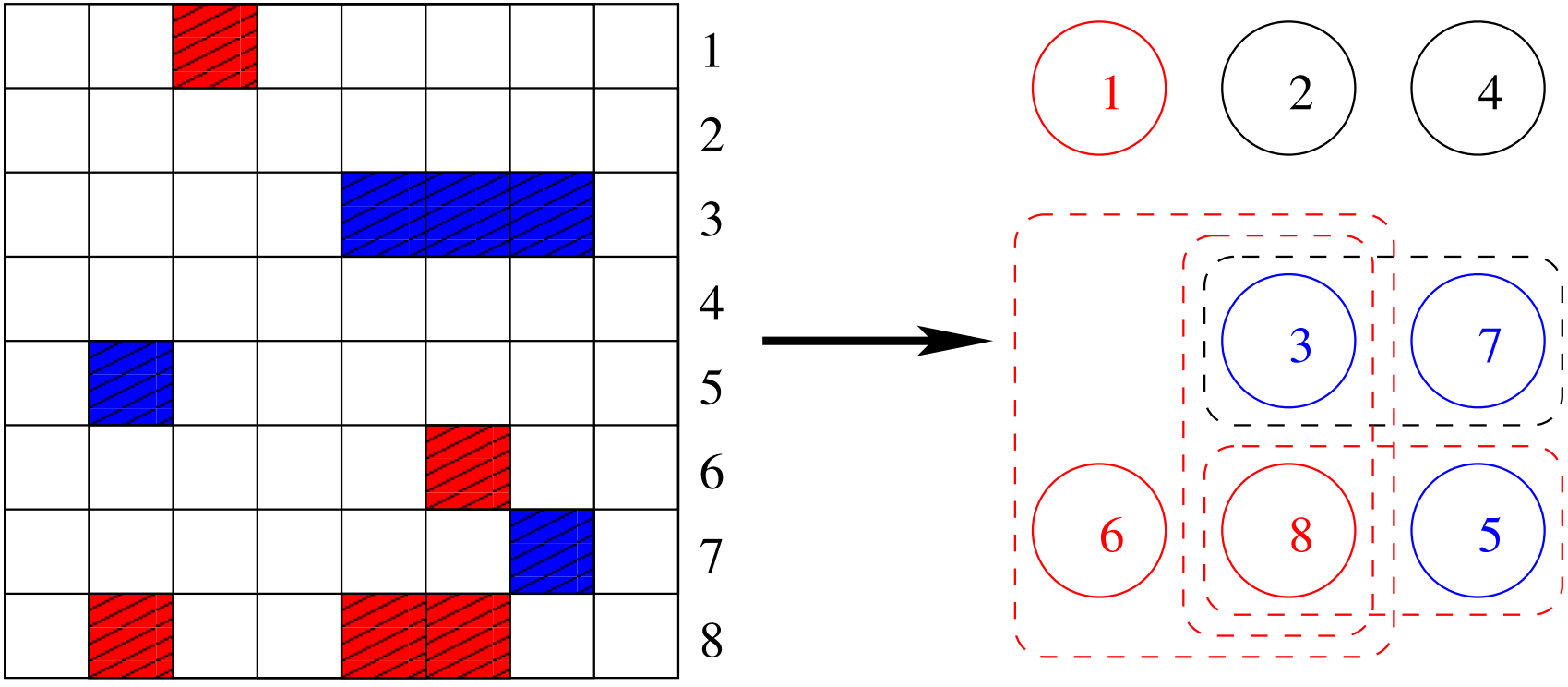
"Shared" columns: communication during fan-out



Column-net model; a cut net means a shared column

From matrix to hypergraph

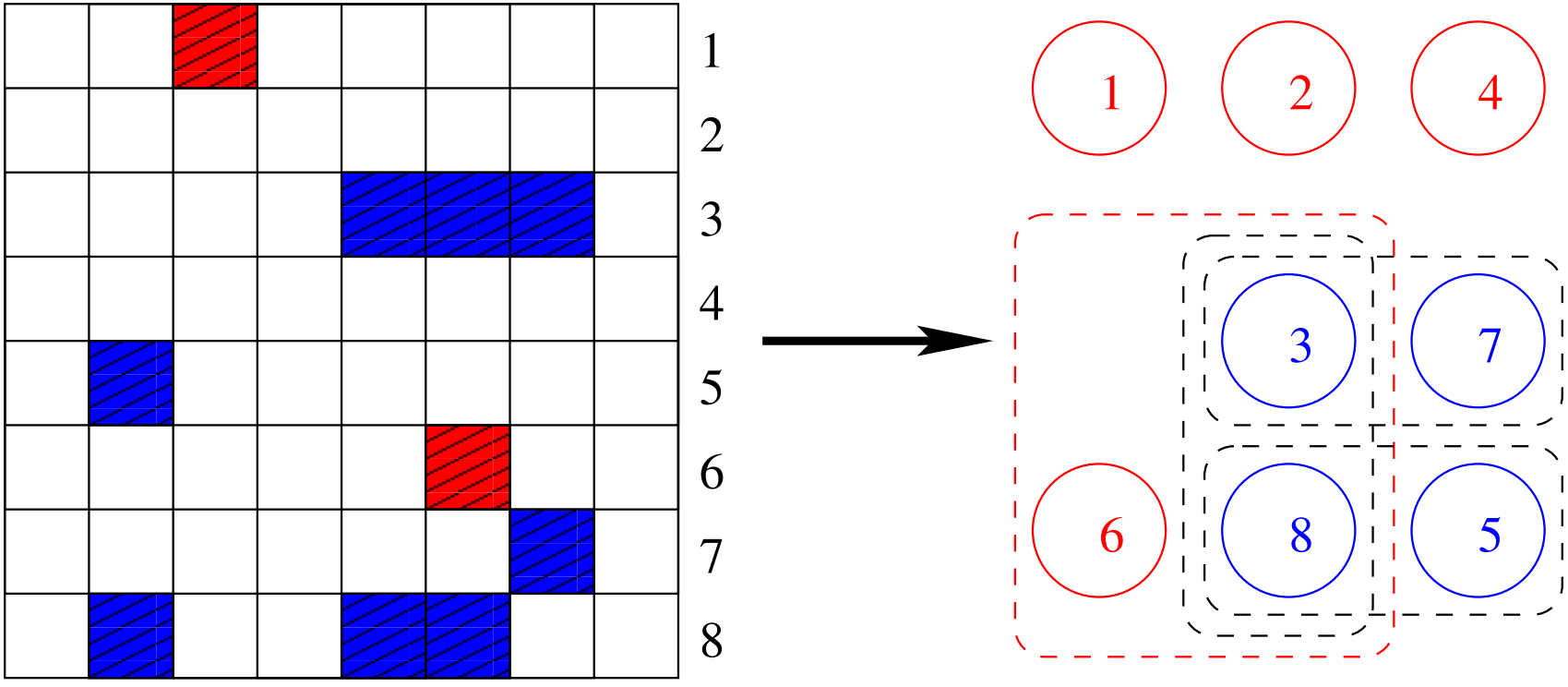
"Shared" columns: communication during fan-out



Column-net model; a cut net means a shared column

From matrix to hypergraph

"Shared" columns: communication during fan-out



Column-net model; a cut net means a shared column

From matrix to hypergraph

Definition (Column-net model of a sparse matrix)

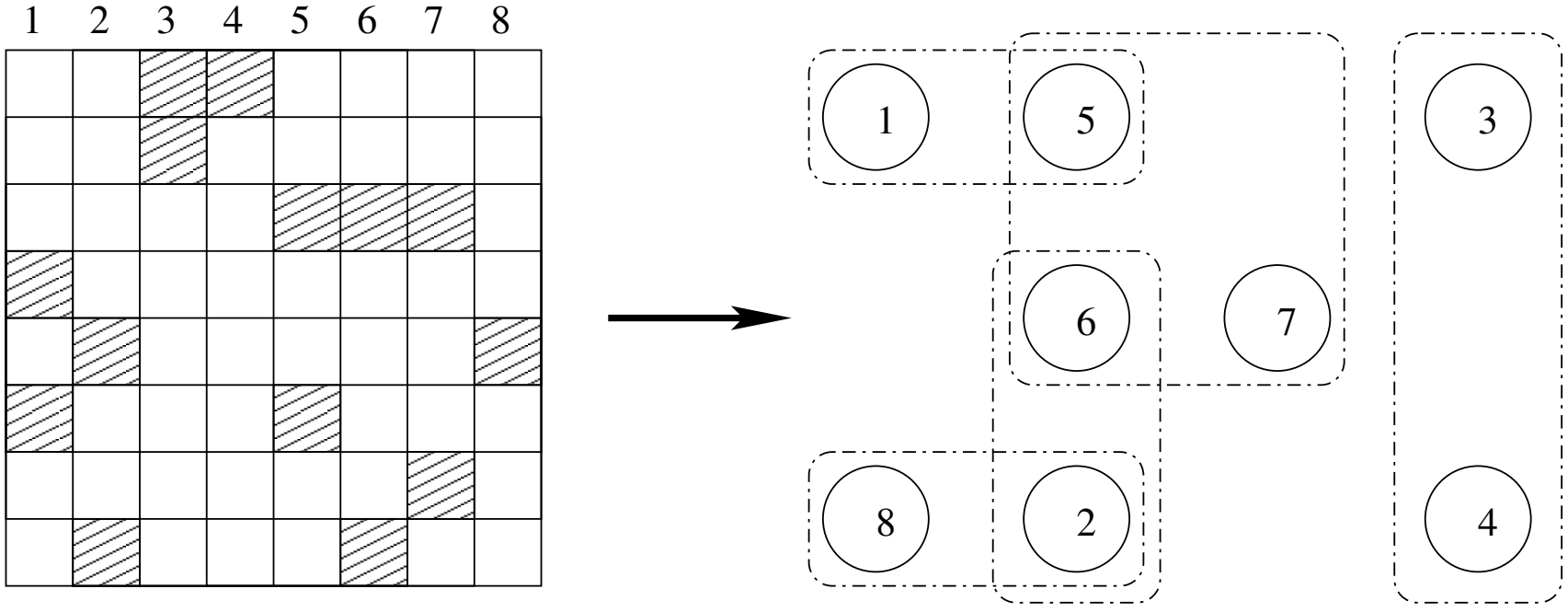
Let A be an $m \times n$ sparse matrix, $I = \{0, 1, \ldots, m-1\}$, and $J = \{0, 1, \ldots, n-1\}$. Define $\mathcal{V} = I$, and $\forall i \in I$ define a net $n_i \in \mathcal{N}$ with

$$n_i = \{j \in J \mid a_{ij} \neq 0\}.$$

Then $(\mathcal{V}, \mathcal{N})$ is the **column-net model** of A.

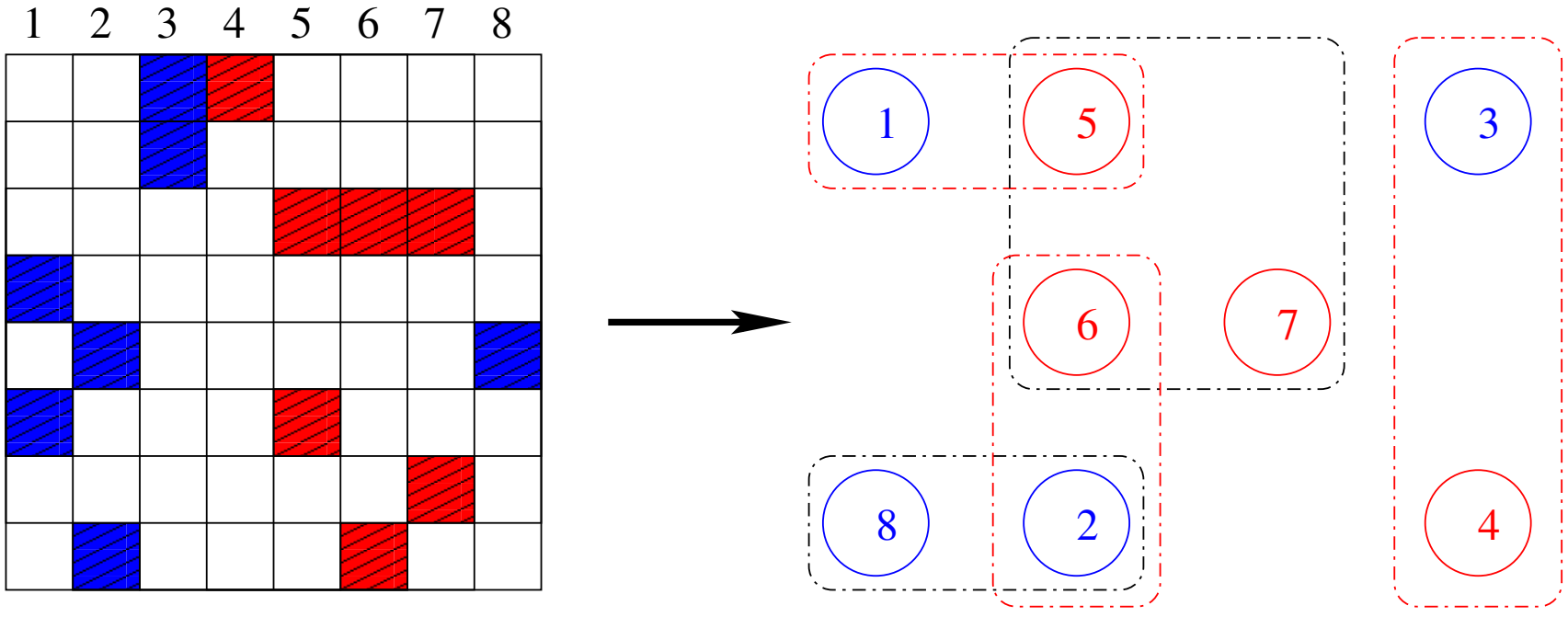
From matrix to hypergraph

"Shared" rows: communication during fan-in



From matrix to hypergraph

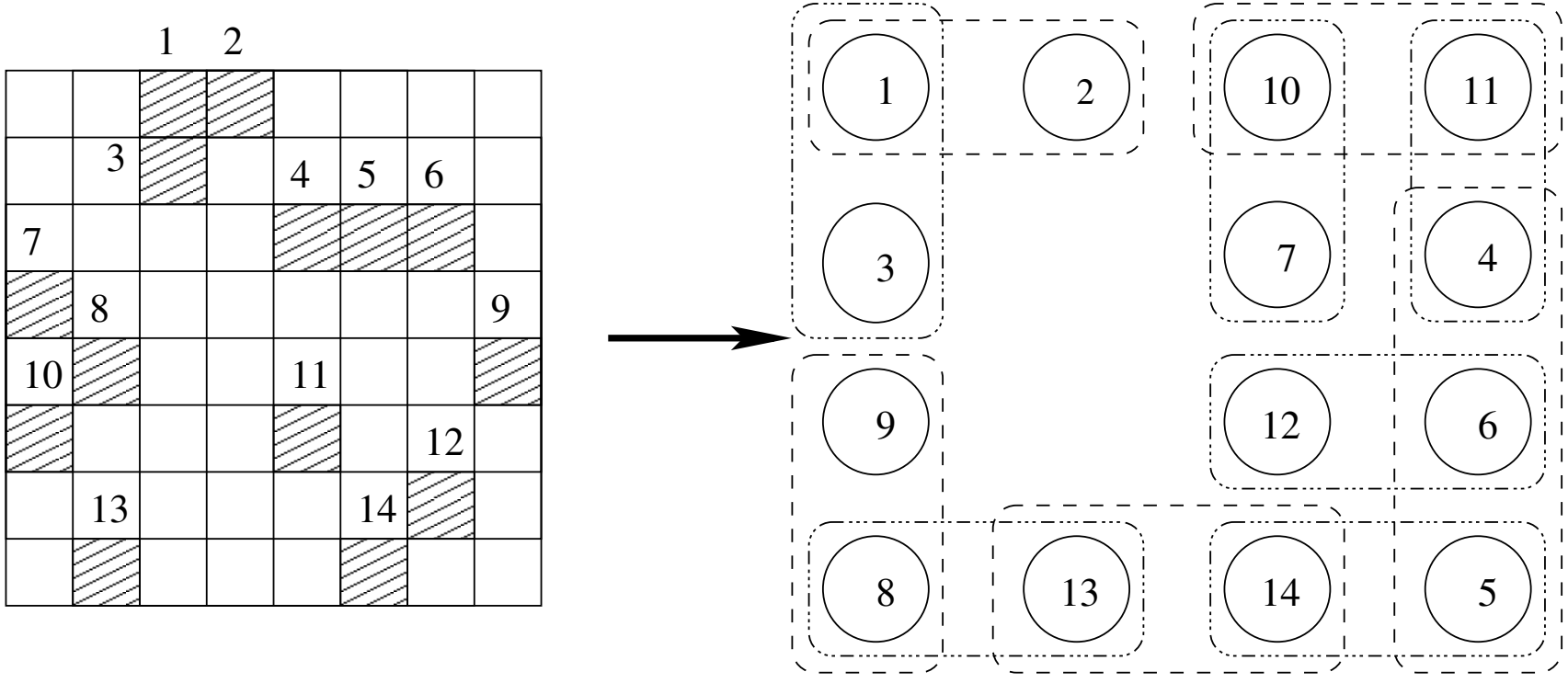
"Shared" rows: communication during fan-in



Row-net model; a cut net means a shared row. Definition is analogous to that of the column-net model, but with the roles of matrix rows and columns in reverse.

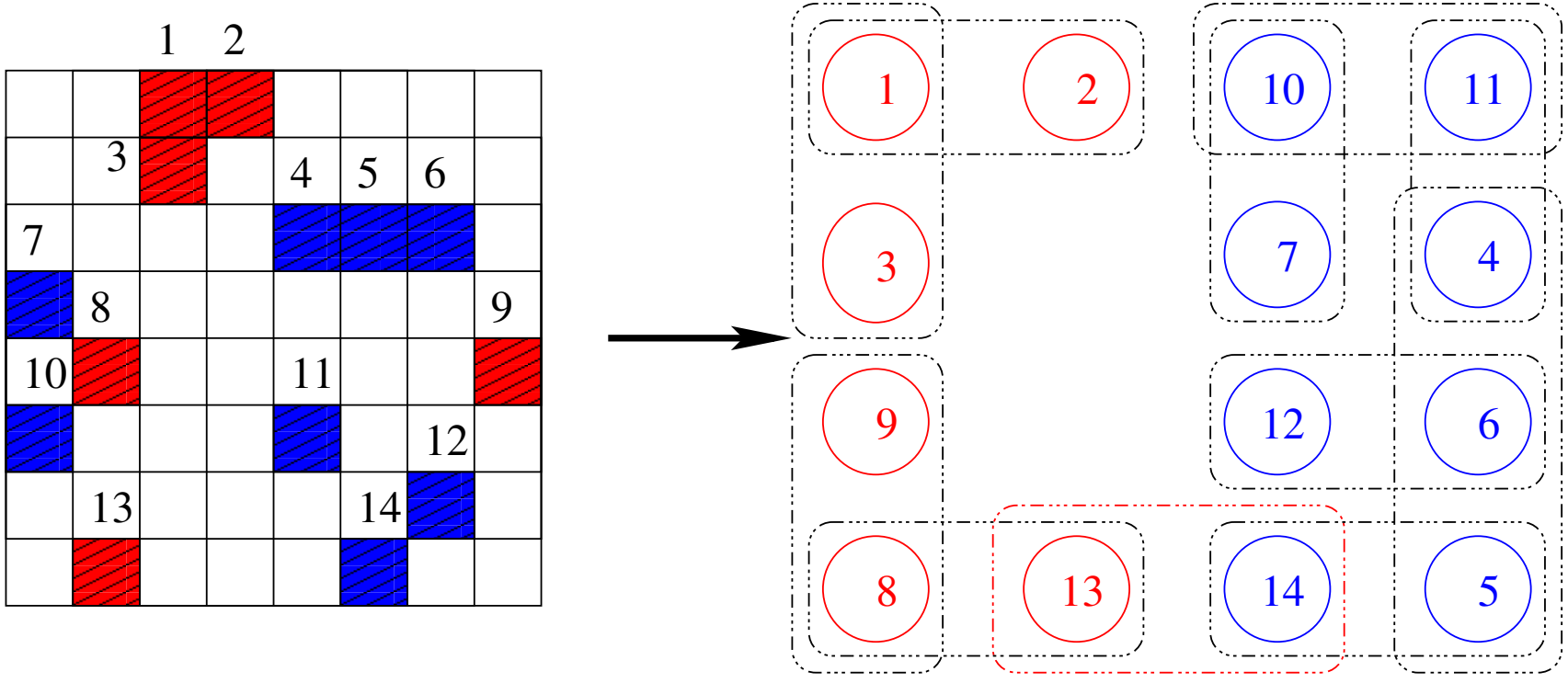
From matrix to hypergraph

Catch all communication:



From matrix to hypergraph

Catch all communication:



Fine-grain model; a cut net means **either** fan-out or fan-in.

Matrix communication costs via hypergraphs

Do hypergraph models allow precise modeling of the communication volumes?

Definition (Connectivity of a hyperedge)

Let $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ be a hypergraph. Let $\mathcal{P}_k = \{\mathcal{V}_0, \ldots, \mathcal{V}_{k-1}\}$ be a k-way partitioning of $\mathcal{H}$. Then, the connectivity λ_i of the hyperedge $n_i \in \mathcal{N}$ is given by

$$\lambda_i = |\{\mathcal{V}_i \in \mathcal{P}_k \mid n_i \cap \mathcal{V}_i \neq \emptyset\}|.$$

Matrix communication costs via hypergraphs

Given a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ and a partitioning $\mathcal{P}_k$ of $\mathcal{V}$:

- a net is **cut** precisely if its connectivity is larger than 1.

Definition (Cut-net metric)

The cost of a partitioning according to the cut-net metric is given by

$$\sum_{n_i \in \mathcal{N}} \begin{cases} 1, & \text{if } \lambda_i > 1 \\ 0, & \text{otherwise.} \end{cases}$$

Question: does this model the communication volume?

Matrix communication to hypergraph costs

Answer: **no**. But we can:

Definition ($(\lambda - 1)$-metric)

Let $\mathcal{H} = (\mathcal{V}, \mathcal{N})$, $\mathcal{P}_k$, and λ_i be as before. Then, the $(\lambda - 1)$-metric is given by

$$\sum_{n_i \in \mathcal{N}} (\lambda_i - 1) .$$

This models the communication model exactly. It counts

- the amount of fan-out communication in the column-net model,
- the amount of fan-in communication in the row-net model, and
- the total amount of communication in the fine-grain model.

Hypergraph partitioning

General data partitioning

Definition (Hypergraph partitioning)

Let $\mathcal{H} = (\mathcal{V}, \mathcal{N})$. A **partitioning** of $\mathcal{H}$ into p parts is a partitioning $\mathcal{V}_0, \mathcal{V}_1, \ldots, \mathcal{V}_{p-1}$ of $\mathcal{V}$ into p parts such that

$$(1) \quad \mathcal{V} = \cup_{s=0}^{p-1} \mathcal{V}_s, \text{ and}$$

$$(2) \quad \forall i, j \in \{0, 1, \ldots, p-1\}, \mathcal{V}_i \cap \mathcal{V}_j = \emptyset.$$

Hypergraphs models of sparse matrices, combined with hypergraph partitioning, directly results in sparse matrix partitionings.

- A partitioning of a column-net model of A corresponds to a partitioning of the rows of A (a 1D row-wise distribution).

Hypergraph partitioner

Following the example of sparse matrix partitioning:

- Model the sparse matrix using a hypergraph.
- **Partition the vertices** of that hypergraph.

State-of-the-art hypergraph partitioning is a multi-level scheme:

1. First **coarsen** the input hypergraph.
2. If the hypergraph remains too large, call this multi-level scheme recursively; otherwise, do random partitioning or optimal partitioning.
3. Undo coarsening.
4. **Refine** the resulting partitioning refinement (e.g., local search).

Partitioning: coarsening hypergraphs

Step 1: hypergraph coarsening

Definition (Coarsened hypergraph)

Let $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ be a hypergraph. Let $V_0, V_1, \ldots, V_{k-1}$ be a k-way partitioning of $\mathcal{V}$. Let $\mathcal{V}_c = \{V_0, \ldots, V_{k-1}\}$. For each $n_i \in \mathcal{N}$, there is a $n_i^c \in \mathcal{N}_c$ with

$$n_i^c = \{V_i \in \mathcal{V}_c \mid n_i \cap V_i \neq \emptyset\}.$$

Then $\mathcal{H}_c = (\mathcal{V}_c, \mathcal{N}_c)$ is a **coarsened hypergraph** of $\mathcal{H}$.

Coarsened hypergraphs should be structurally 'similar' to the original hypergraph.

Partitioning: coarsening hypergraphs

Step 1: hypergraph coarsening

Wanted: a measure for similarity.

- Assume a row-net model, where
- coarsening means combining matrix columns into 'supercolumns'.
- Hence, 'similar' columns should be combined:

Definition (structural inner product)

Let A, m, n, and I as before. Write $A_j^{\text{col}}, A_k^{\text{col}}$ for the jth and kth column of A, respectively. The **structural inner-product** $\langle j, k \rangle_{I_A}$ is

$$|\{i \in I \mid a_{ij} \neq 0 \text{ and } a_{ik} \neq 0\}|.$$

SpMV multiplication (parallel)

Step 1: hypergraph coarsening

Many similarity metrics are based on this structural inner product, and arise by using different normalisation or scaling techniques. Consider, for example, the row-net model:

1. are matching nonzeroes from two columns on highly-occupied rows just as important as those on more sparse rows?
2. are 50 out of 100 matching nonzeroes better than matching on 2 out of 2 nonzeros? Normalisation:
   1. use the minimum of $|A_j^{\text{col}}|$ and $|A_k^{\text{col}}|$;
   2. use the maximum;
   3. use the cosine $(\sqrt{\min \cdot \max})$;
   4. use the Jaccard metric $(\min + \max - \langle j, k \rangle_A)$.

Partitioning: coarsening hypergraphs

Step 1: hypergraph coarsening

A valid coarsening strategy is based on matching:

- Let $V = \{A_0^{\text{col}}, A_1^{\text{col}}, \ldots, A_{n-1}^{\text{col}}\}$.
- Let the edge $e_i(A_i^{\text{col}}, A_j^{\text{col}}) \in E$ have weight $w(i)$ equal to the similarity measure of the two columns.
- $G = (V, E, w)$ forms a fully connected edge-weighted graph.
- Let M be a weighted maximum matching of G.
- coarsen according to M

Partitioning: coarsening hypergraphs

Step 1: hypergraph coarsening

Merging similar columns in pairs to reduce the problem size (repeat this until the problem is small):

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

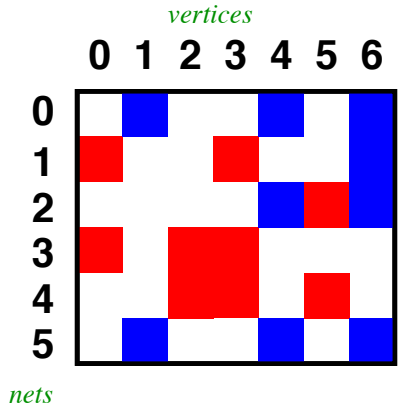
Partitioning: HKLFM

Step 4: refinement

After uncoarsening, reduce communication through local search methods.

- E.g., Kernighan-Lin, with improved implementation by Fiduccia and Mattheyses (KLFM).
- The cost function to minimise during local search is the $(\lambda - 1)$-metric.
- Moves that violate the load-balance criterion are marked invalid.

Partitioning: HKLFM



Sketch of refinement using the row-net model:

- HKLFM tries to improve initial uncoarsened 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.

Partitioning

References:

Catalyürek & Aykanat, *Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication*, IEEE Transactions on Parallel Distributed Systems 10 (1999), pp. 673-693

Kernighan & Lin, *An efficient heuristic procedure for partitioning graphs*, Bell Systems Technical Journal 49 (1970): pp. 291-307

Fiduccia & Mattheyses, *A linear-time heuristic for improving network partitions*, Proceedings of the 19th IEEE Design Automation Conference (1982), pp. 175-181.

Example software: Mondriaan (Bisseling et al., UU), Zoltan (Devine et al., Sandia), PaToH (Çatalyürek & Aykanat, OSU), and Scotch (Pellegrini, Bordelais).

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).
- The result is a p-way matrix partitioning $A_0, \ldots, A_{p-1}$.
- We used hypergraphs $\mathcal{H} = (\mathcal{V}, \mathcal{N})$, which generalise the notion of a graph.
- Multilevel methods for hypergraph partitioning find good splits of a sparse matrix in reasonable time.
- The sparse matrix partitioner introduced here optimises communication **volume**. Other possible metrics:
  - h-relations, or
  - number of messages.
- In the book, the vector distribution is used to **balance communication**, i.e., uses the minimised communication volume to get minimised h-relations.