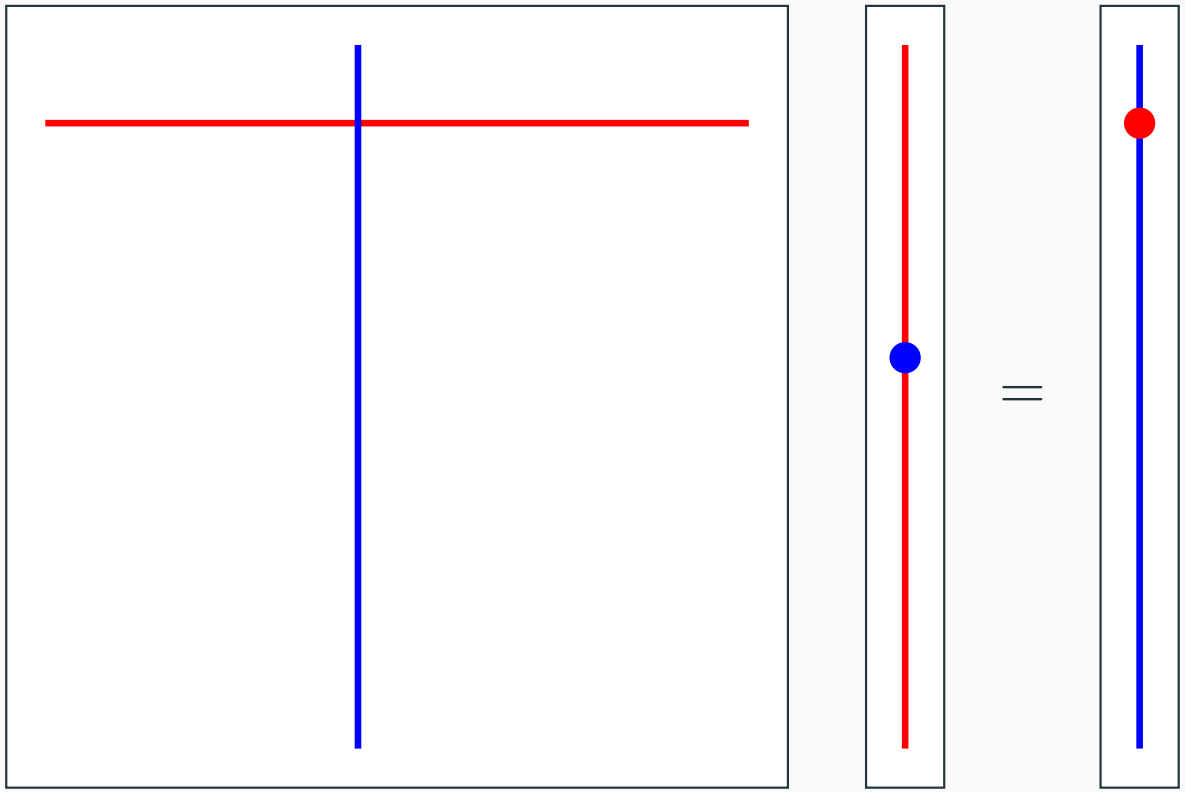
Sparse Matrix Partitioning

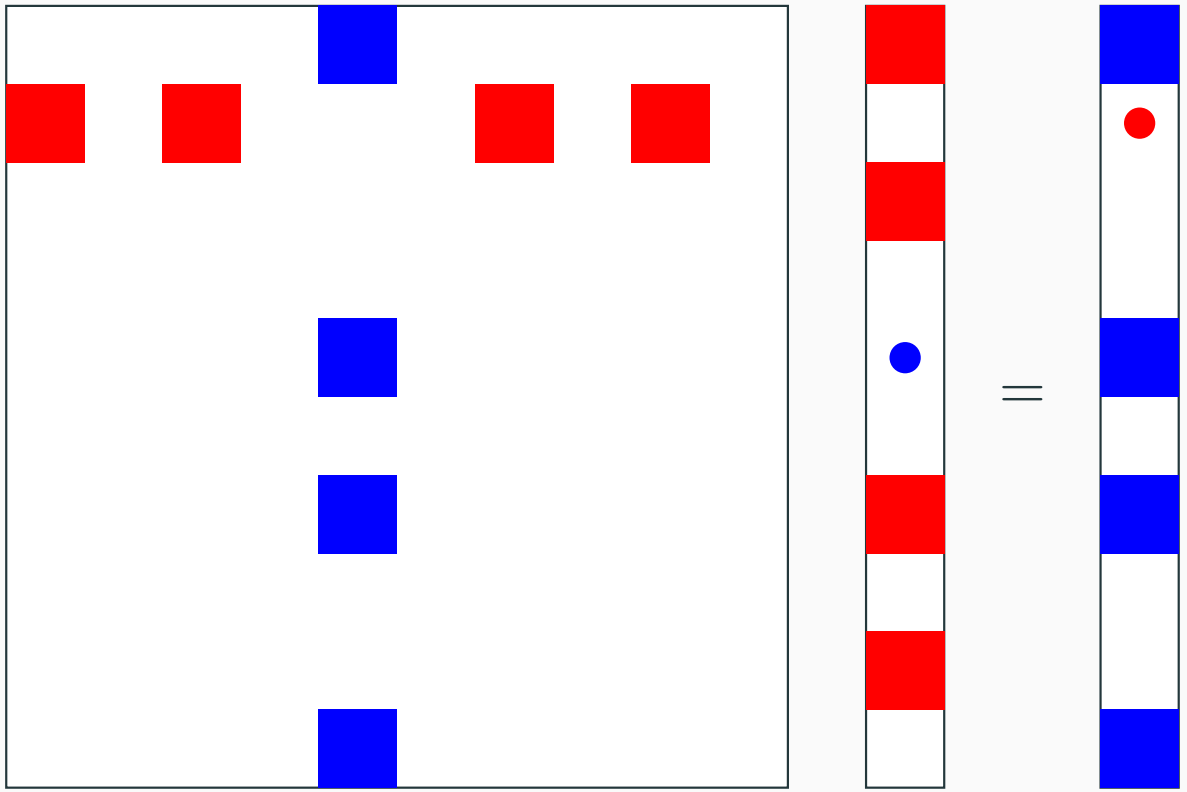
Jan-Willem Buurlage, CWI Amsterdam

Tomography seminar, DTU

Dense matrix-vector multiplication (GEMV)



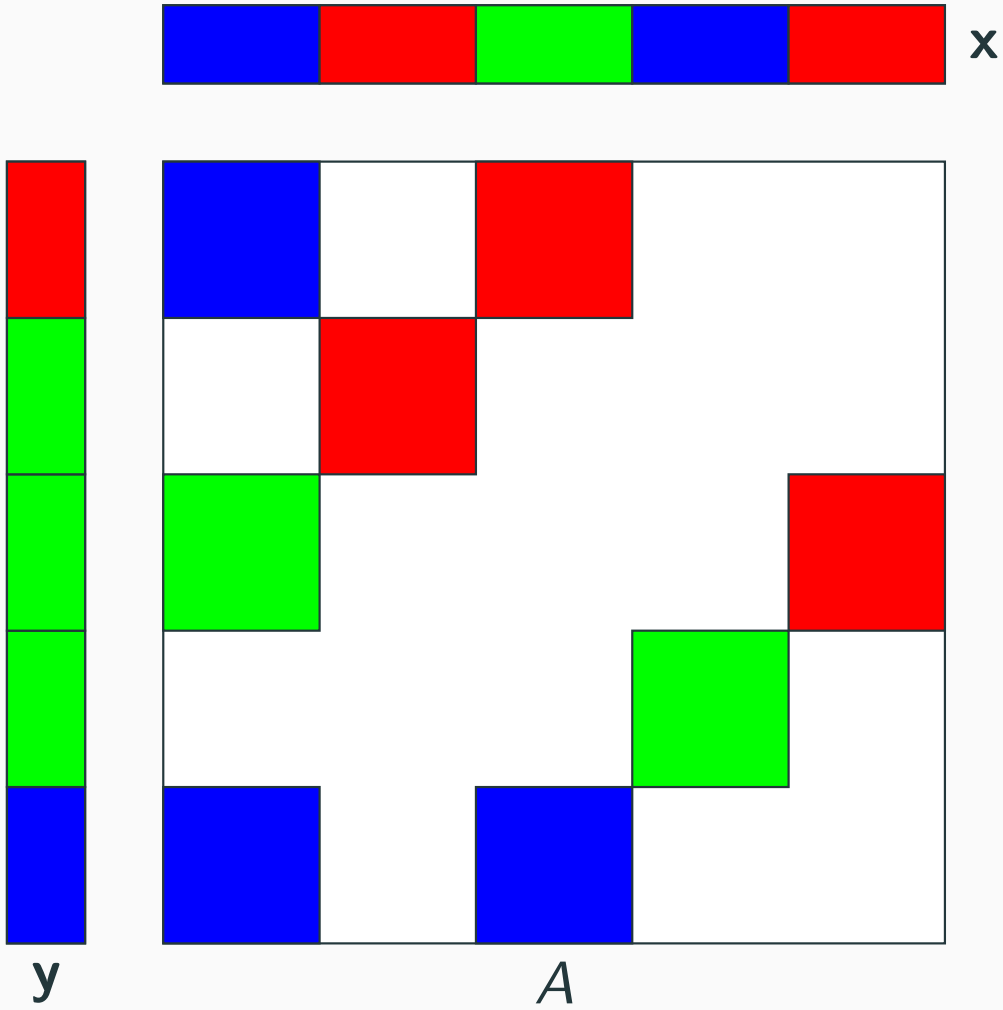
Sparse matrix-vector multiplication (SpMV)



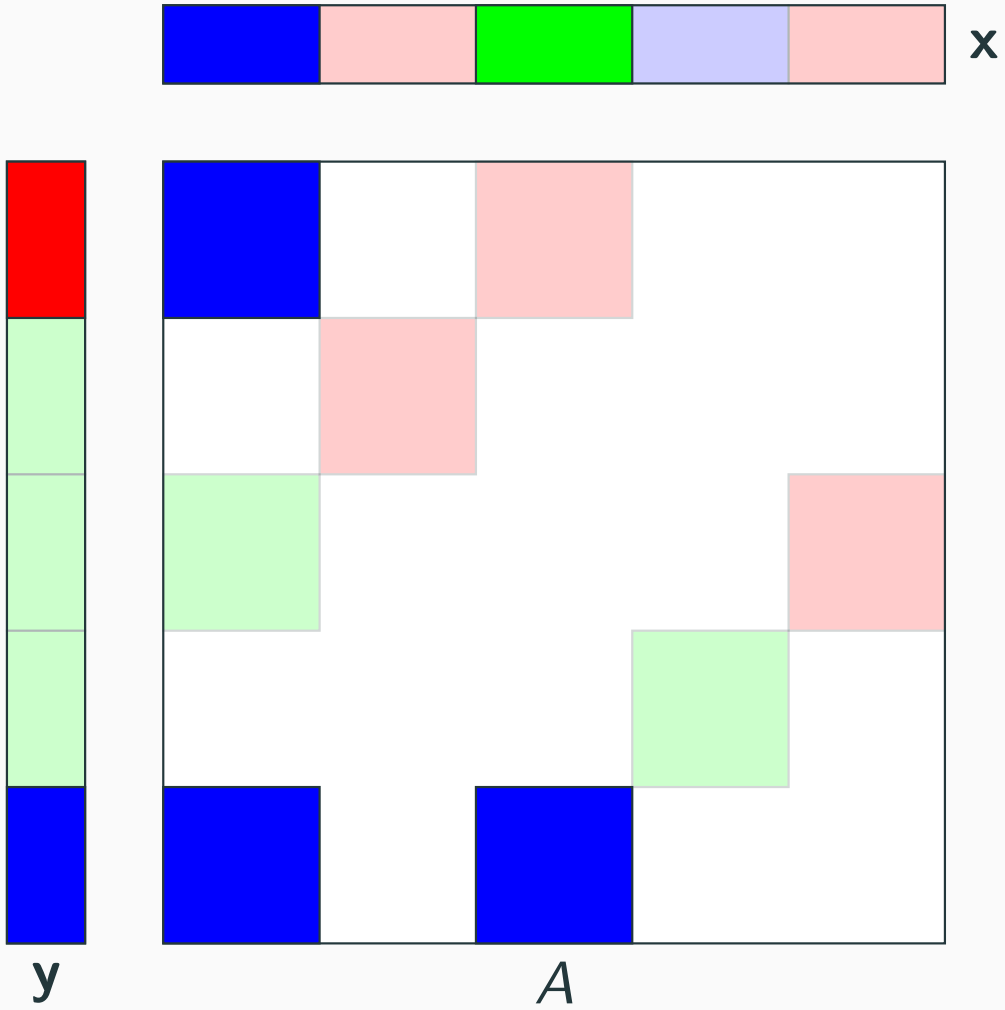
Parallel SpMV

- When performing an SpMV in parallel, we distribute the data $(A, \mathbf{x}, \mathbf{y})$ over processing elements.
- The distribution of the nonzeros of A are leading; the distribution of $\mathbf{x}$ and $\mathbf{y}$ follow.
- Two types of partitionings:
  - assign entire rows (or columns) to a single processor (1D partitioning).
  - treat all nonzeros independently (2D partitioning).

Distribution example



Distribution example (blue processor)



Parallel SpMV (Summary)

$A\mathbf{x} = \mathbf{y}$, from the viewpoint of processor $1 \leq s \leq p$:

1. Obtain the required non-local components of $\mathbf{x}$ (fan-out).
2. Compute the partial sums $(u_i)_s$ (local SpMV).
3. Communicate each non-local partial sum (fan-in).
4. Compute the local components of $\mathbf{y}$ using the received partial sums (reduction operation).

Partitioning quality

- **Question:** what makes a distribution good?
- Roughly the same number of nonzeros to each processor:

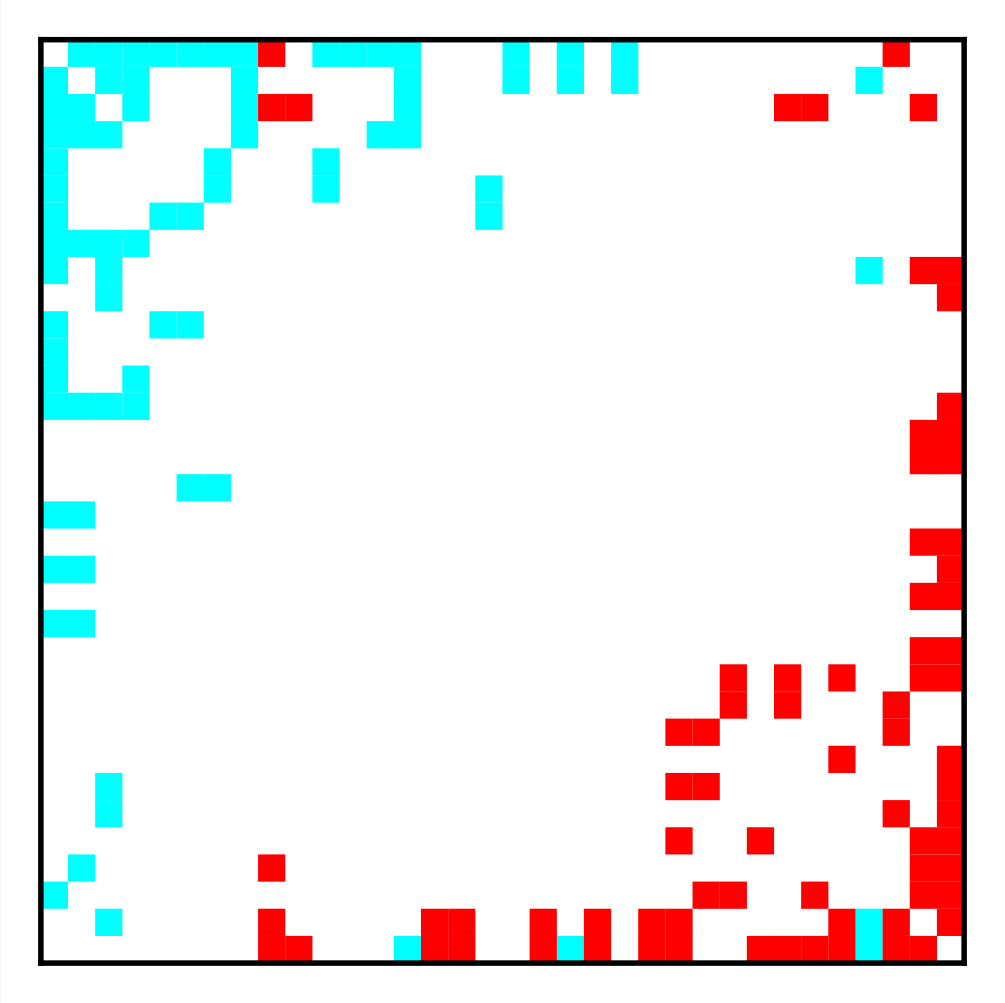
$$|A_s| \leq (1+\epsilon)\frac{|A|}{p}$$

- Minimize communication volume V:

$$V = \underbrace{\sum_{j=1}^{n}(\mu_j - 1)}_{\text{fan-out}} + \underbrace{\sum_{i=1}^{m}(\lambda_i - 1)}_{\text{fan-in}},$$

where λ_i denotes the number of processors that hold a portion of the ith row, and similarly μ_j for the jth column.

Communication volume



- `karate`: optimal is $V = 8$

Hypergraph models

- We want to find a p-way partitioning of A while minimizing V.
- Look at hypergraph structures $\mathcal{H}$ associated to the sparsity pattern of the matrix A.

Definition

A hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ is a set of vertices $\mathcal{V}$, together with a set of *nets* $\mathcal{N}$ with $n_i \in \mathcal{N}$ a subset of $\mathcal{V}$.

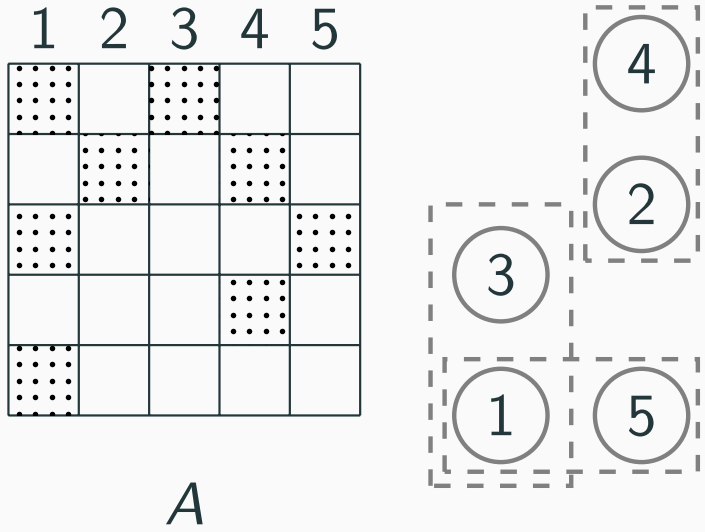
Hypergraph models (cont.)

- We model the matrix A as a set of vertices $\mathcal{V}$, and want to find a p-way partitioning of $\mathcal{V}$.
- We consider three different models:

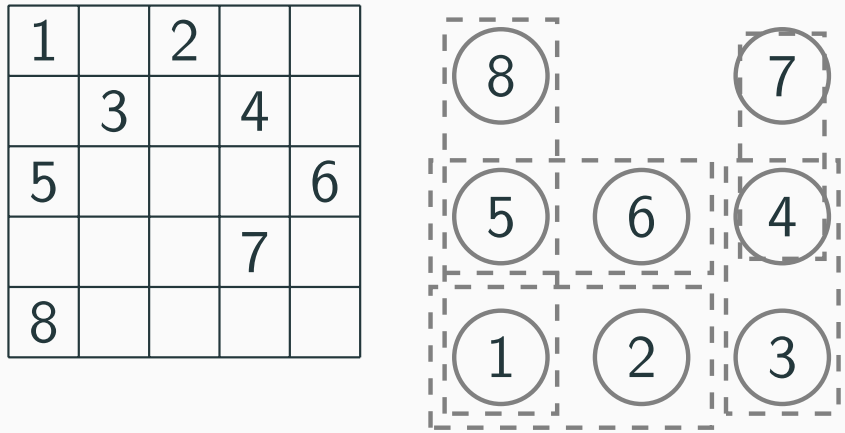
name	*vertices*	*nets*
row-net	columns	rows
column-net	rows	columns
fine-grain	nonzeros	rows and columns

Hypergraph partitioning

row-net:



fine-grain:



Hypergraph partitioning

- $(\lambda - 1)$-metric of a hypergraph partitioning:

$$V = \sum_{n \in \mathcal{N}} (\lambda(n) - 1),$$

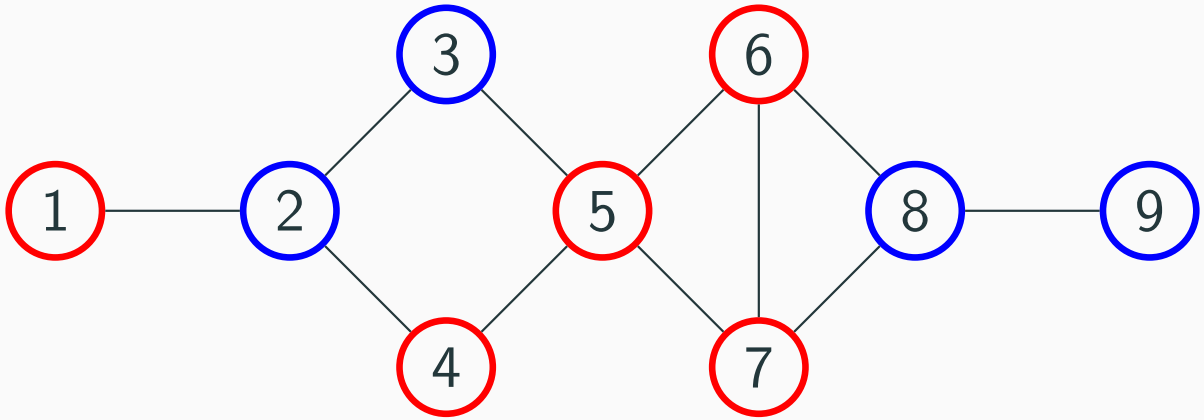
  where $\lambda(n)$ counts the number of non-empty parts in the net n.
- The communication volume of a 1D row, 1D column or 2D partitititioning of A is equal to the $(\lambda - 1)$-metric of the column-net, row-net or fine-grain model respectively.

Label propagation on graphs

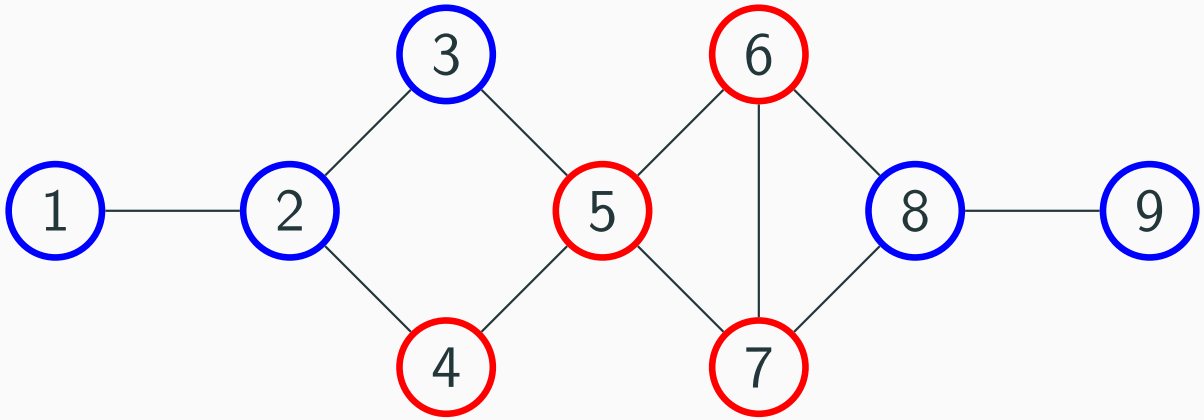
- *Goal:* Given a graph $G = (V, E)$, obtain a p-way partitioning that minimizes the *edge-cut* (i.e. the number of edges between different parts).
- Use label propagation. Here we describe a version of the **PULP** algorithm[1]:
  - Assign to each $v \in V$ a random label $L(v) \in \{1, \ldots, p\}$.
  - Consider each vertex v in turn, and update to the majority label amongst its neighbours. Ties are broken randomly.

[1] Slota, Madduri, and Rajamanickam '14

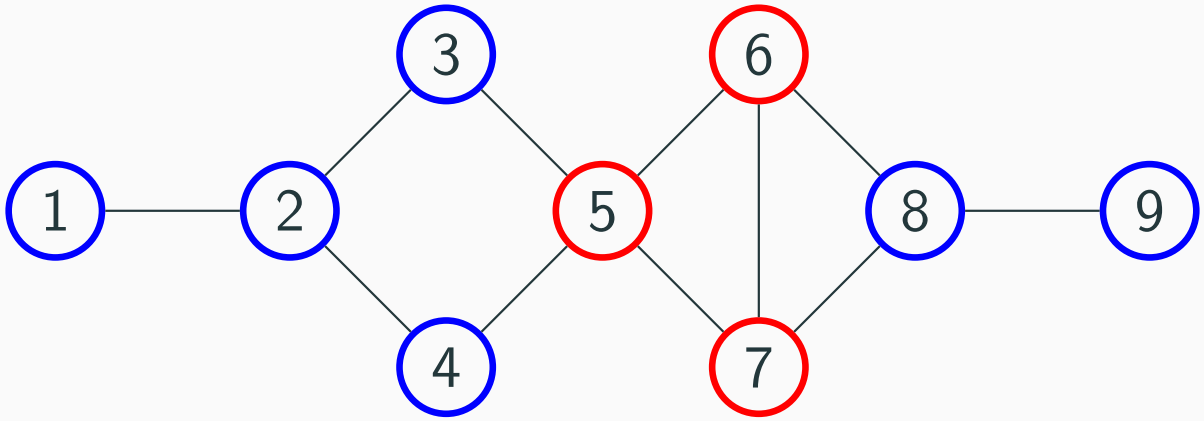
Label propagation (1)



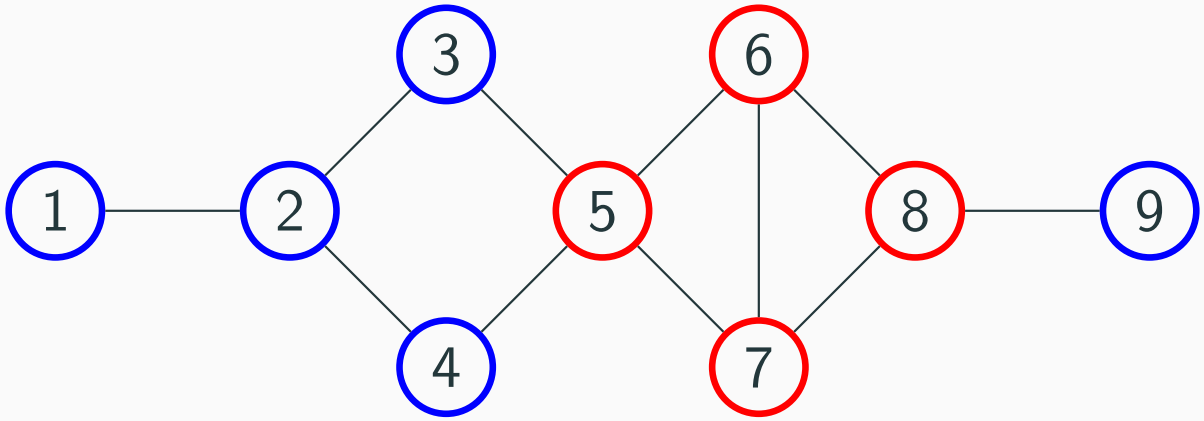
Label propagation (2)



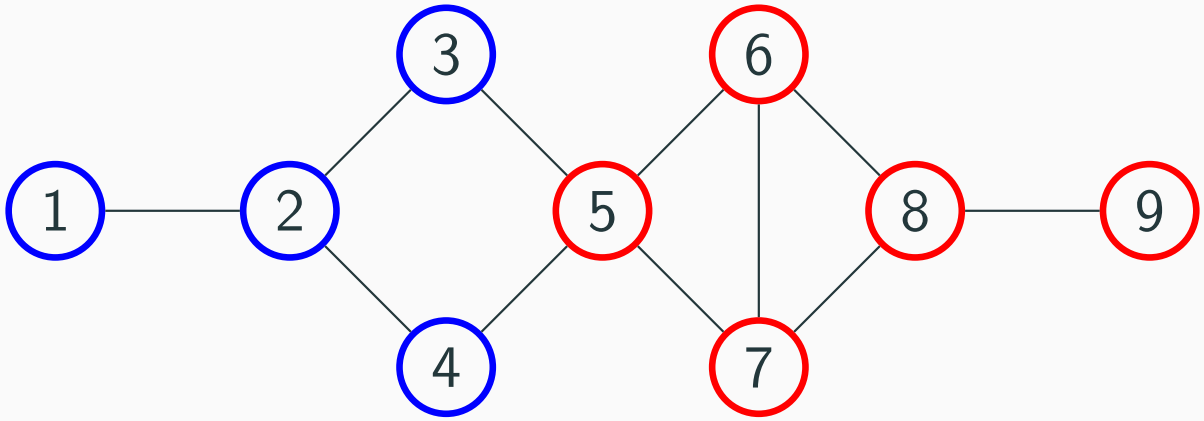
Label propagation (3)



Label propagation (4)



Label propagation (5)



Label propagation for graph partitioning

- Update the label of $v \in V$ by counting the labels around it:

$$C_s(v) = \sum_{(v,u) \in E} \mathbf{1}_s(L(u)).$$

- Form clusters around vertices of high degree, in the hope that vertices of low degree end up at the boundary of a part:

$$C_s(v) = \sum_{(v,u) \in E} \mathbf{1}_s(L(u)) \times \deg(u).$$

- Prevent the algorithm from assigning a single label to all vertices by also taking into account the current size of a part.

Label propagation on hypergraphs

We generalize this method to hypergraphs[2]:

- C_s takes the following form, with w a weight function that has to be chosen:

$$C_s(v) = \sum_{n \in \mathcal{N}_v} w(n, s).$$

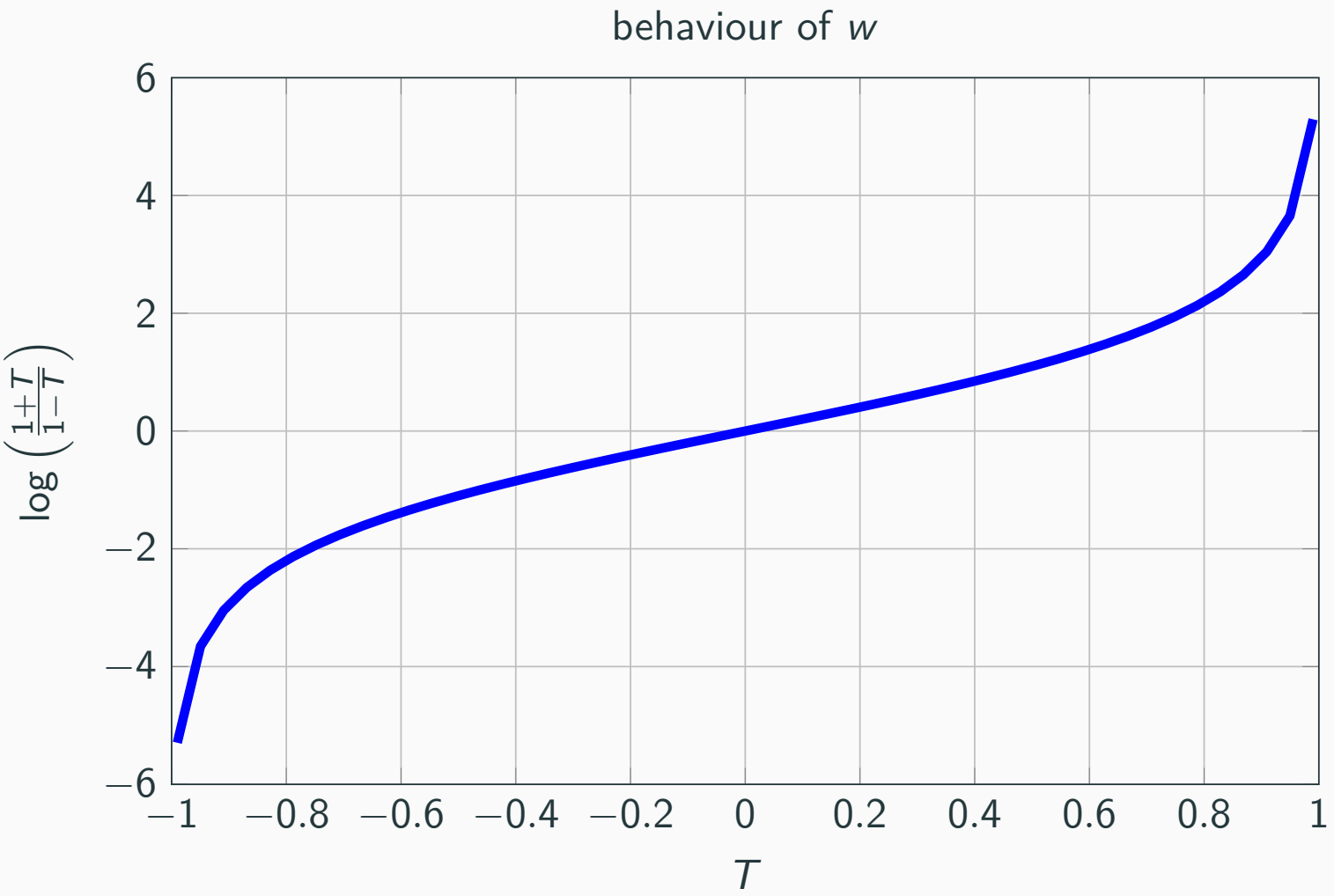
- For the LV-metric, w should encode two key ideas:
  - Do not introduce new labels to a net, and try to eliminate uncommon labels.
  - When net is almost *pure* differently labeled vertices should strongly prefer taking over the majority label.

[2]Self-Improving Sparse Matrix Partitioning and Bulk-Synchronous Pseudo-Streaming, MSc Thesis, JB

Label propagation on hypergraphs

- Relative size of label s in net n: $|\{v \in n \mid L(v) = s\}|/|n|$.
- Scale the relative size T to lie in the range $[-1, 1]$. $T(n, s)$ equal to -1 or 1 means none or all vertices have label s respectively.
- Take w as a function of T

$w(T)$


behaviour of w
$\log\left(\frac{1+T}{1-T}\right)$
6
4
2
0
−2
−4
−6
−1
−0.8
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
0.8
1
T

Initial partitioning

- *Small nets* are most easily kept pure, ignore larger nets at first.
- We construct a chain of growing hypergraphs:

$$\mathcal{A}_0 \subset \mathcal{A}_1 \subset \mathcal{A}_2 \subset \ldots \subset A_M = \mathcal{H}.$$

Here, $\mathcal{A}_i = \{\mathcal{V}, \mathcal{N}_i\}$, and $\mathcal{N}_i$ can be taken to hold e.g. the 2^i smallest nets.

Label propagation based hypergraph partitioning

- Begin with some initial partitioning, e.g. distribute the vertices cyclically.
- For the first $1 \leq i < M$ iterations, consider each vertex $v \in \mathcal{V}$ in turn. Choose the label s that maximizes $C_s(v)$ in the hypergraph $\mathcal{A}_i$, and assign to v this label.
- For $i \geq M$ we put $\mathcal{A}_i = \mathcal{H}$, and we perform this label propagation on the entire hypergraph $\mathcal{H}$.

Iterative partitioning

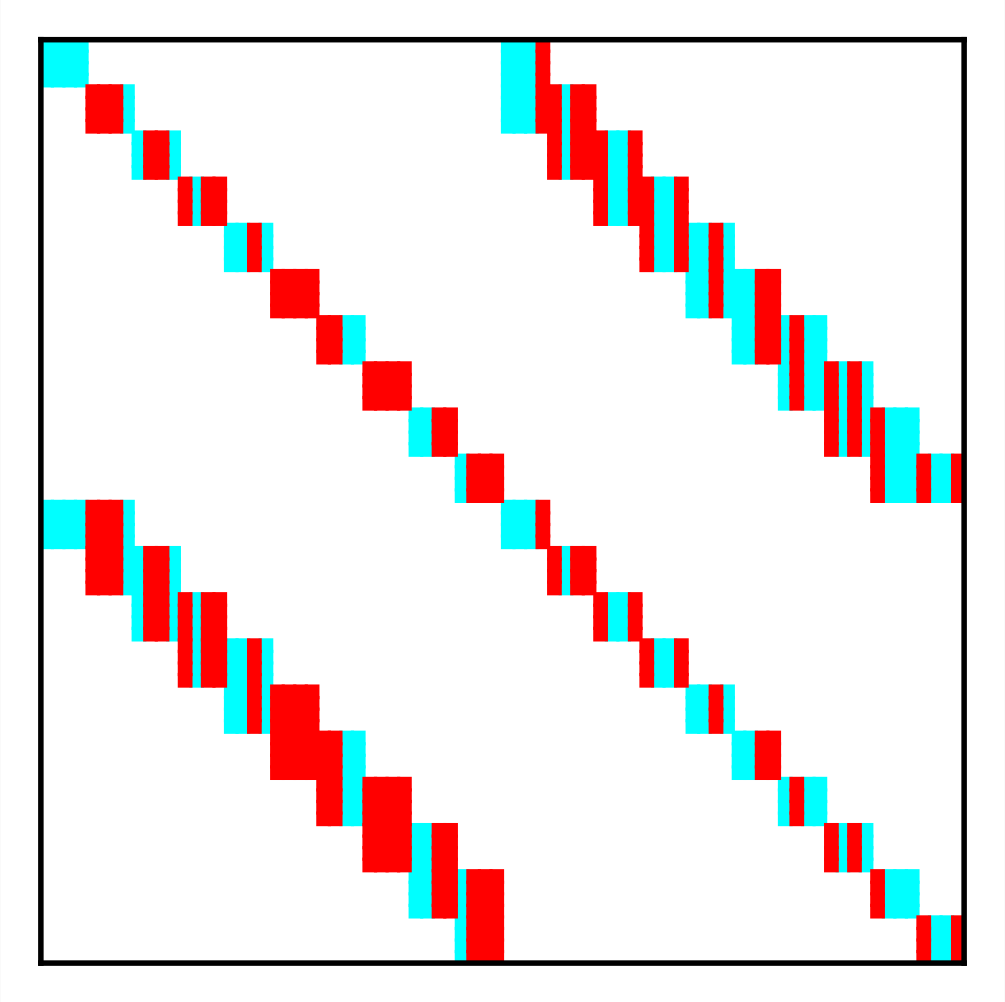


Figure 1

Iterative partitioning

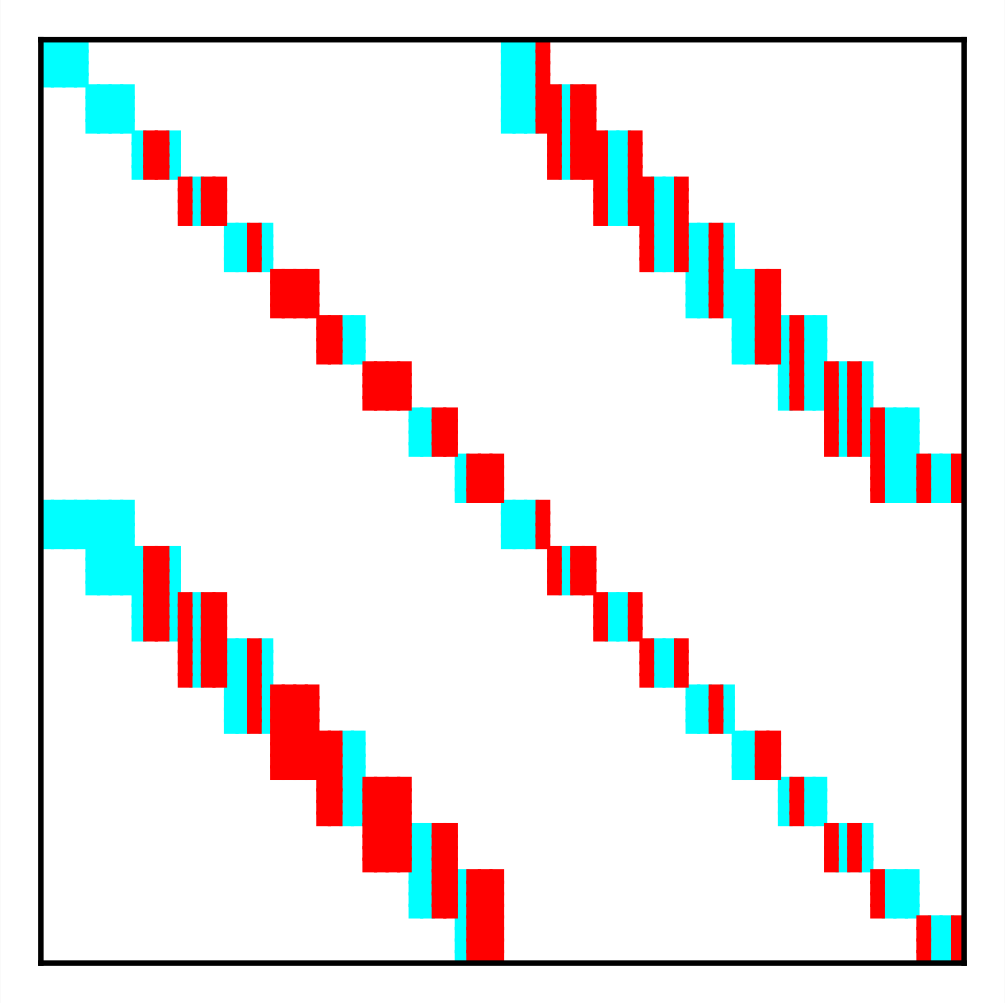


Figure 2

Iterative partitioning

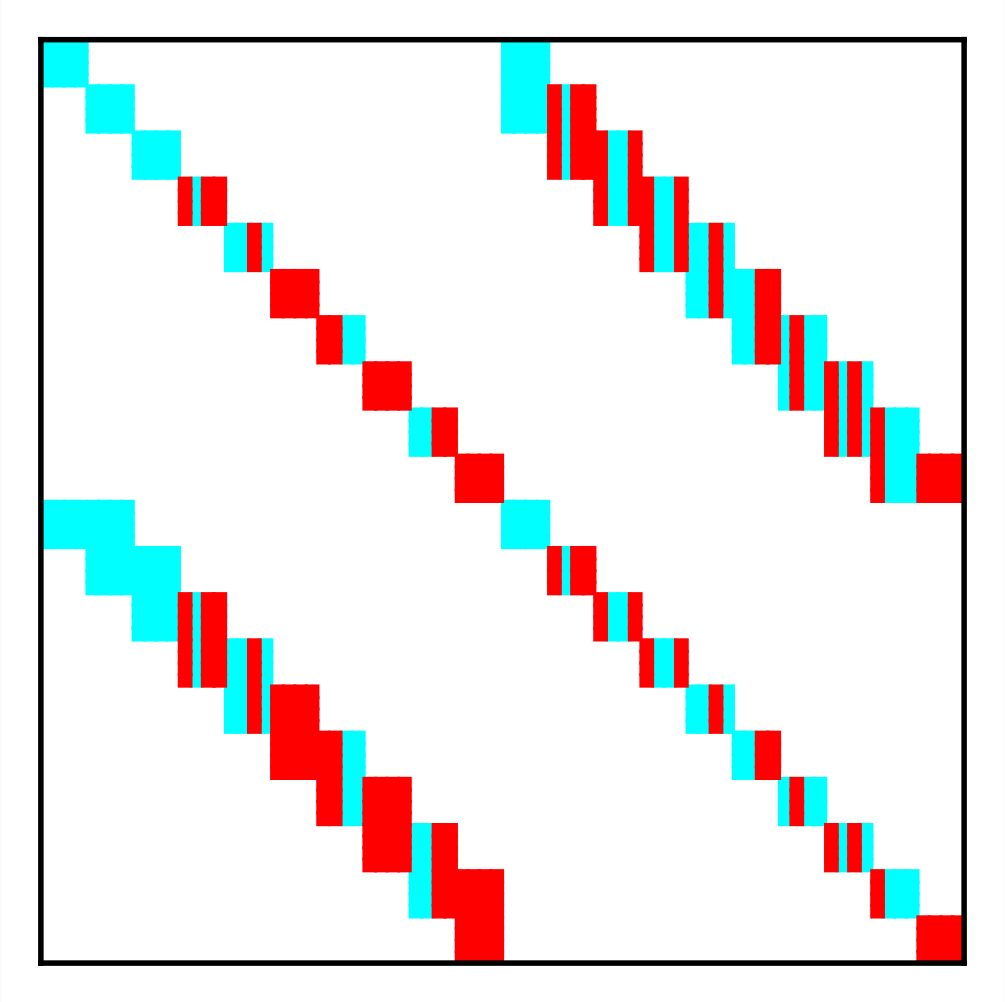


Figure 3

Iterative partitioning

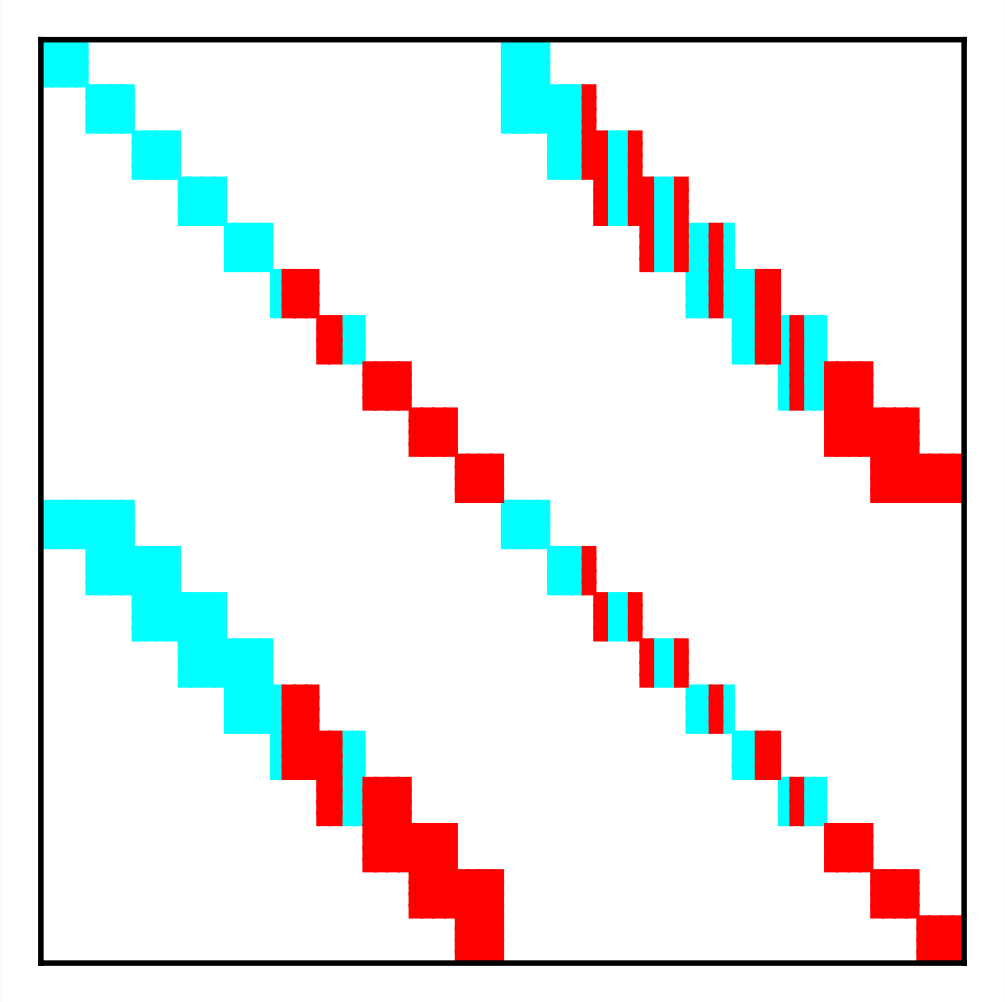


Figure 4

Iterative partitioning

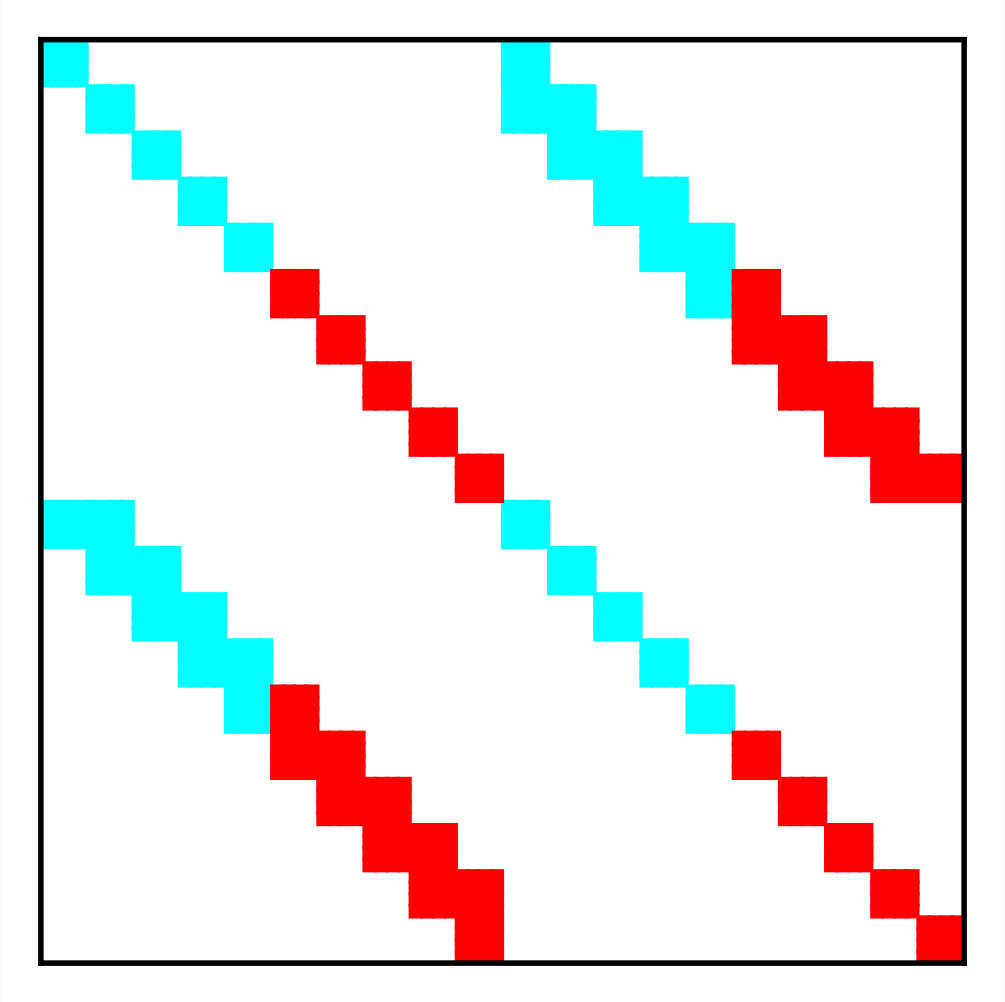
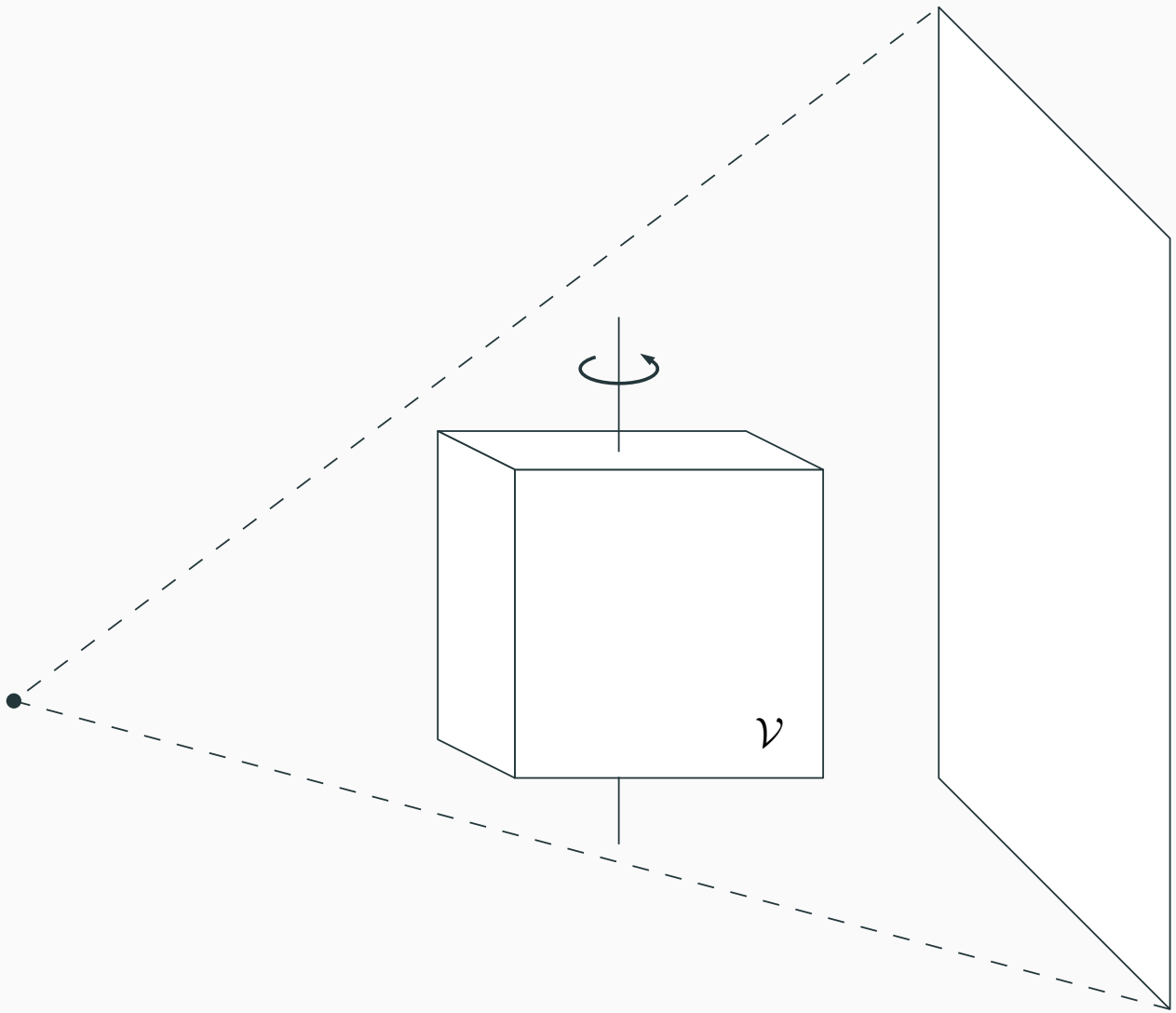


Figure 5

Tomography



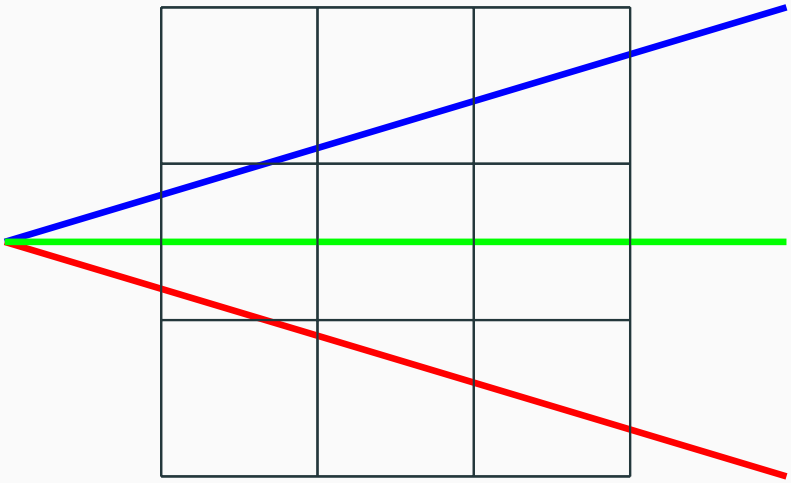
Tomographic reconstruction

- *Projection matrix* W, solve:

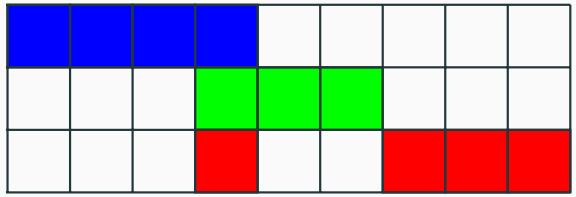
$$W\mathbf{x} = \mathbf{b},$$

  with $\mathbf{x}$ the *image*, and $\mathbf{b}$ the *projection data*.
- Rays, from the source to a detector pixel, define rows. Each column of the matrix is a volume element, or *voxel*.
- Each intersection of a ray with a voxel, gives rise to a nonzero in W. Note: W is sparse, with n voxels we have $\mathcal{O}(n^{1/3})$ nonzeros in each row.
- For each projection image, we obtain a block of rows.

Example



$\mathcal{V}$



A

Large-scale tomography

- For tomographic reconstruction, the SpMV's $W\mathbf{x}$ and $W^T\mathbf{y}$ are the most expensive operations.
- 3D volumes with at least 1000^3 voxels. Already at this resolution, W has $\mathcal{O}(10^{12})$ entries $\Rightarrow$ TB's!
- Can not be stored explicitely, instead generated from the acquisition geometry.

Large-scale tomography (cont.)

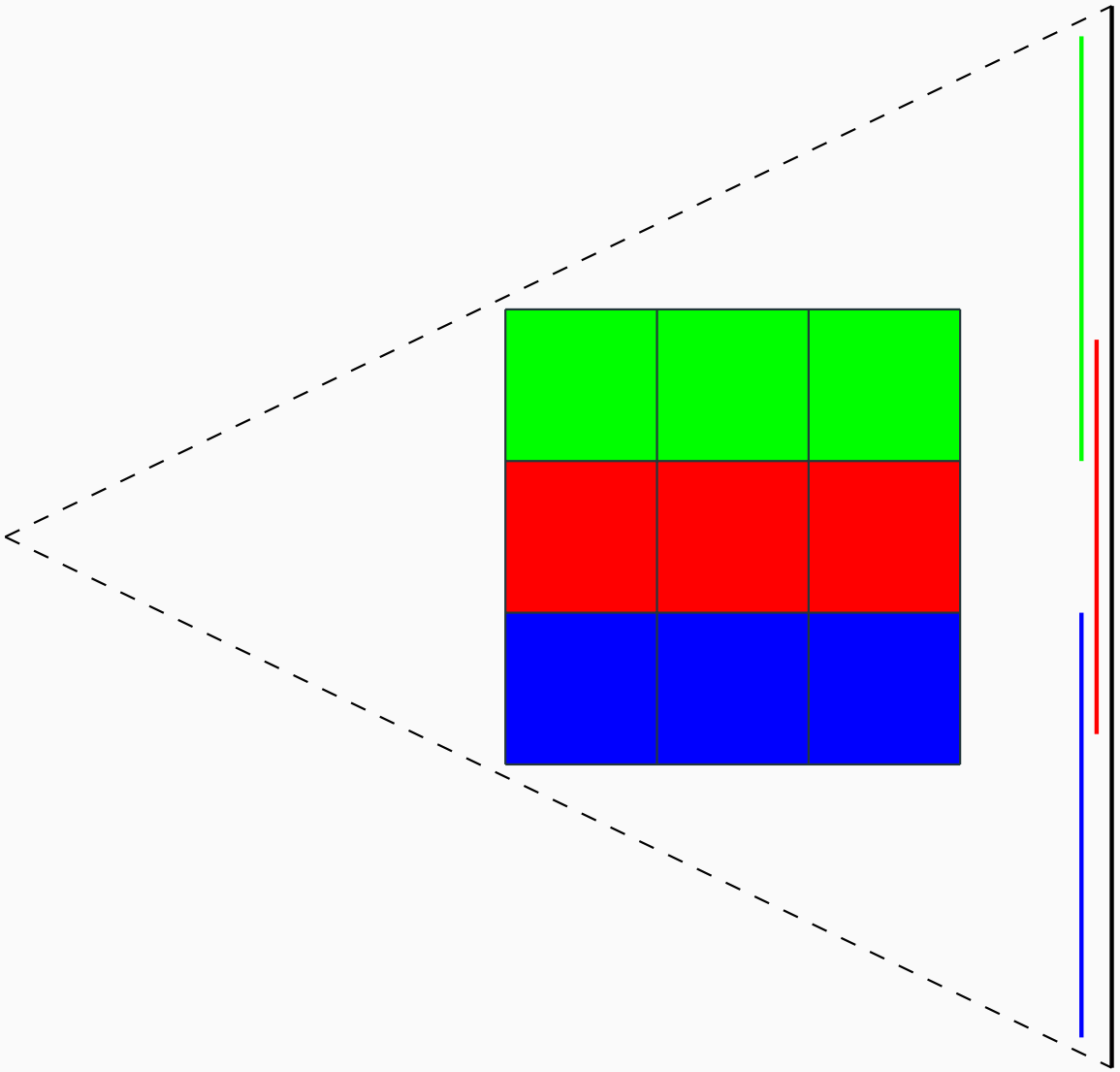
- We want to parallelize the forward projection and backward projection operations
- How to distribute W? Naive choices lead to prohibitively large communication sizes
- Available sparse matrix partitioning methods do not apply, since the hypergraph models are at least of size $\mathcal{O}(\text{nnz}(A))$.

Geometric partitioning

- We exploit the geometric structure of the problem to find a partitioning[3]
- Generate a cuboid partitioning of the object volume, corresponding to a 1D column partitioning
- The communication volume is equal to the total *line cut*, the number of interfaces between parts crossed by a ray.

[3] Joint work with Rob Bisseling (UU) and Joost Batenburg (CWI)

Example



Recursive bisectioning

- Idea: Split the volume into two subvolumes recursively.

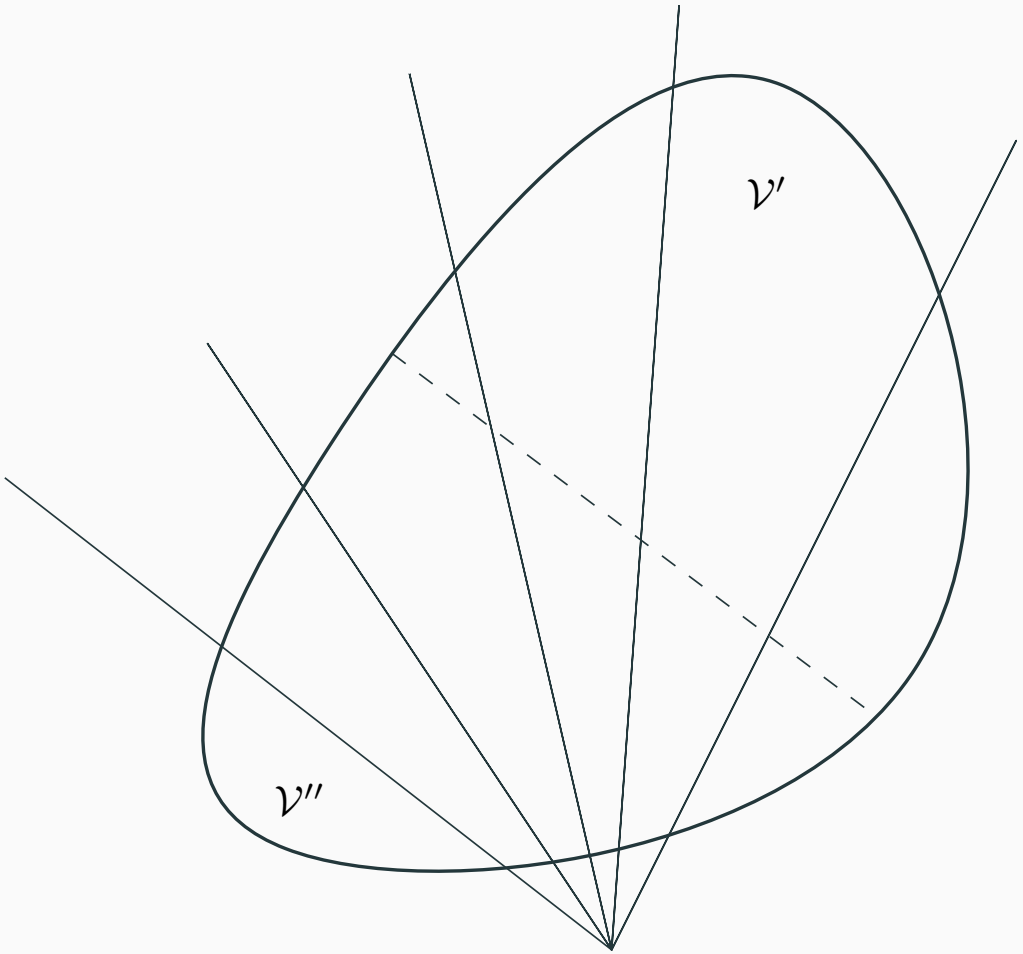
Theorem

Let $\mathcal{V} = \mathcal{V}_1 \cup \ldots \cup \mathcal{V}_n$ be a cuboid partitioning. Then for any acquisition geometry $\mathcal{G}$ we have:

$$V_{\mathcal{G}}(\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_n) = V_{\mathcal{G}}(\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_{n-1} \cup \mathcal{V}_n) + V_{\mathcal{G}}(\mathcal{V}_{n-1}, \mathcal{V}_n).$$

- Conclusion: recursively bisecting is OK

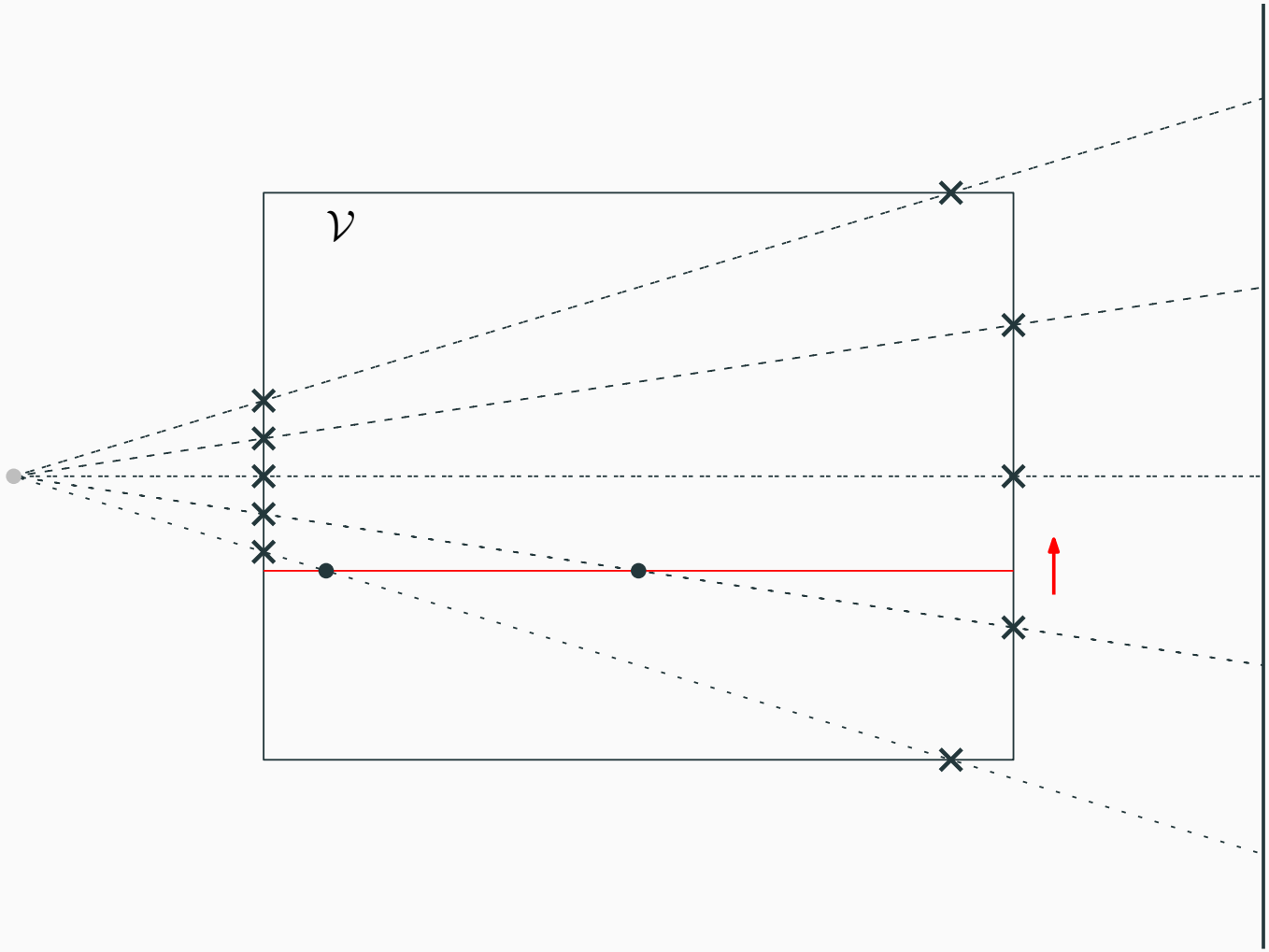
Interface intersection



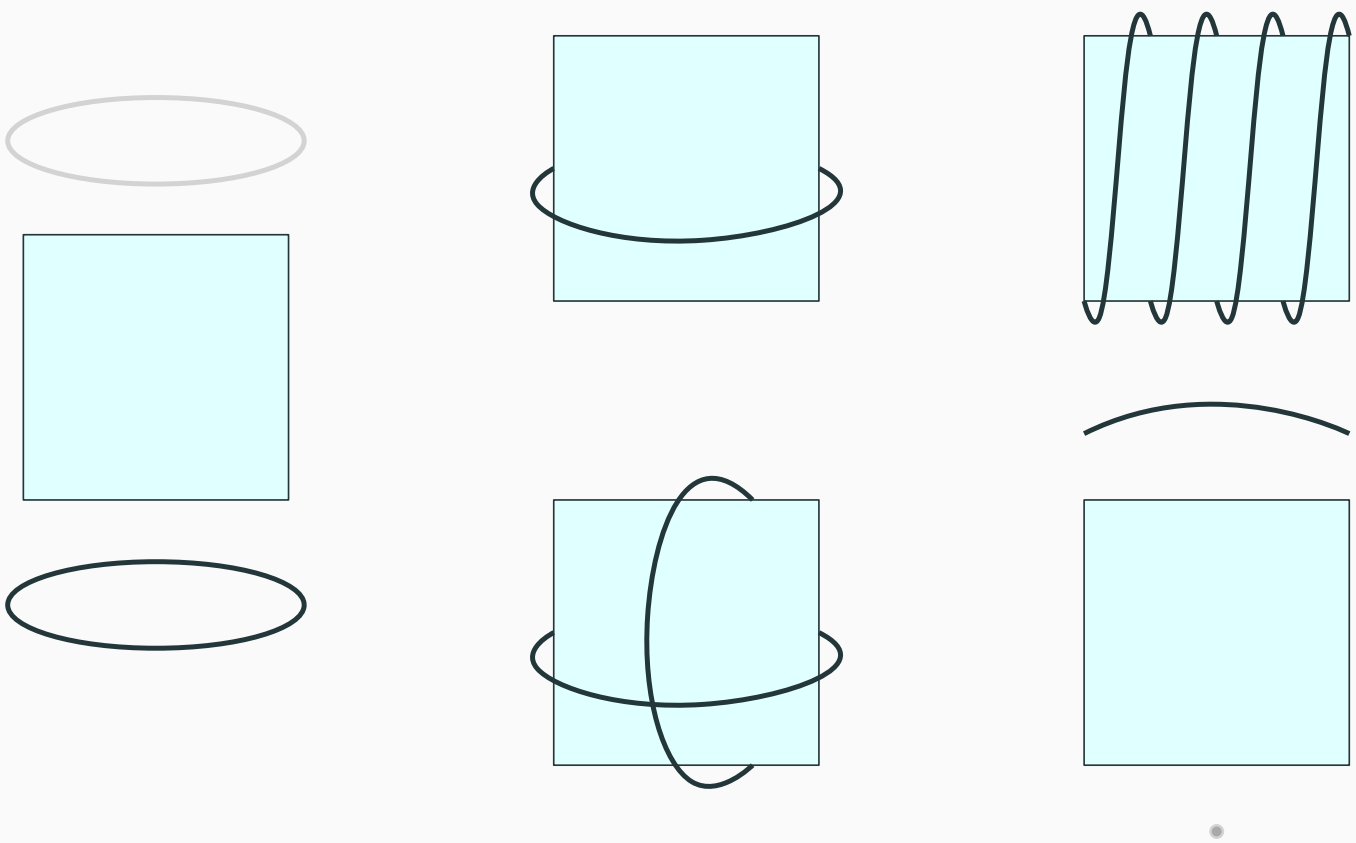
Bisectioning algorithm

- Choose the splitting interface with the minimum number of rays passing through it
- Evenly distribute the computational work
- Imagine sweeping a candidate interface along the volume, keep track of the current number of rays passing through. Only changes at coordinates where a line intersects the boundary!

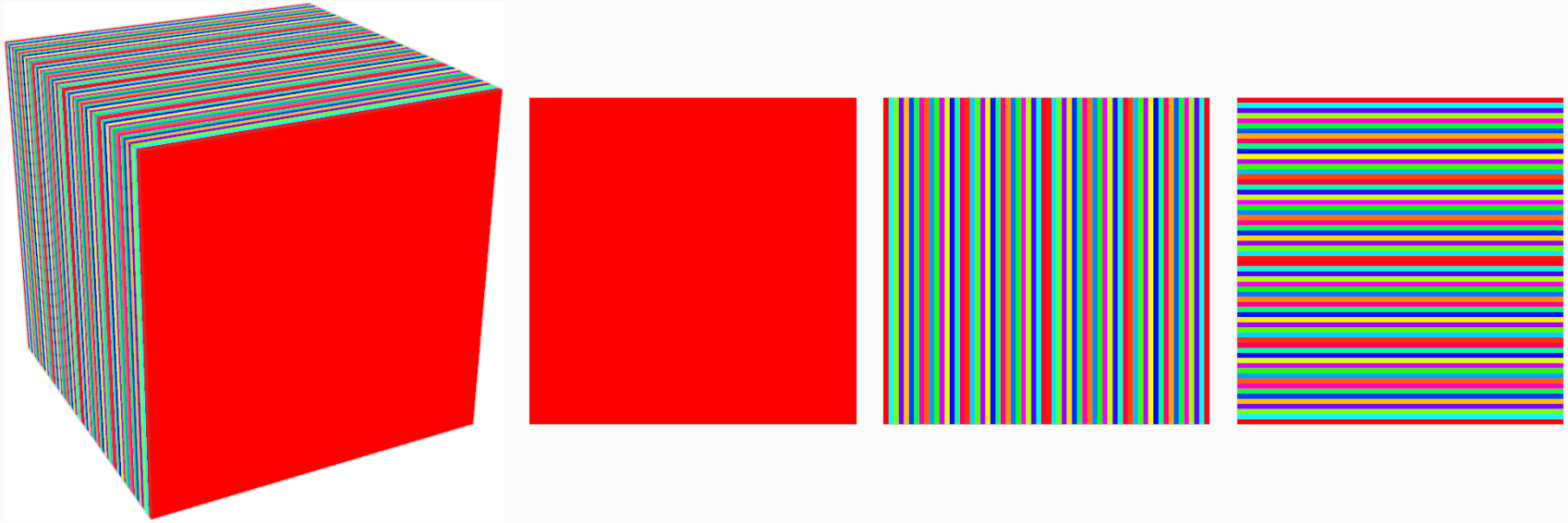
Plane sweep



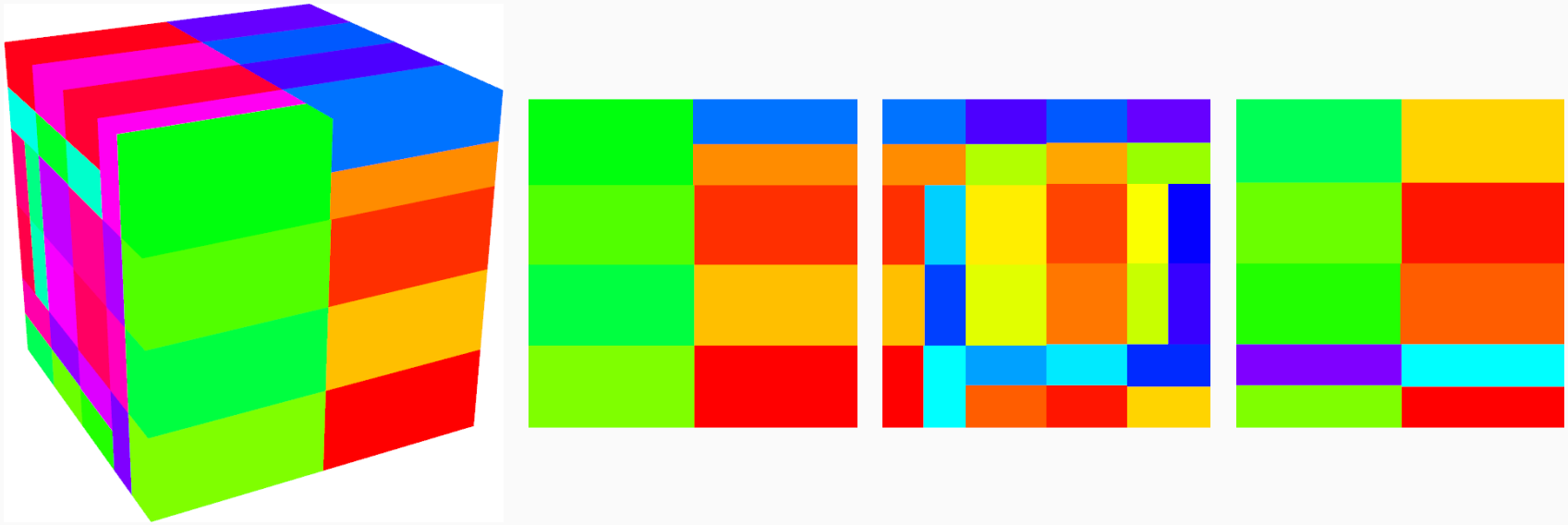
Acquisition geometries



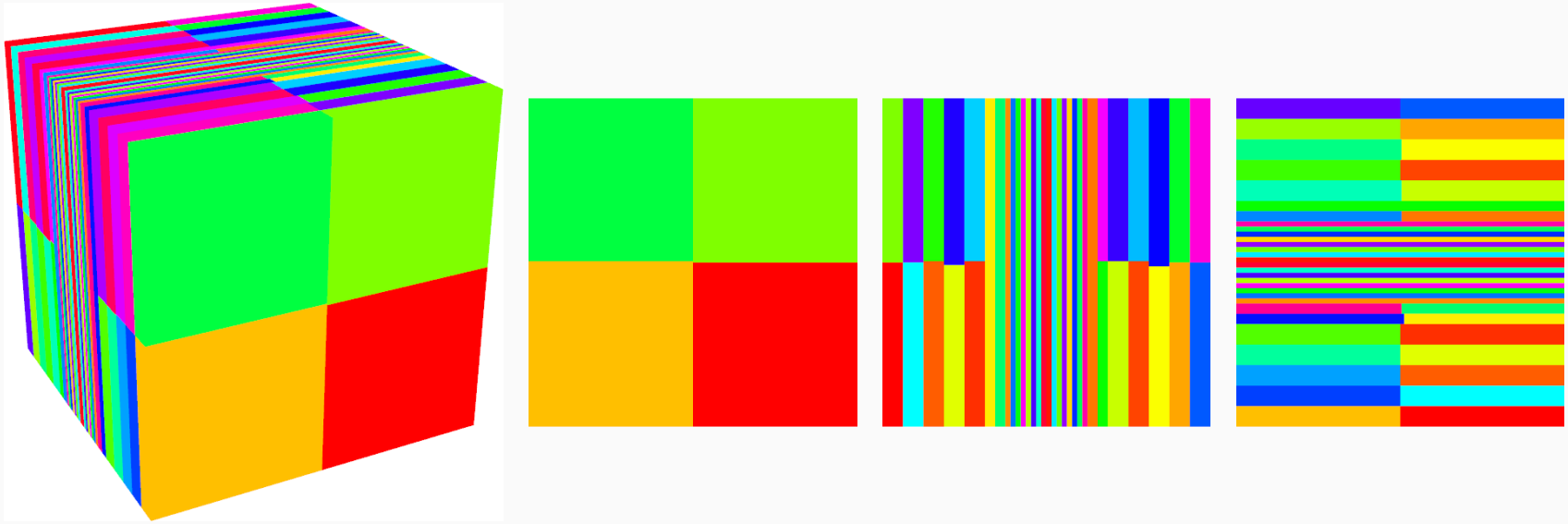
Results (SAPB)



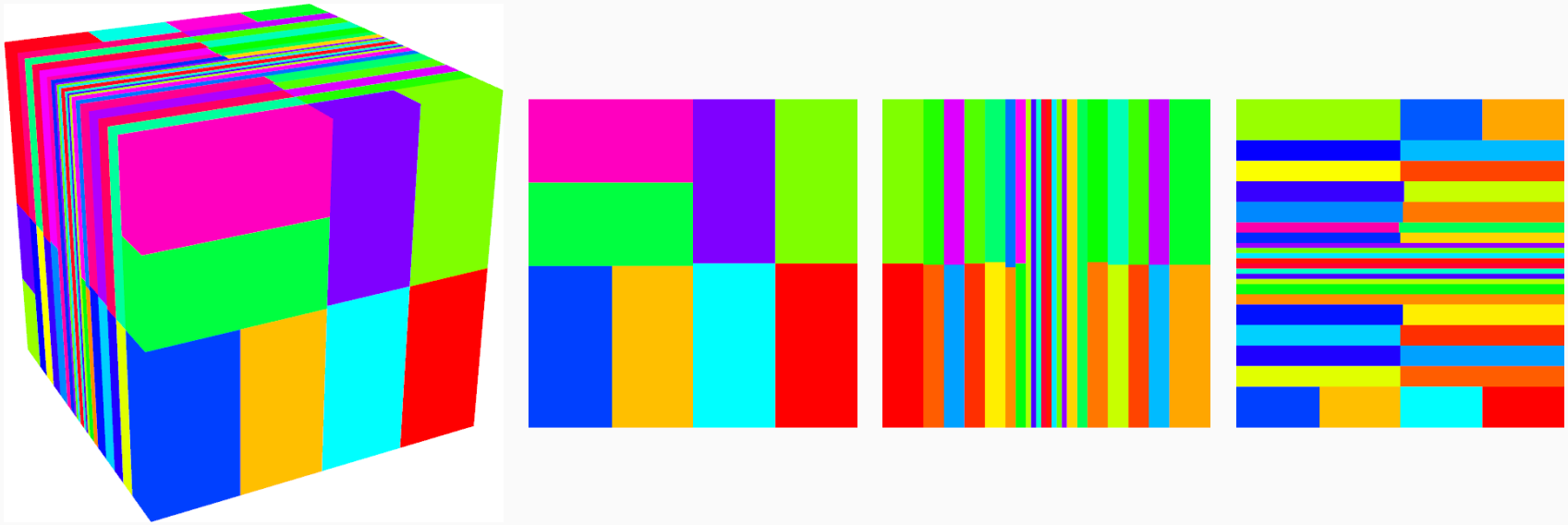
Results (DAPB)



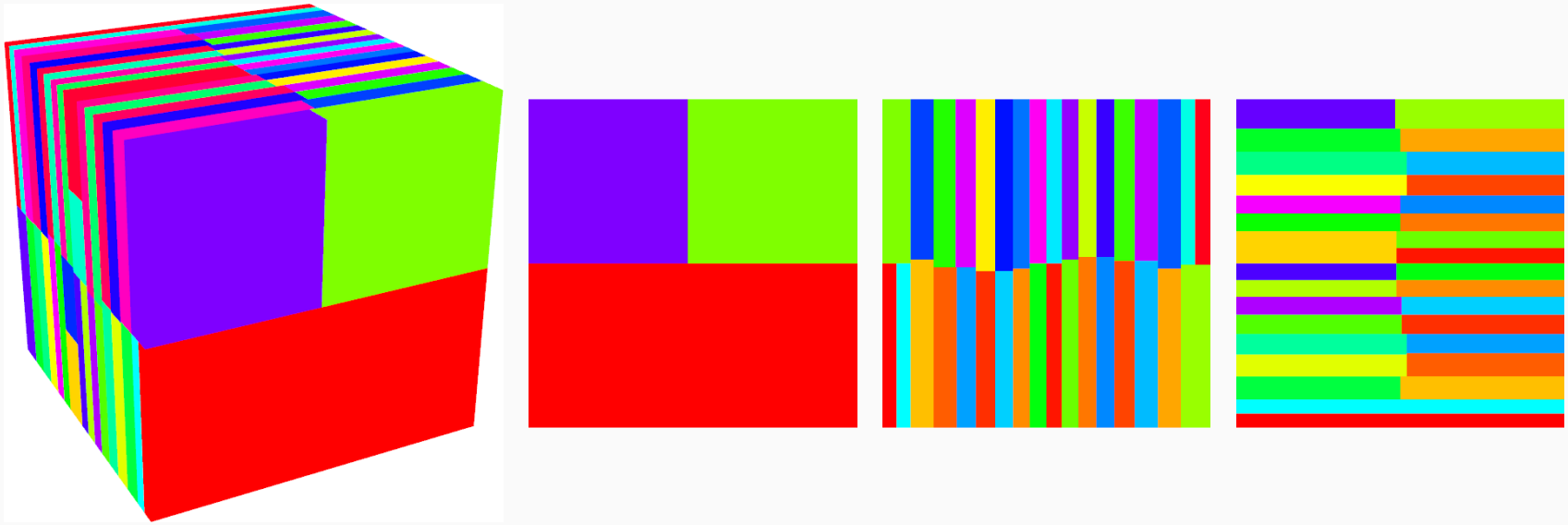
Results (CCBn)



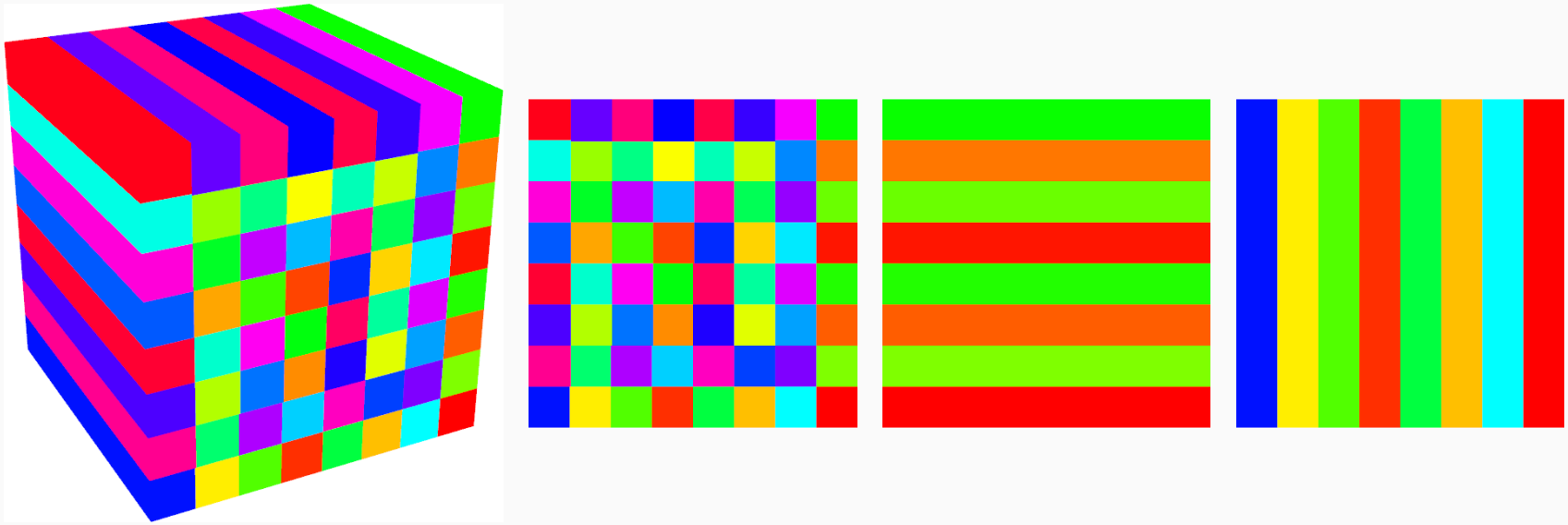
Results (CCBw)



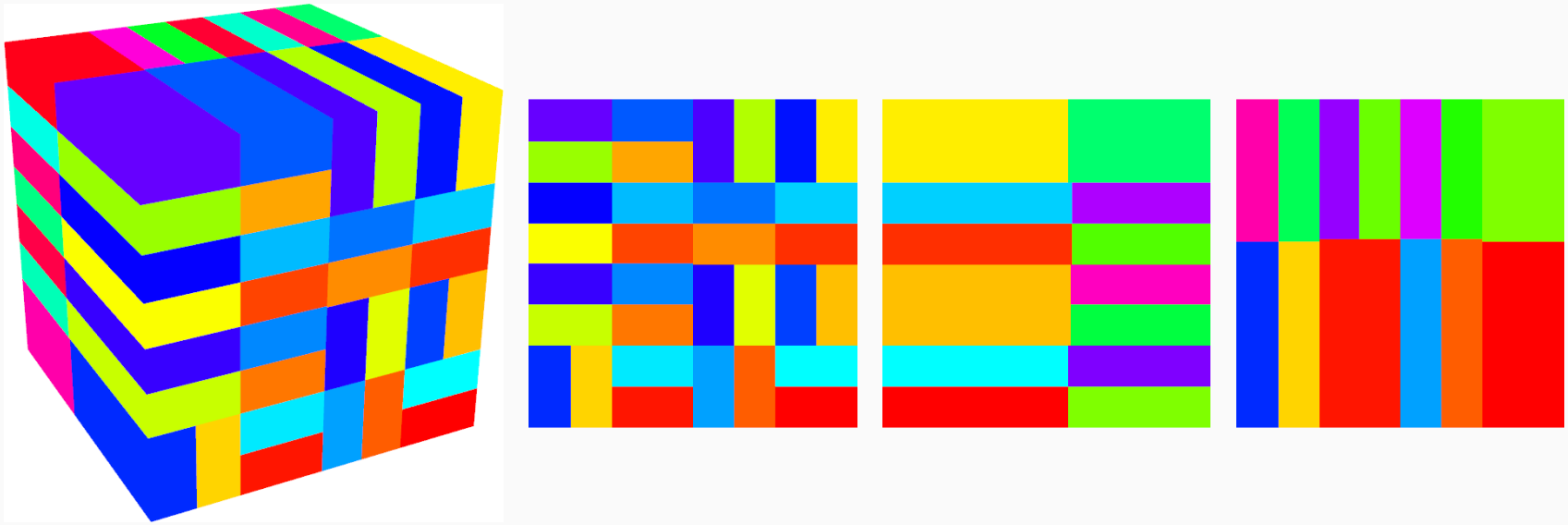
Results (HCB)



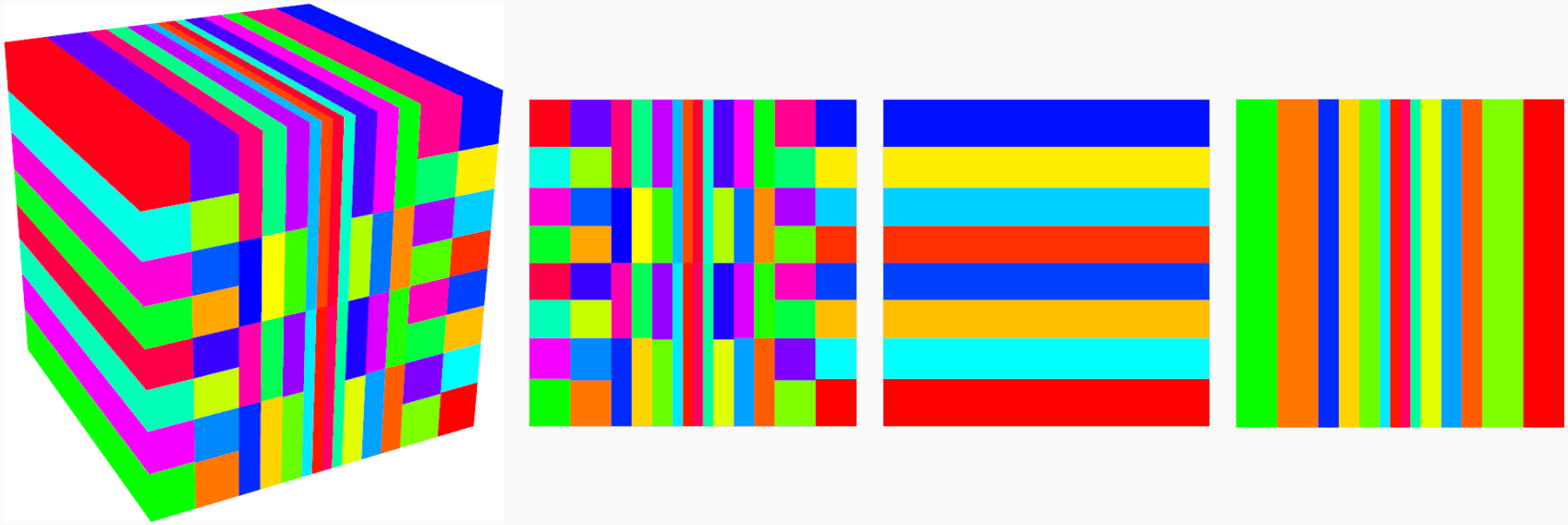
Results (LAMn)



Results (LAMw)



Results (TSYN)



Movie

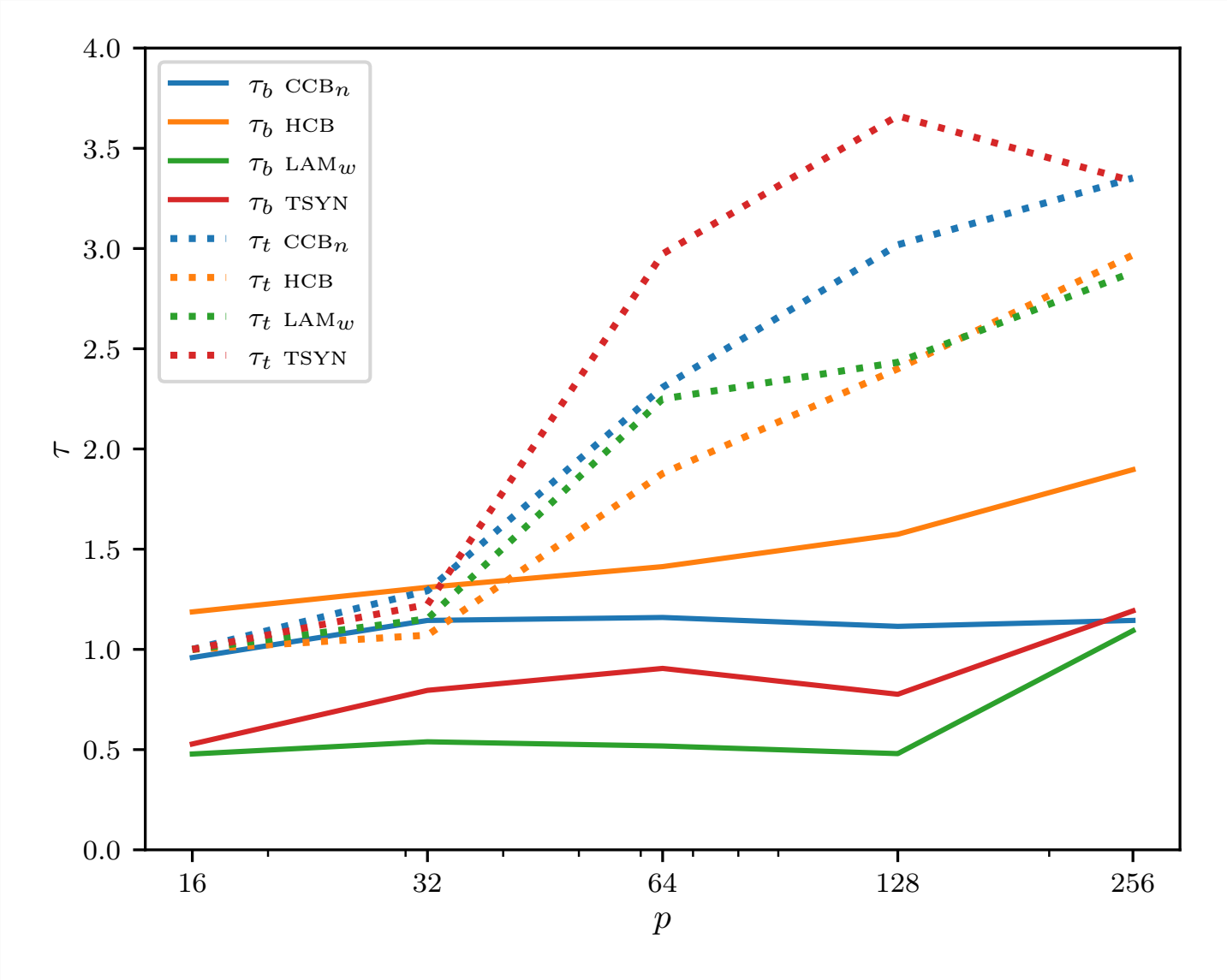
<Partitioning movie>

Results (Communication volume)

- Results for $p = 256$

Geometry	V (slab)	V (grb)	Improvement
SAPB	0	0	0%
DAPB	1×10^{10}	8×10^{8}	92%
CCBn	1×10^{9}	3×10^{8}	69%
CCBw	2×10^{9}	4×10^{8}	82%
HCB	2×10^{9}	4×10^{8}	71%
LAMn	3×10^{9}	4×10^{8}	89%
LAMw	5×10^{9}	6×10^{8}	90%
TSYN	2×10^{9}	3×10^{8}	87%

Results (Communication time)



Thank you

Questions?