Partitioning for Parallel Sparse Matrix-Vector Multiplication

Michael Wolf

CS 591MH

September 13, 2007

Parallel Matrix-Vector Multiplication

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \begin{bmatrix} 1 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 1 & 9 & 0 & 5 & 0 & 0 & 0 \\ 8 & 0 & 1 & 7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 & 0 & 0 & 7 \\ 0 & 0 & 0 & 0 & 1 & 8 & 0 & 0 \\ 0 & 4 & 0 & 0 & 3 & 1 & 3 & 0 \\ 0 & 0 & 0 & 6 & 0 & 9 & 1 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}$$

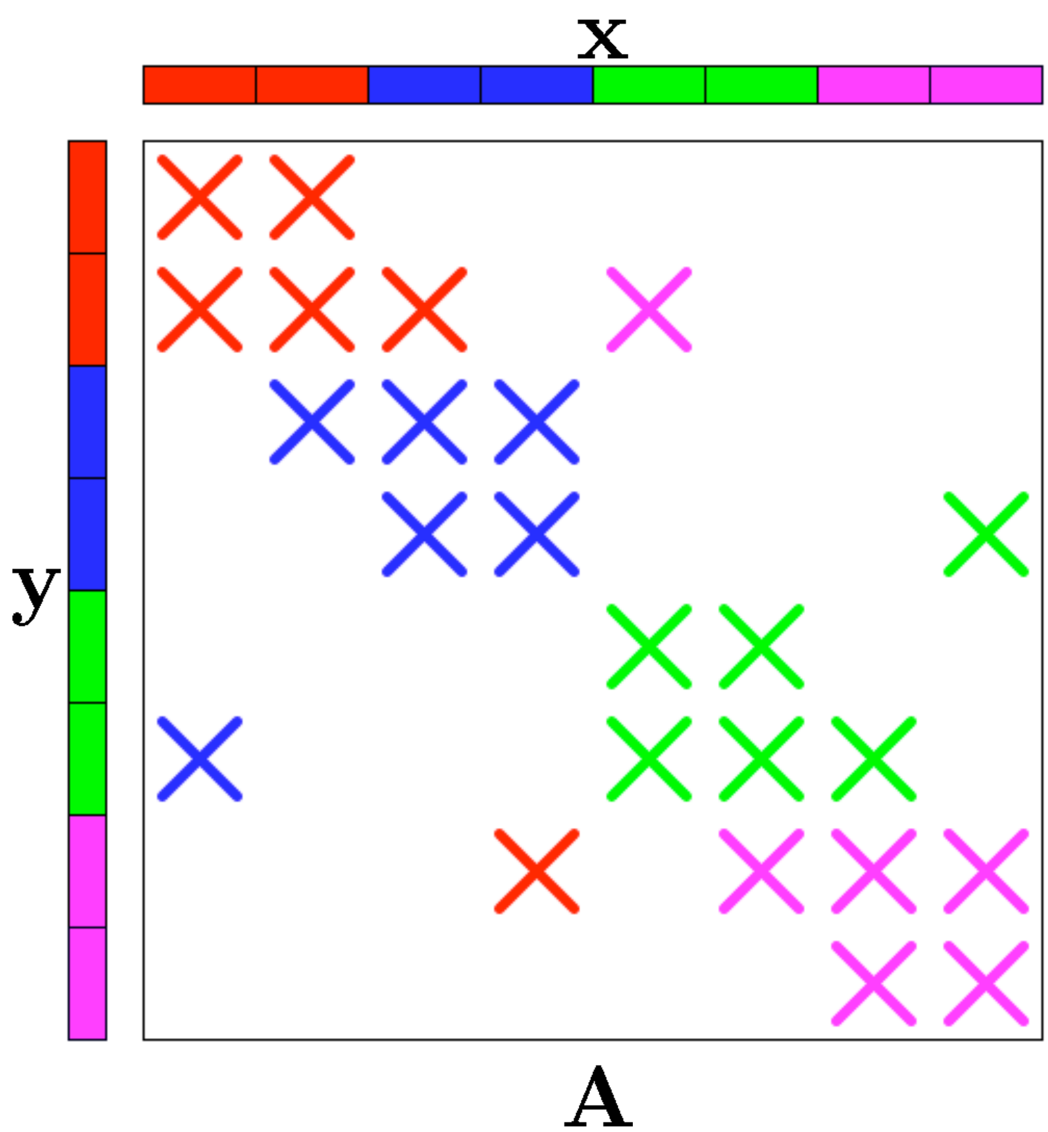
$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

- Vectors partitioned identically

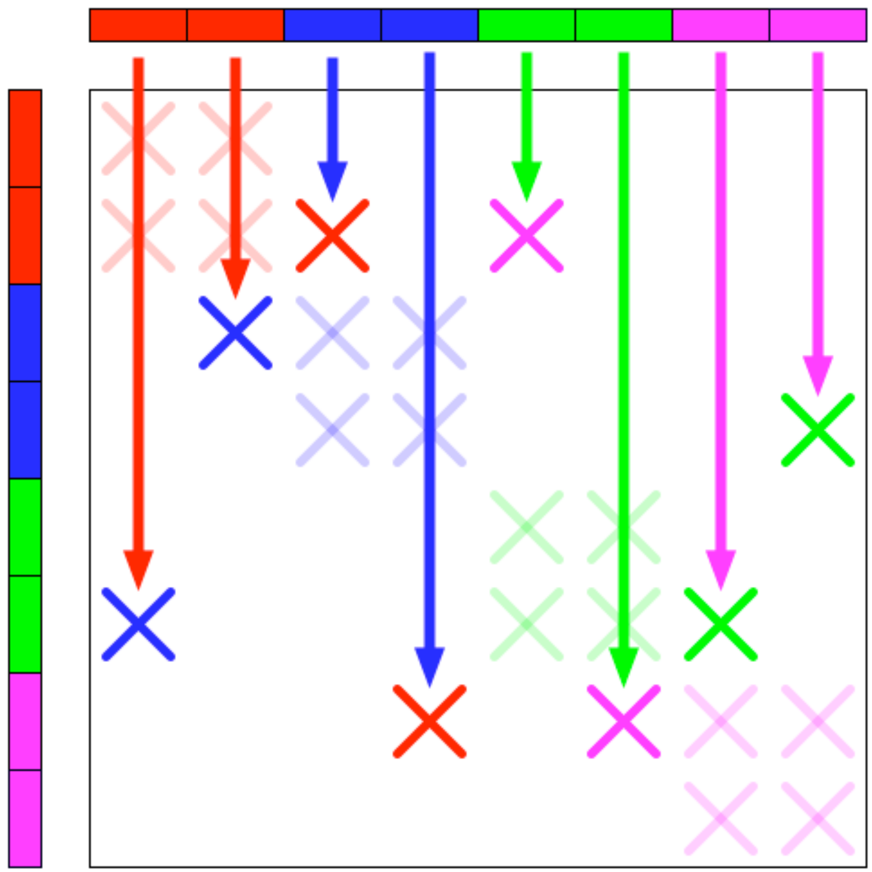
Objective

- Ideally we minimize total run-time
- Settle for easier objective
  - Work balanced
  - Minimize total communication volume
- Can partition matrices in different ways
  - 1-D
  - 2-D
- Can model communication in different ways
  - Graph
  - Bipartite graph
  - Hypergraph

Parallel Matrix-Vector Multiplication



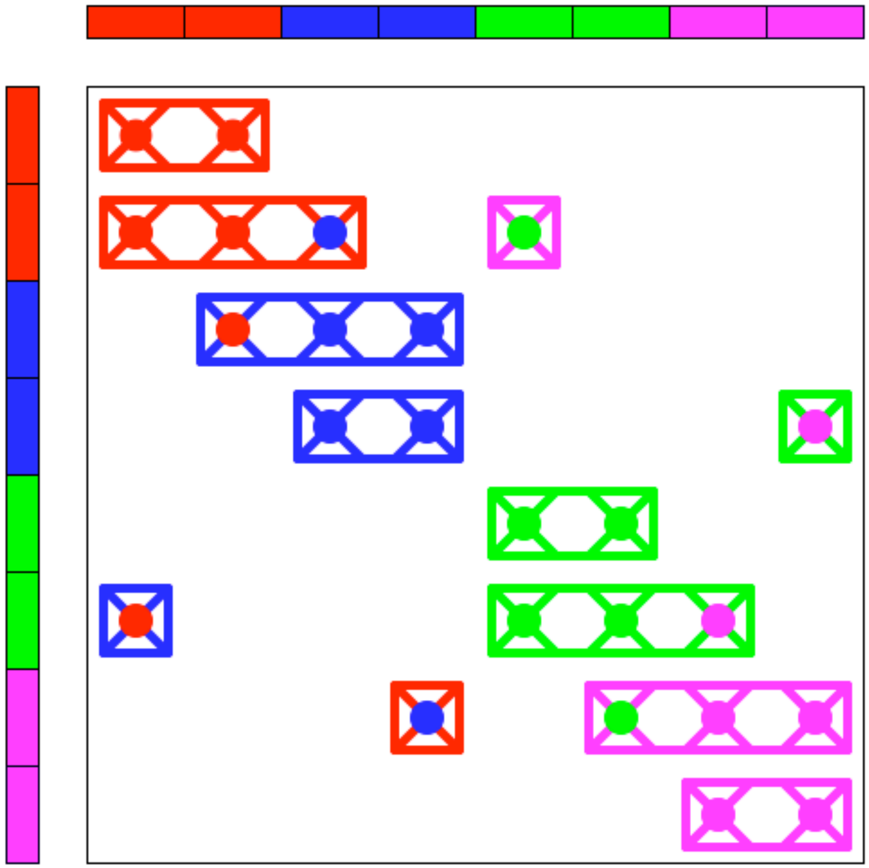
Parallel Matrix-Vector Multiplication Stage 1



"fan-out"

- x_j sent to remote processes that have nonzeros in column j

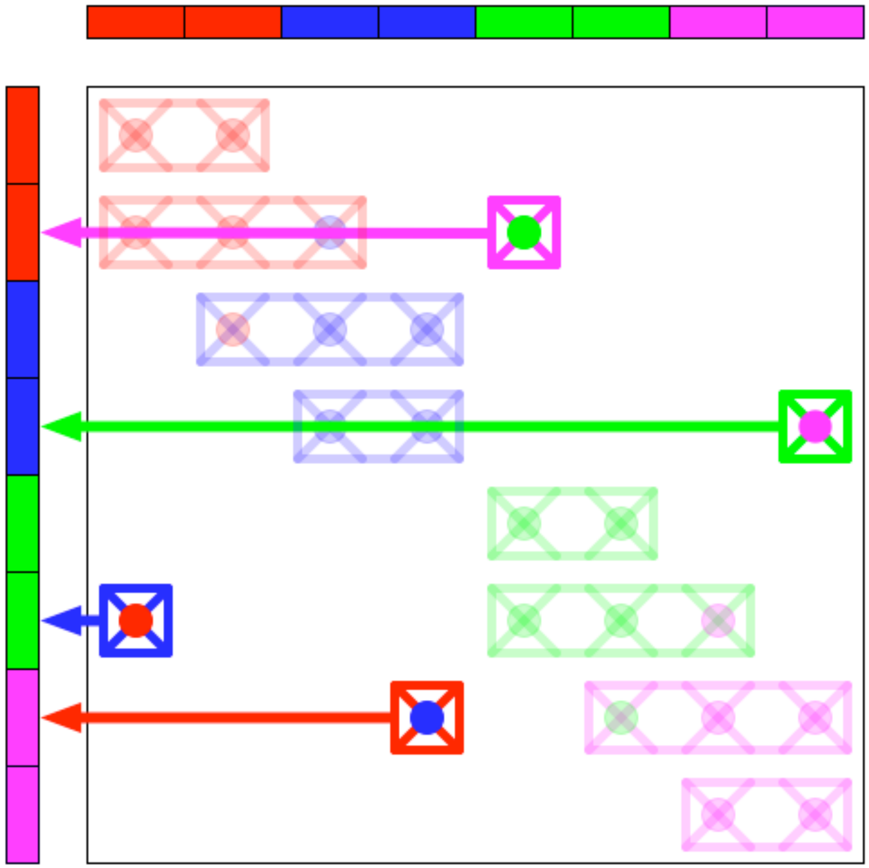
Parallel Matrix-Vector Multiplication Stage 2



$$y_i = \sum a_{ij} x_j, \quad \forall i,j : a_{ij} \neq 0$$

- Local partial inner-products

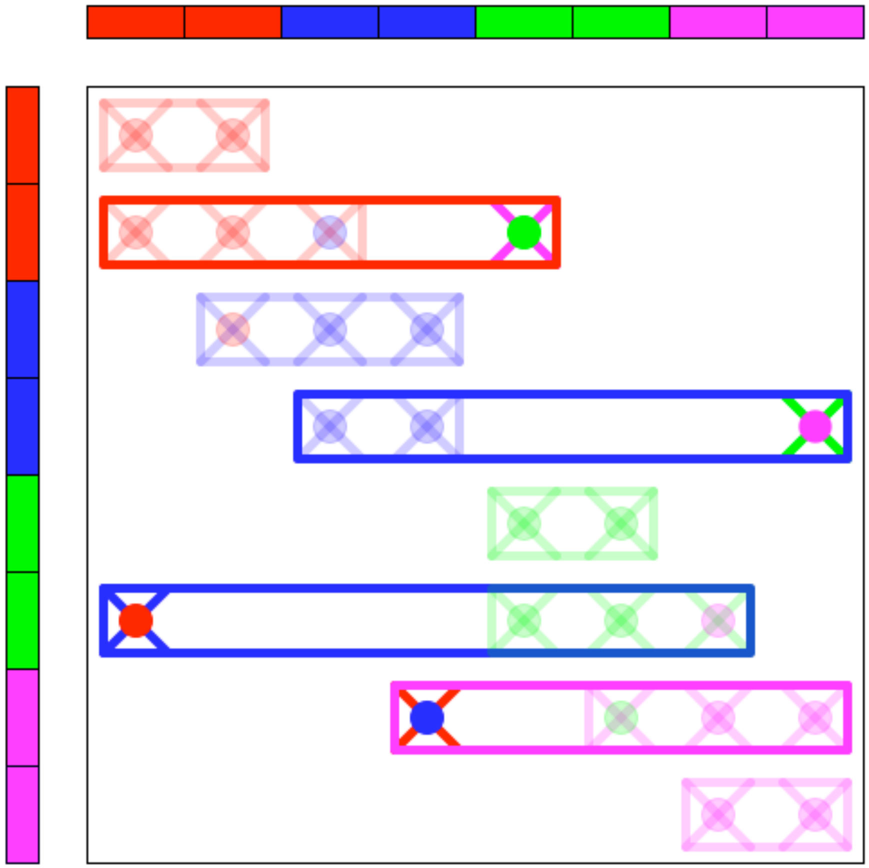
Parallel Matrix-Vector Multiplication Stage 3



"fan-in"

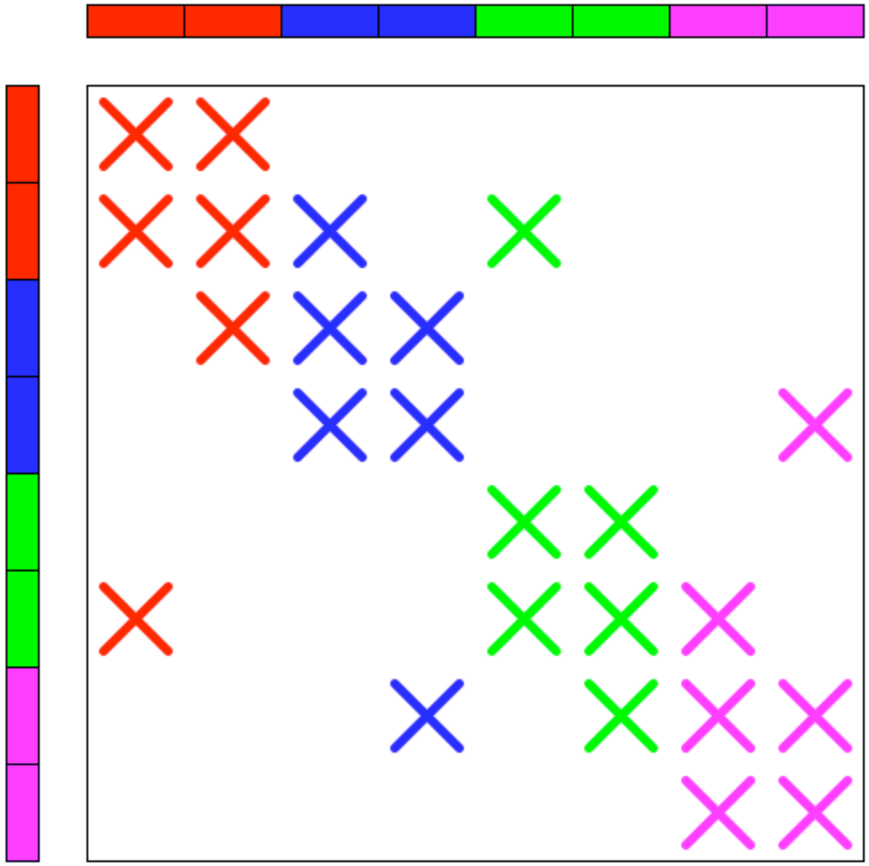
- Send partial inner-products to process that owns corresponding vector element y_i

Parallel Matrix-Vector Multiplication Stage 4



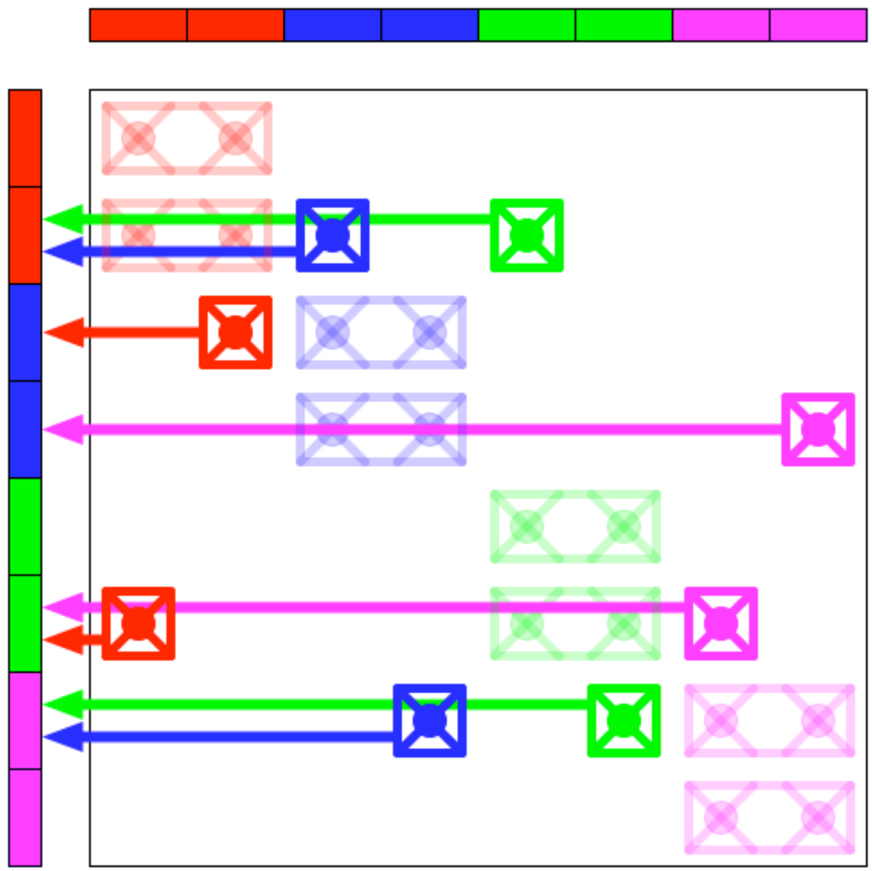
- Accumulate partial inner-products to obtain complete resulting vector

1-D Column Partitioning



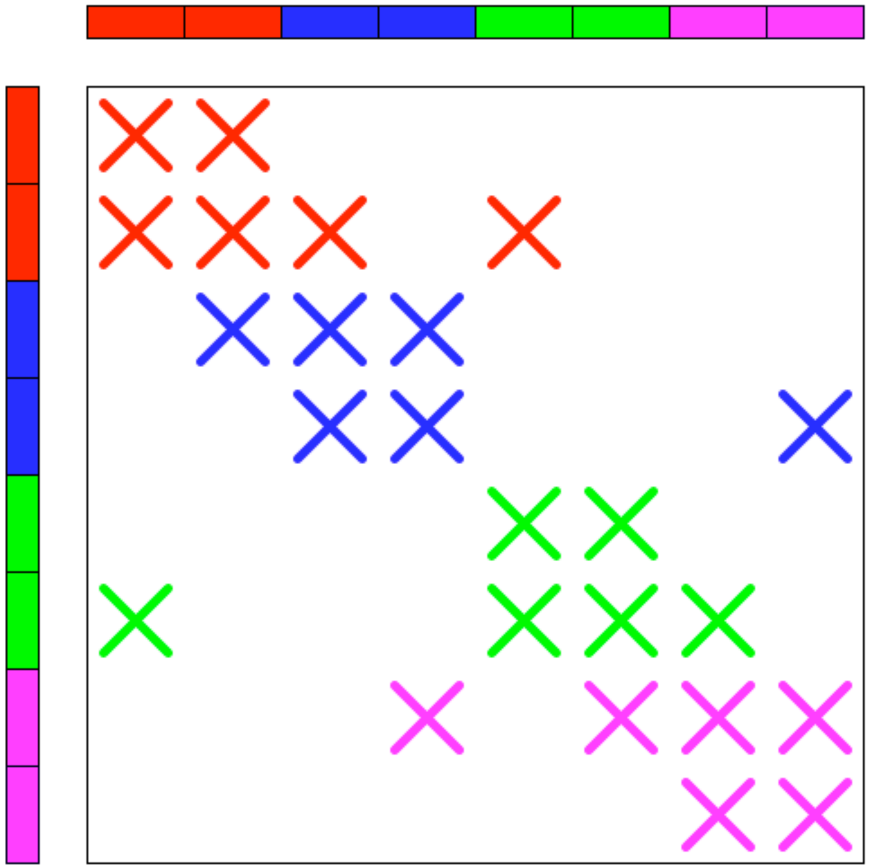
- Each process assigned nonzeros for set of columns

1-D Column Partitioning



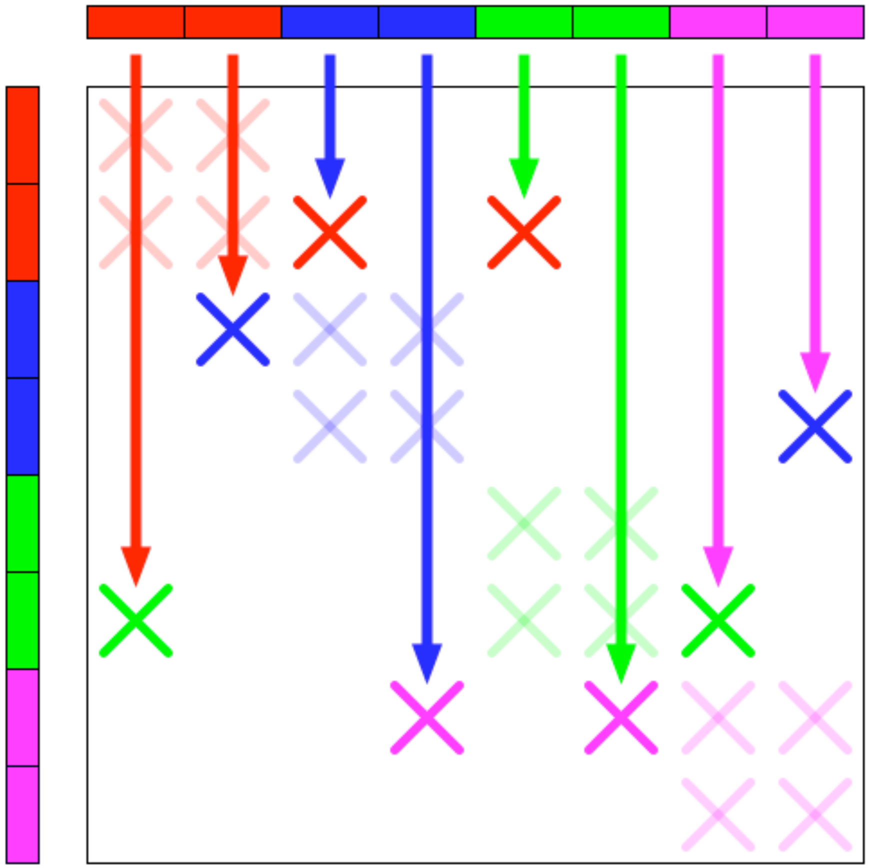
- Only “fan-in” communication stage necessary

1-D Row Partitioning



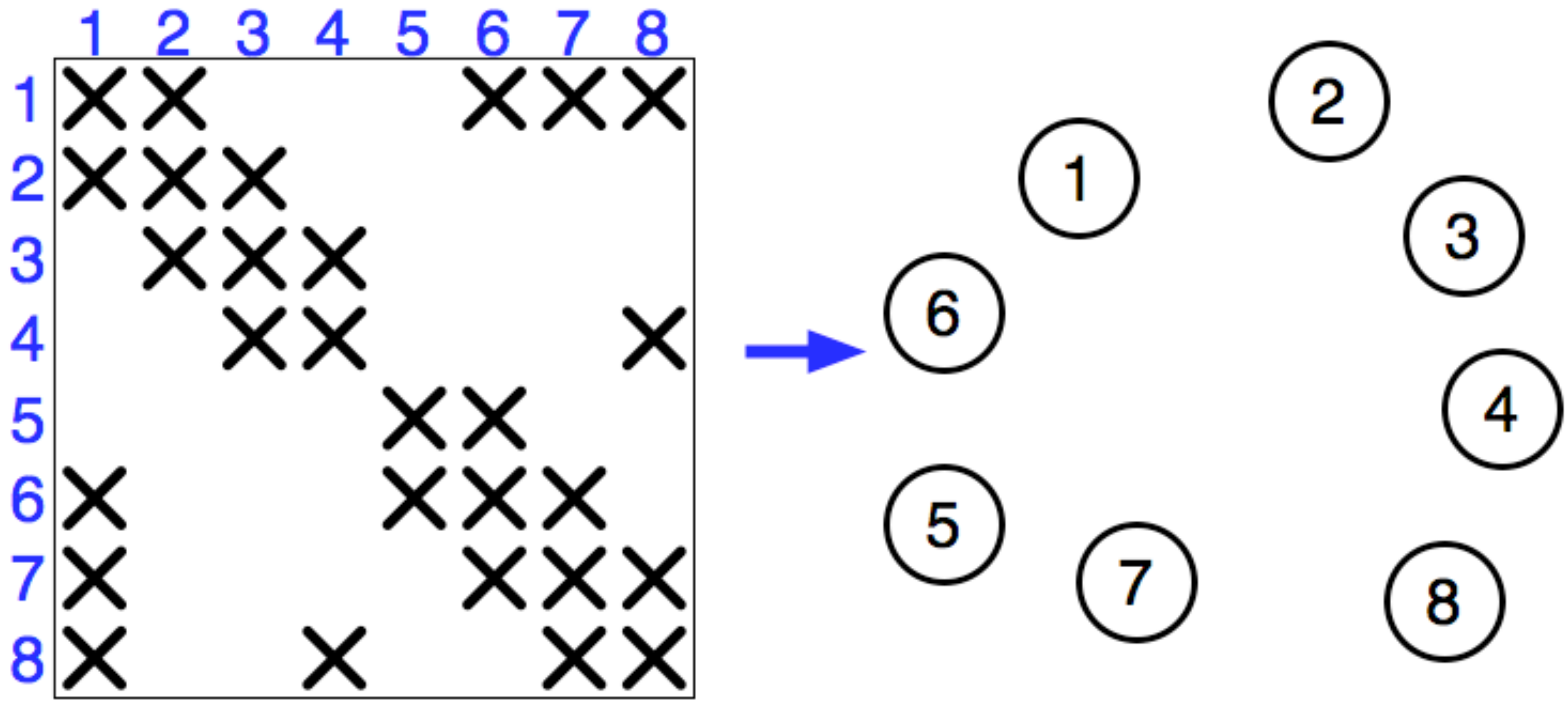
- Each process assigned nonzeros for set of rows

1-D Row Partitioning



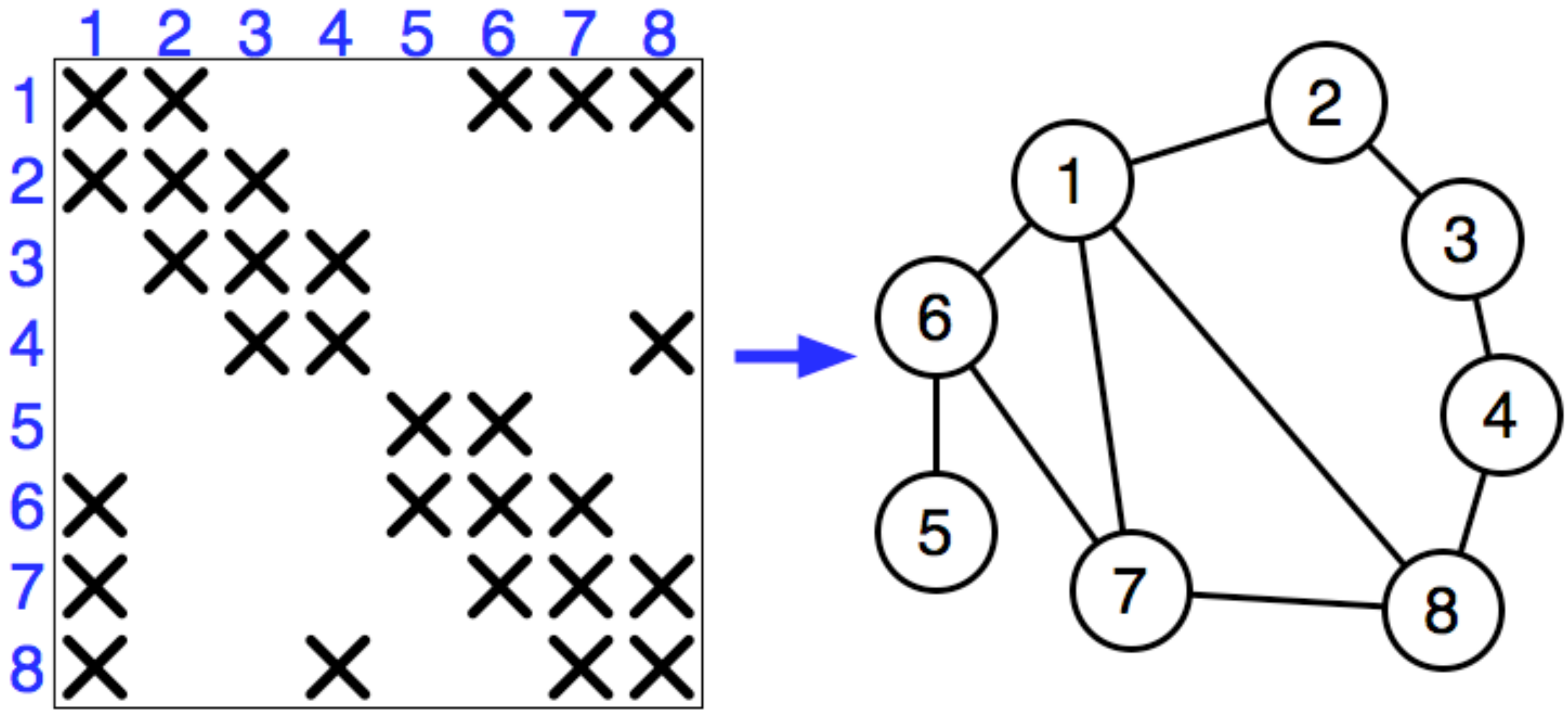
- Only "fan-out" communication stage necessary

Graph Model of 1-D Partitioning



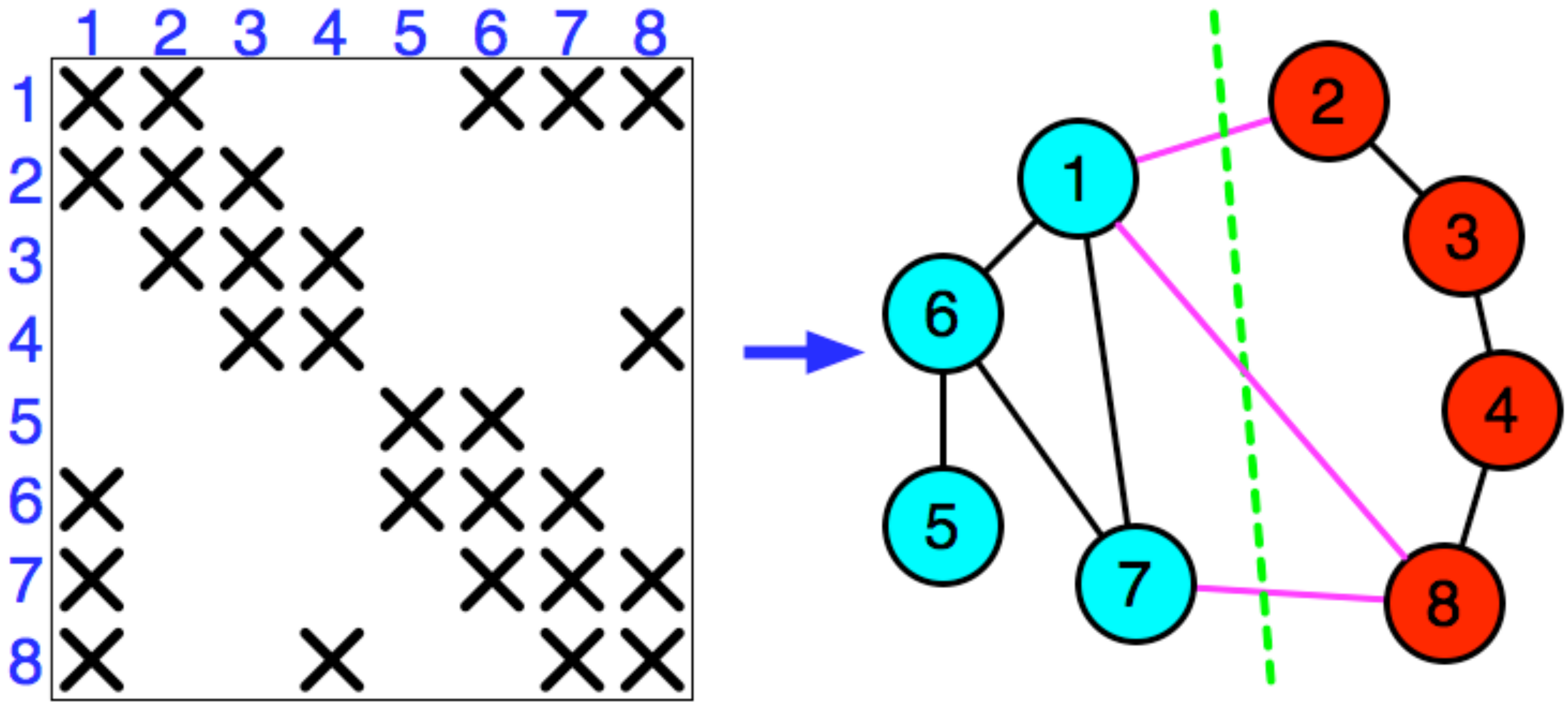
- Each row or column represented by graph vertex
  - Weighted by number of nonzeros in row/column

Graph Model of 1-D Partitioning



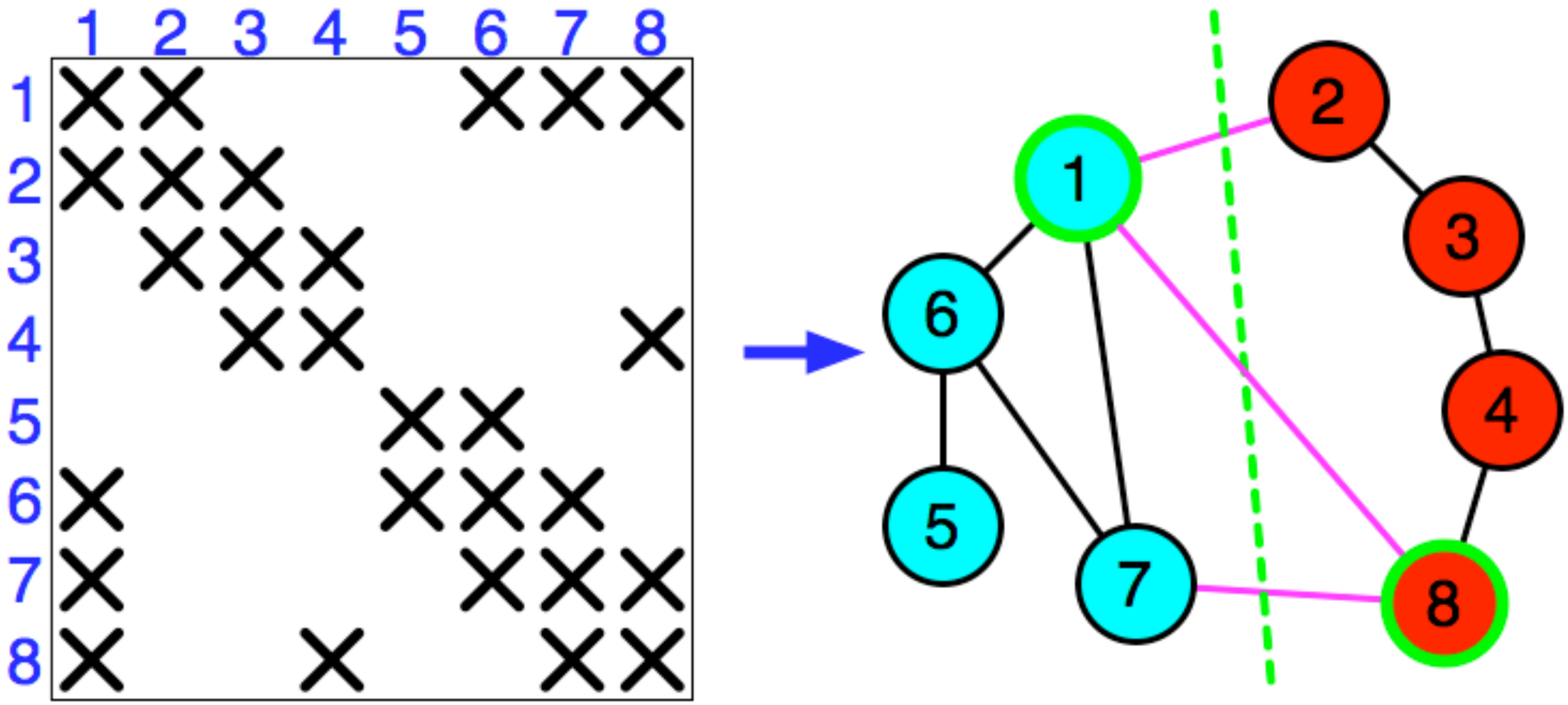
- Nonzeros represented by edges between 2 vertices (corresponding to nonzero row, col)

Graph Model of 1-D Partitioning



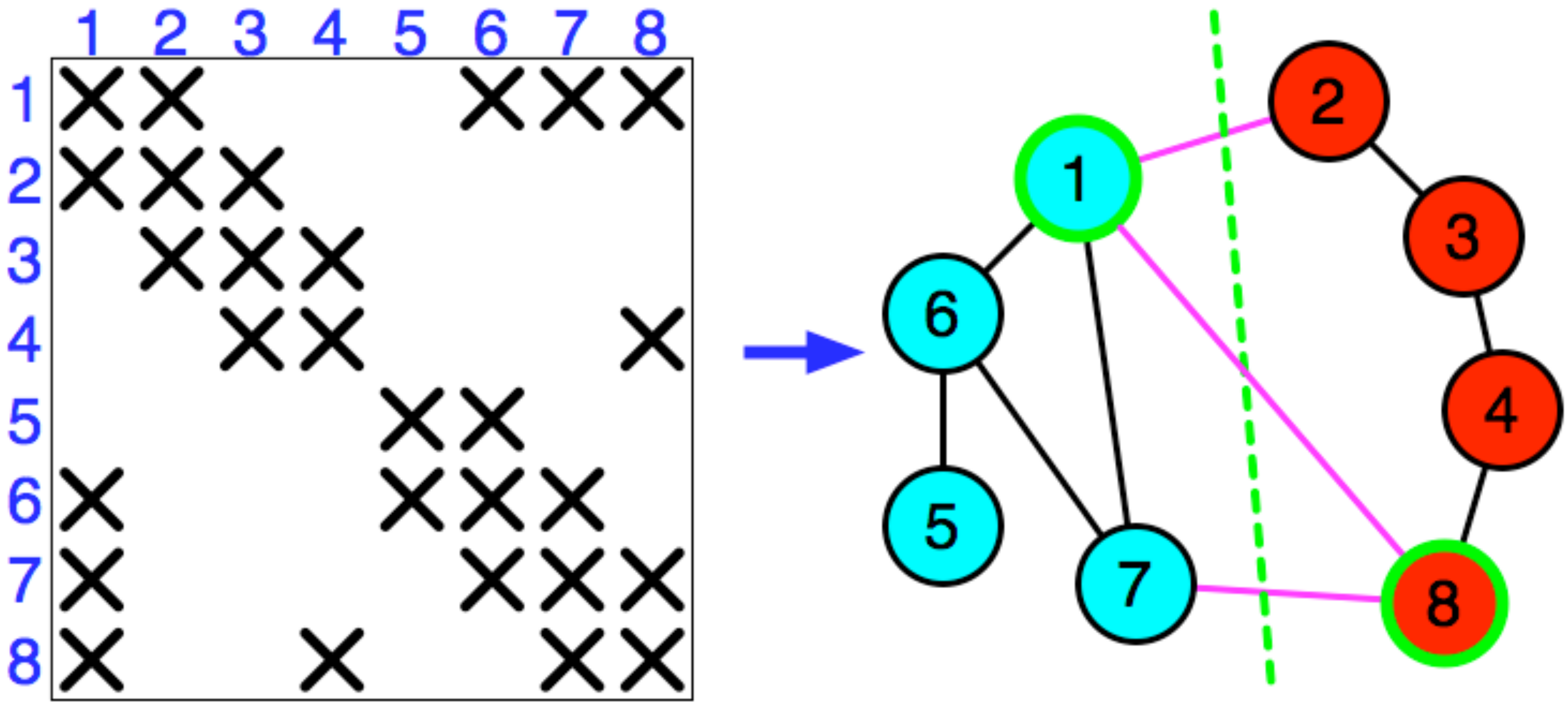
- Partition into k equal sets
  - Such that number of cut edges is minimized

Graph Model Shortcomings



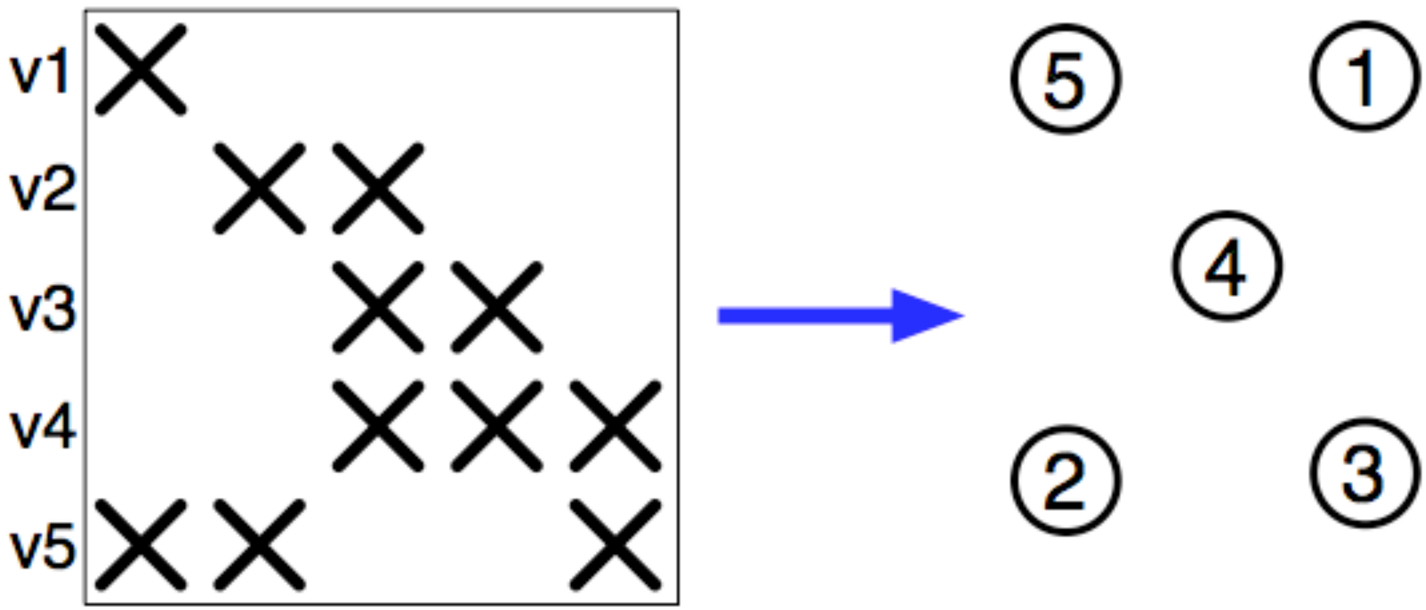
- Inaccurate approximation of communication volume
  - Approximate volume: 6
  - Actual volume: 4

Graph Model Shortcomings



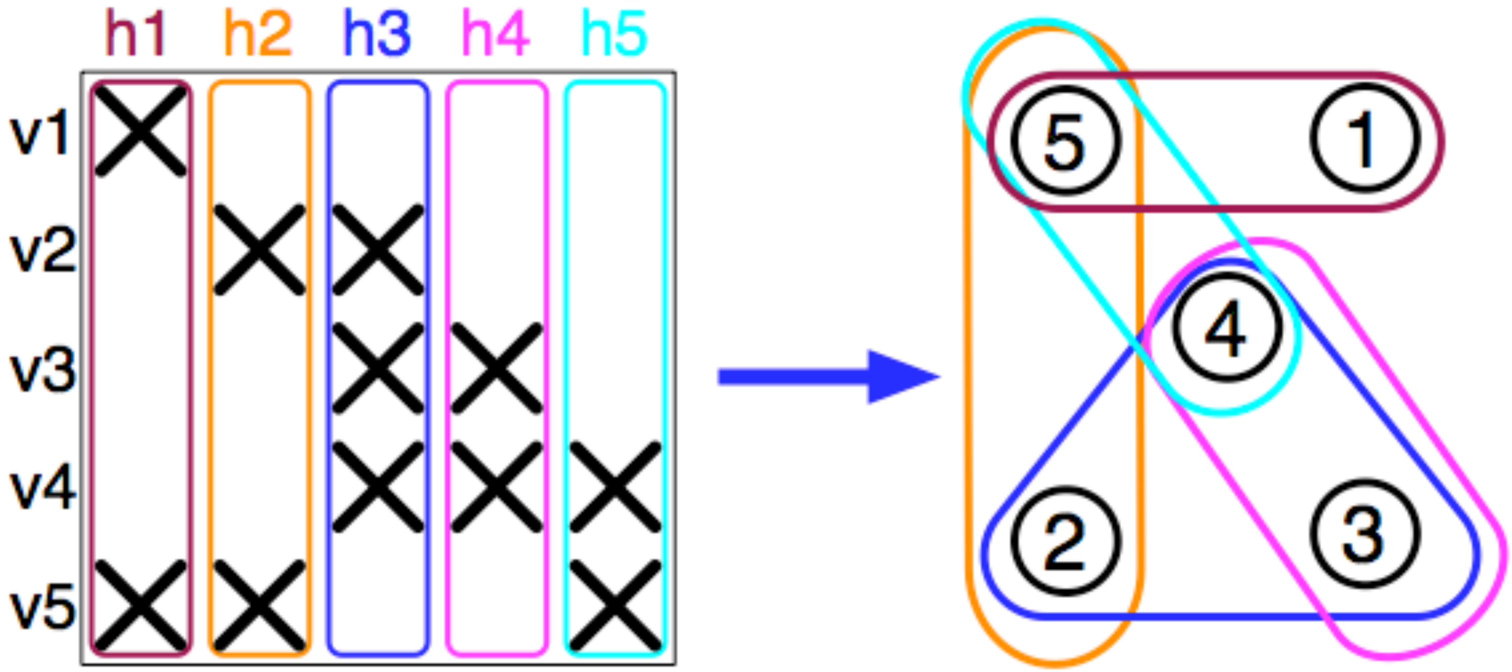
- Requires symmetric nonzero pattern
- NP-hard to solve optimally

Hypergraph Model of 1-D (Row) Partitioning



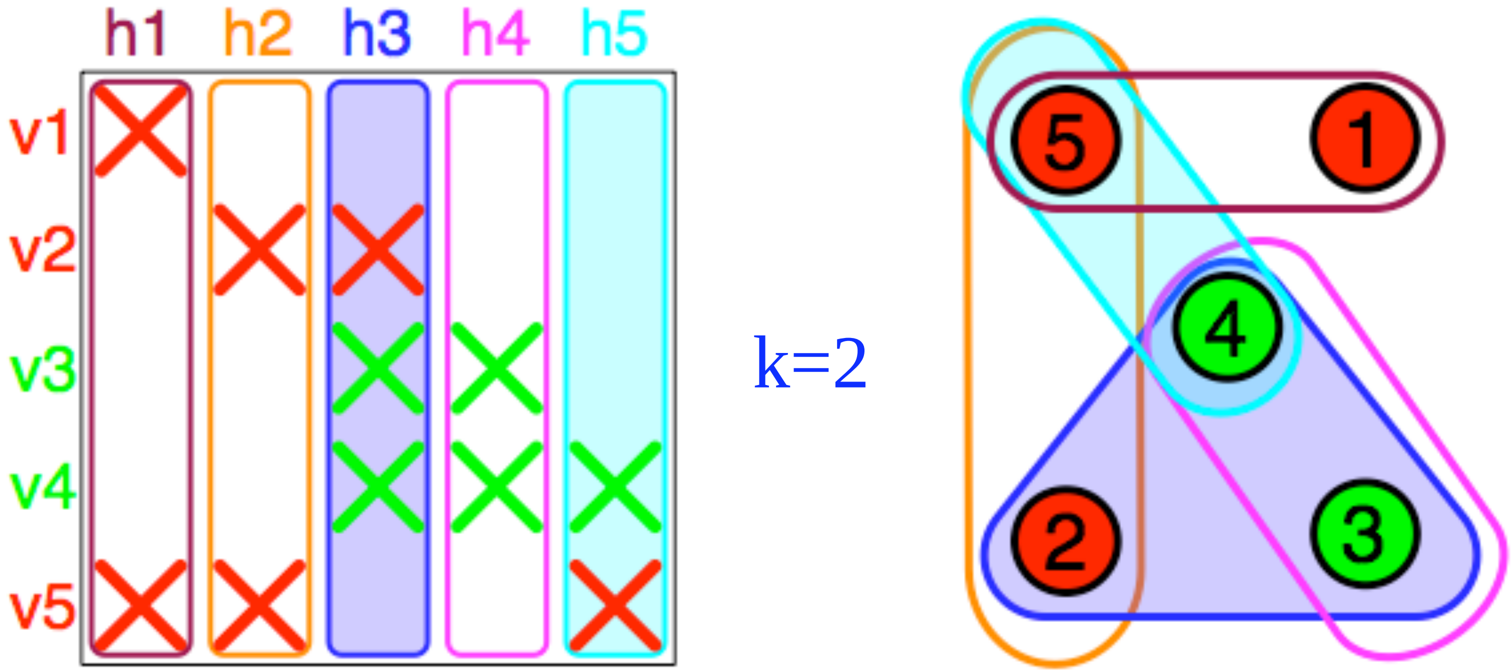
- Nonzero pattern can be unsymmetric
- Rows represented by vertices in hypergraph
  - Weighted by number of nonzeros in row

Hypergraph Model of 1-D (Row) Partitioning



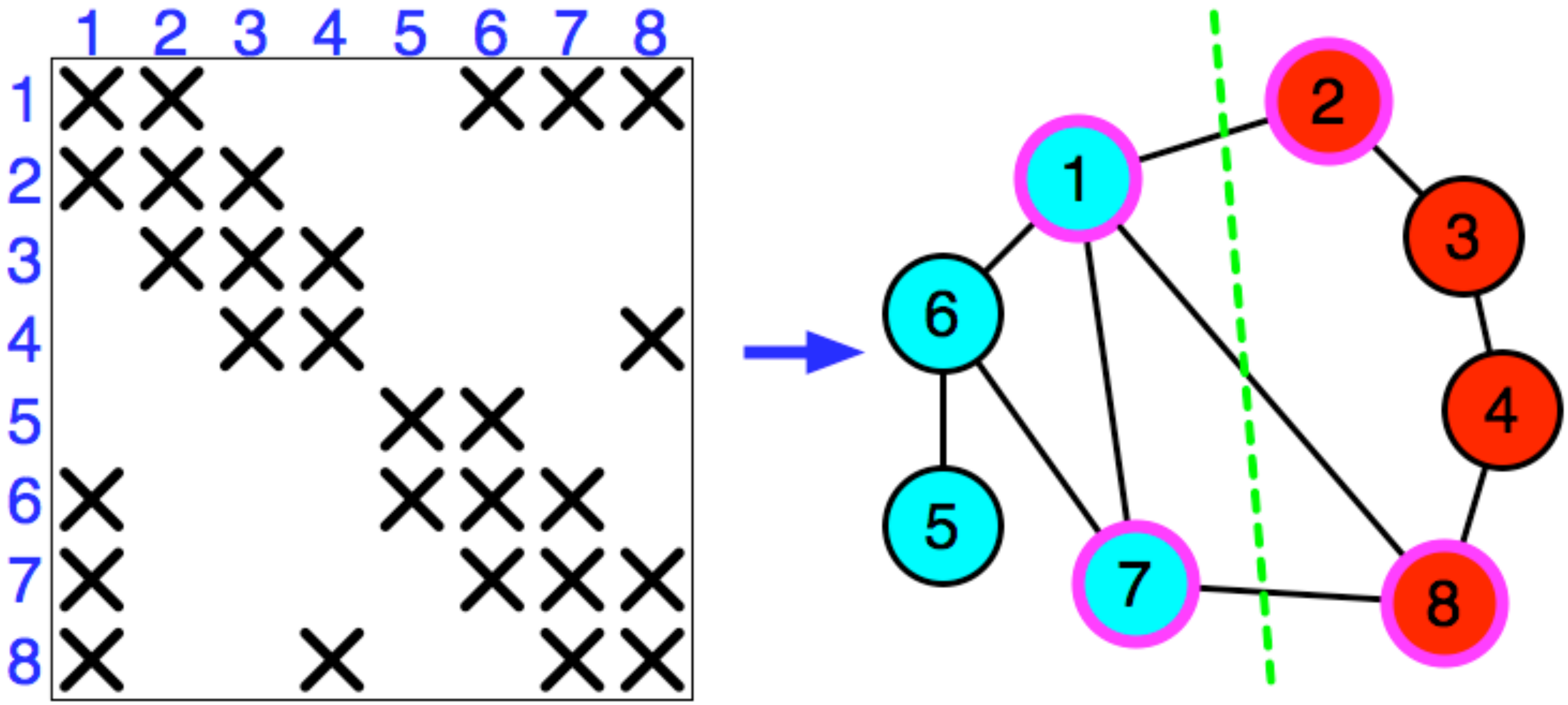
- Columns represented by hyperedges in hypergraph

Hypergraph Model of 1-D (Row) Partitioning



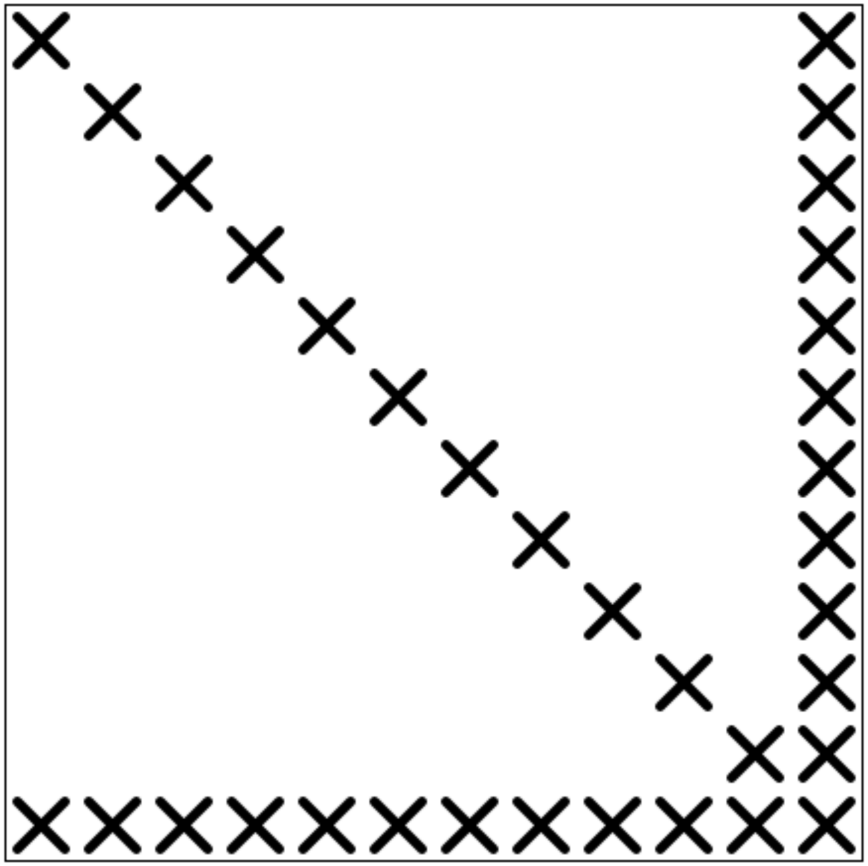
- Partition vertices into k equal sets
- Hyperedge cut = communication volume
  - Aykanat and Catalyurek (1996)
- NP-hard to solve optimally

Graph Model Revisited



- Bisection: count boundary vertices
- Slightly more complicated for k>2

When 1-D Partitioning is Inadequate

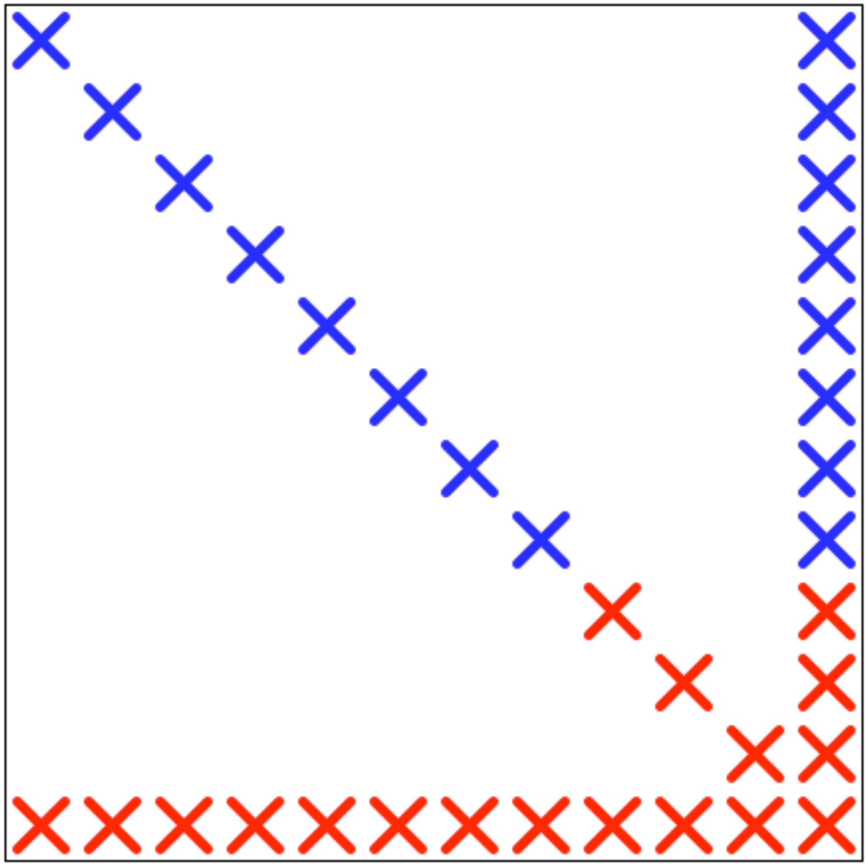


"Arrowhead" matrix

$n=12$

$nnz=30$

When 1-D Partitioning is Inadequate



n=12

nnz=30

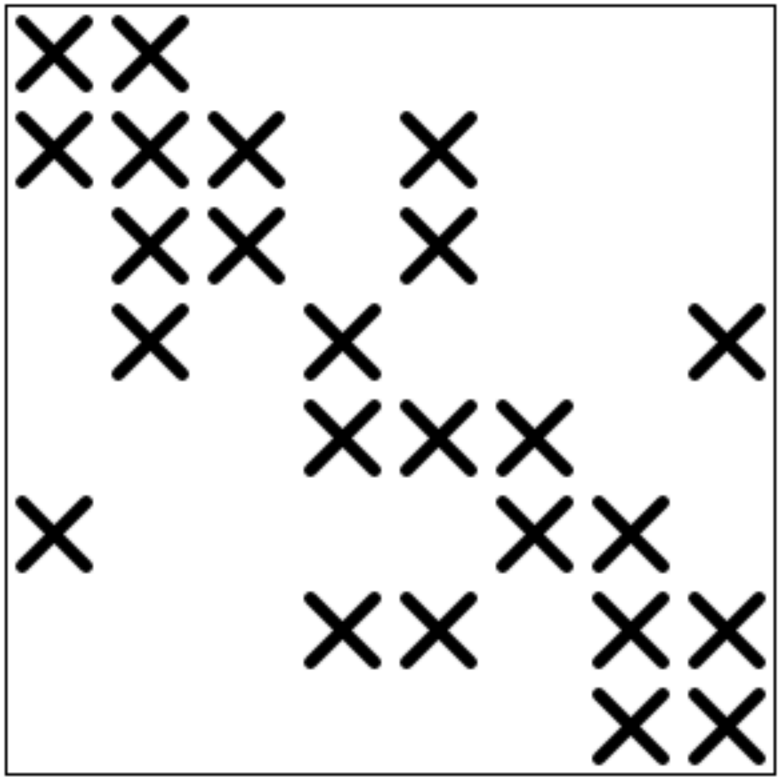
volume = 9

- For nxn matrix for any 1-D bisection:
  - nnz = 3n-2
  - Volume ≈ 3/4*n

2-D Partitioning Methods

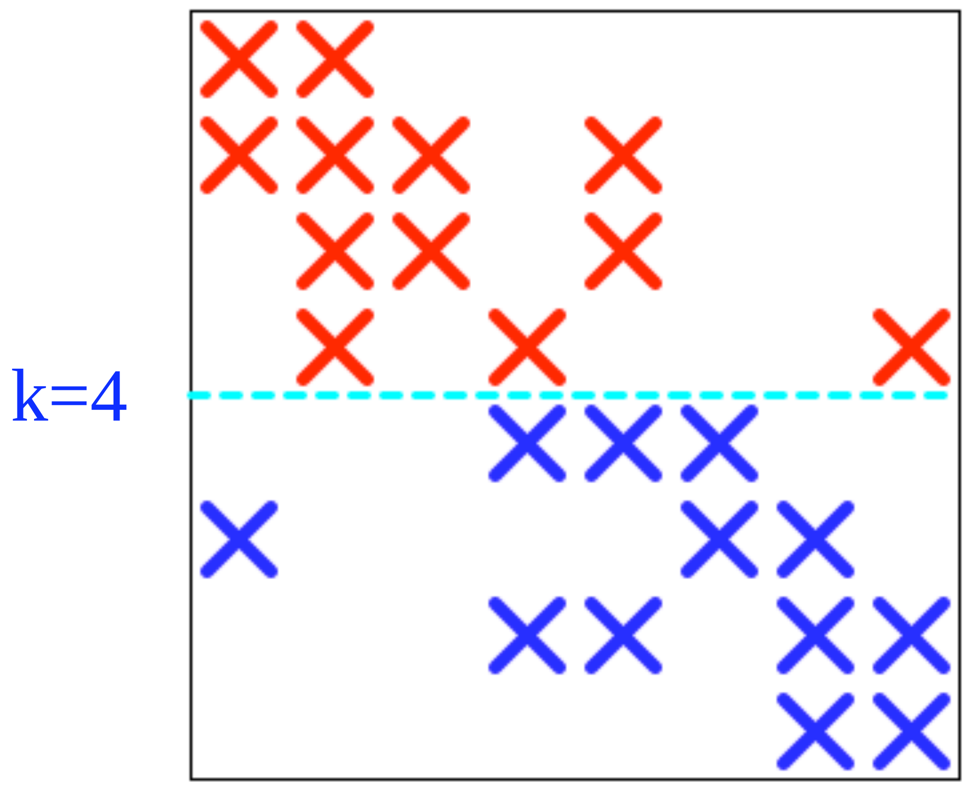
- More flexibility
- Yield lower communication volume for many problems

2-D Partitioning Methods: Cartesian



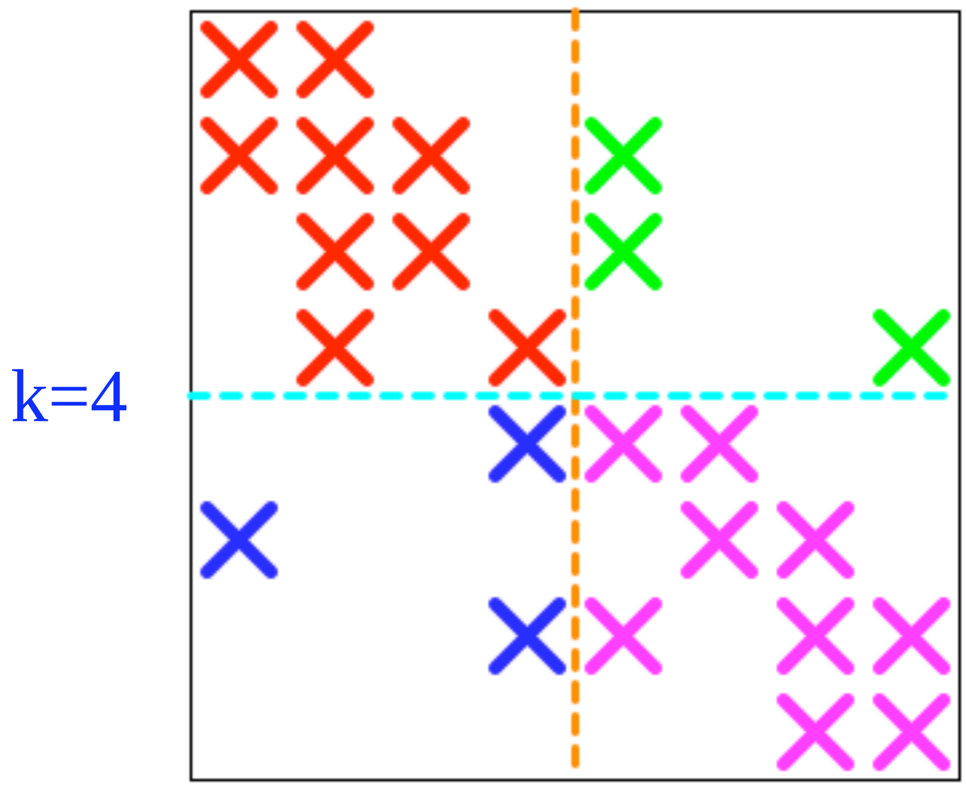
- Different variations
- Two-stage partitioning of rows and columns with 1D hypergraph partitioning

2-D Partitioning Methods: Cartesian



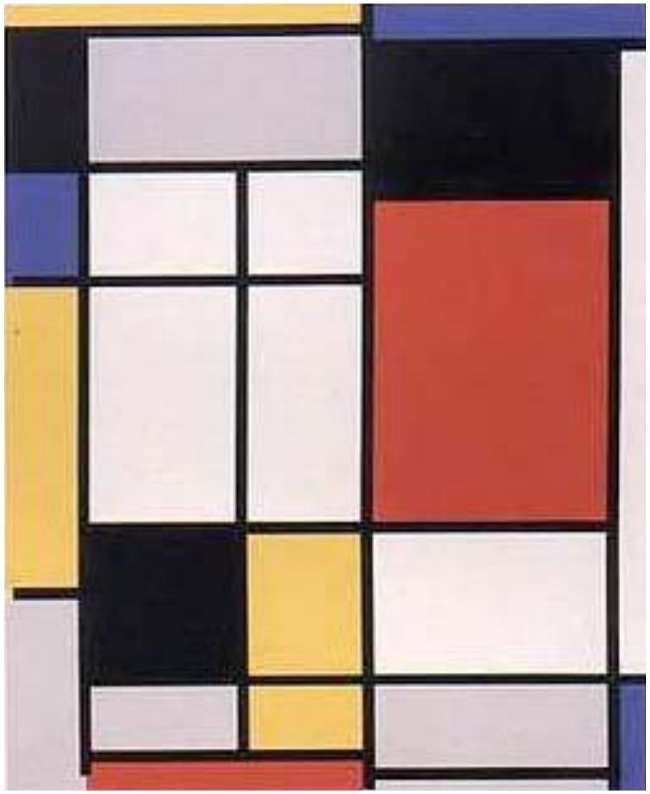
- Block version shown for clarity
- Stage 1: partition rows

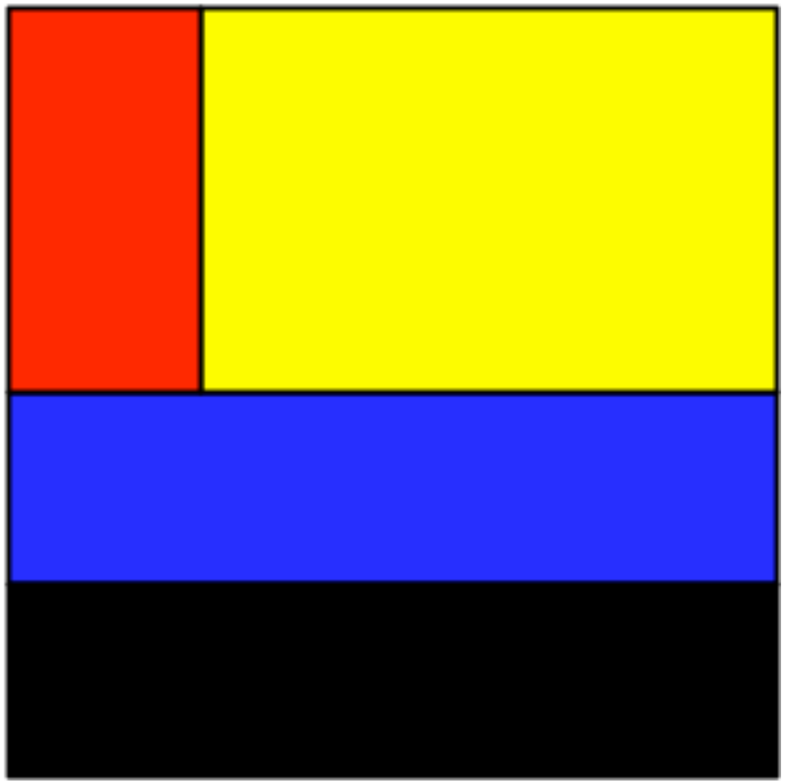
2-D Partitioning Methods: Cartesian



- Stage 2: partition columns
- Load imbalance

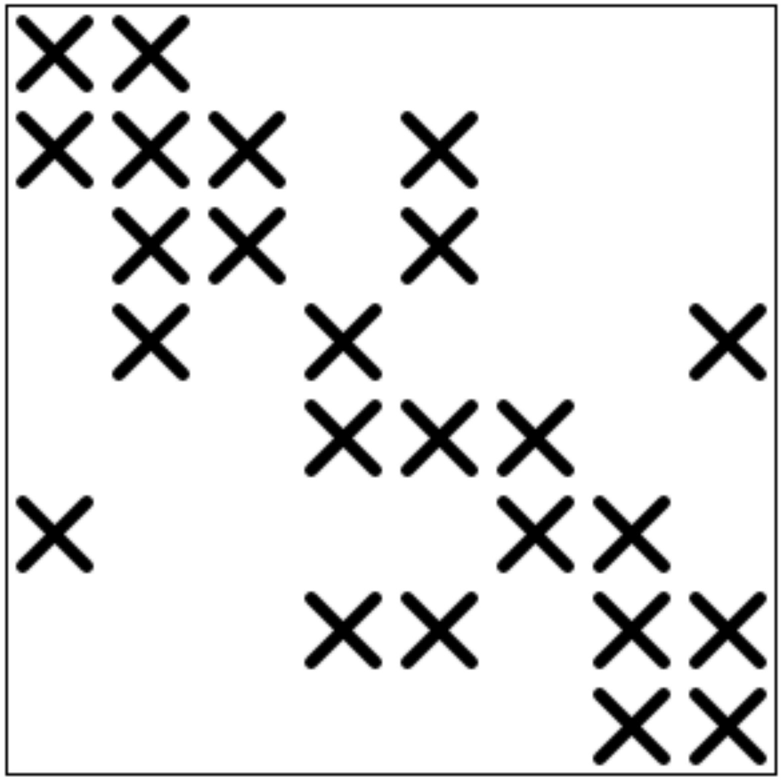
2-D Partitioning Methods: Mondriaan





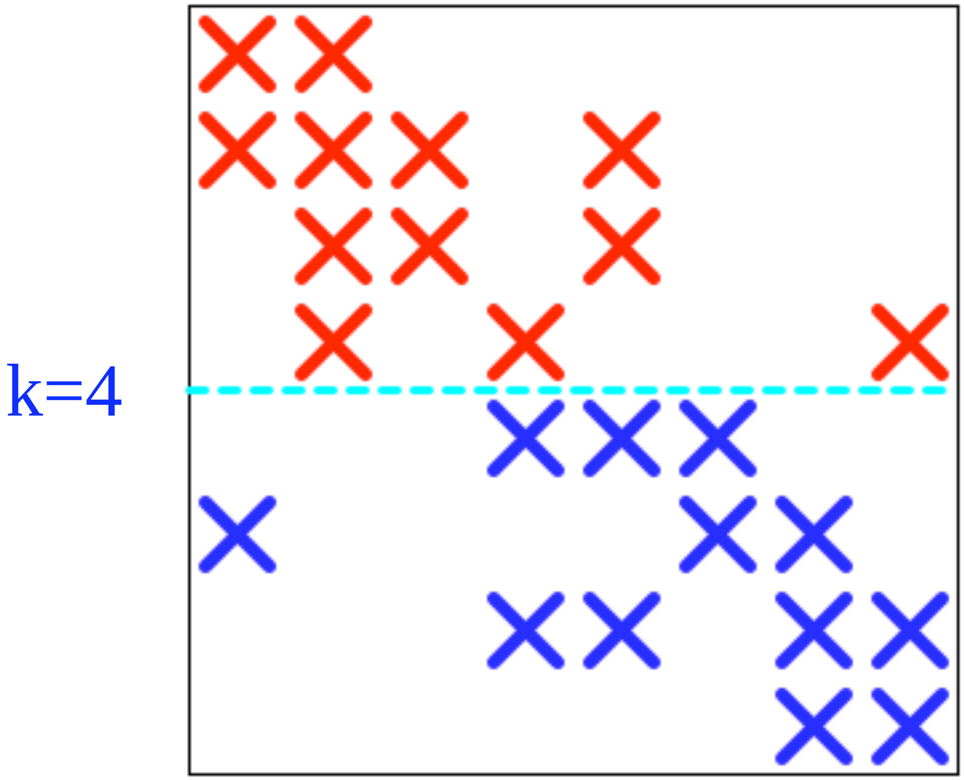
- Piet Mondria(a)n
  - –Dutch painter (1872-1944)
  - –Colored rectangles
  - –Black rectilinear lines
- 2D Mondriaan Method
  - –Bisseling, Vastenhouw
  - –Irregular rectangle partitions

2-D Partitioning Methods: Mondriaan



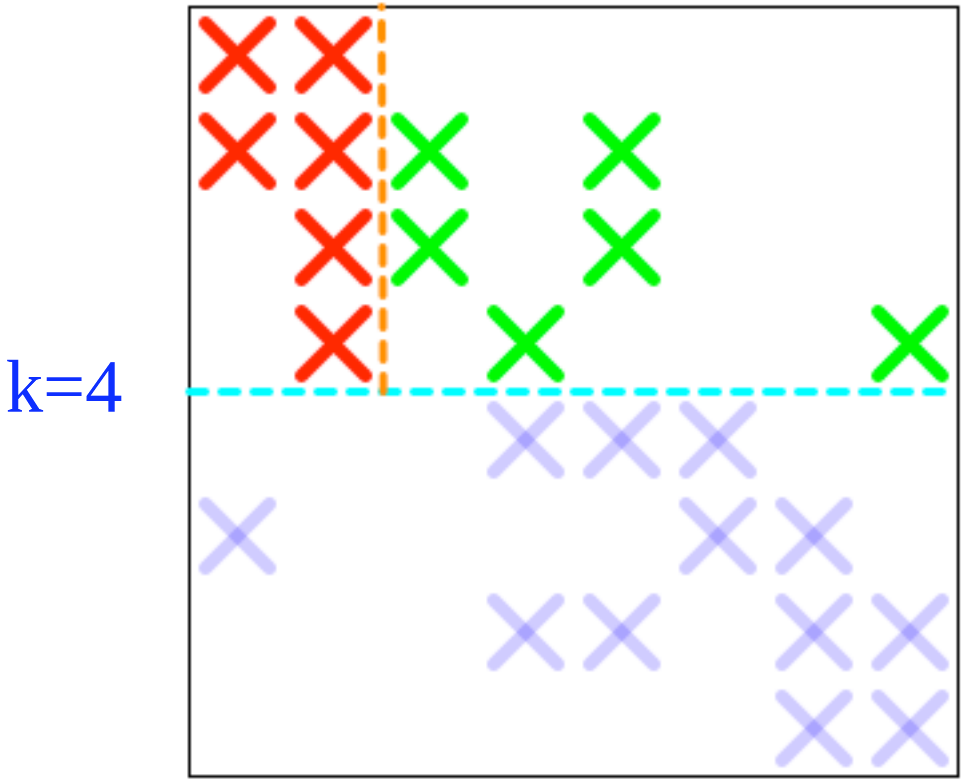
- Recursive bisection hypergraph partitioning
- Each level: 1D row or column partitioning

2-D Partitioning Methods: Mondriaan



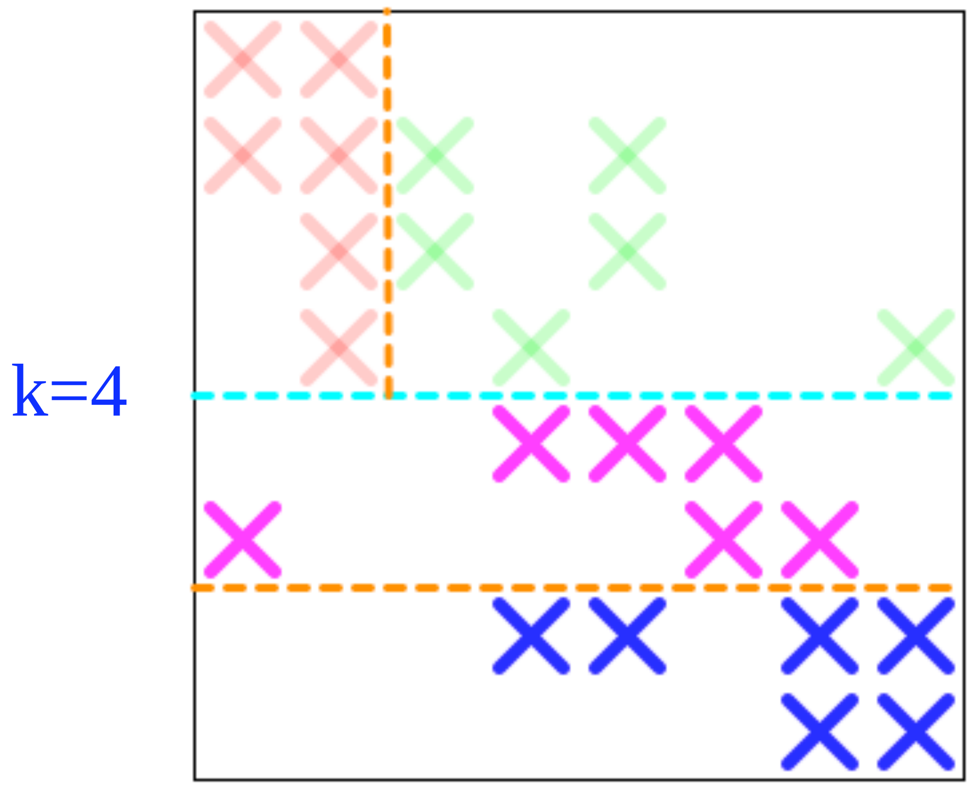
- Block version shown for clarity
- Level 1-- entire matrix
- Row partitioning (cut: 4 vs. 5)

2-D Partitioning Methods: Mondriaan



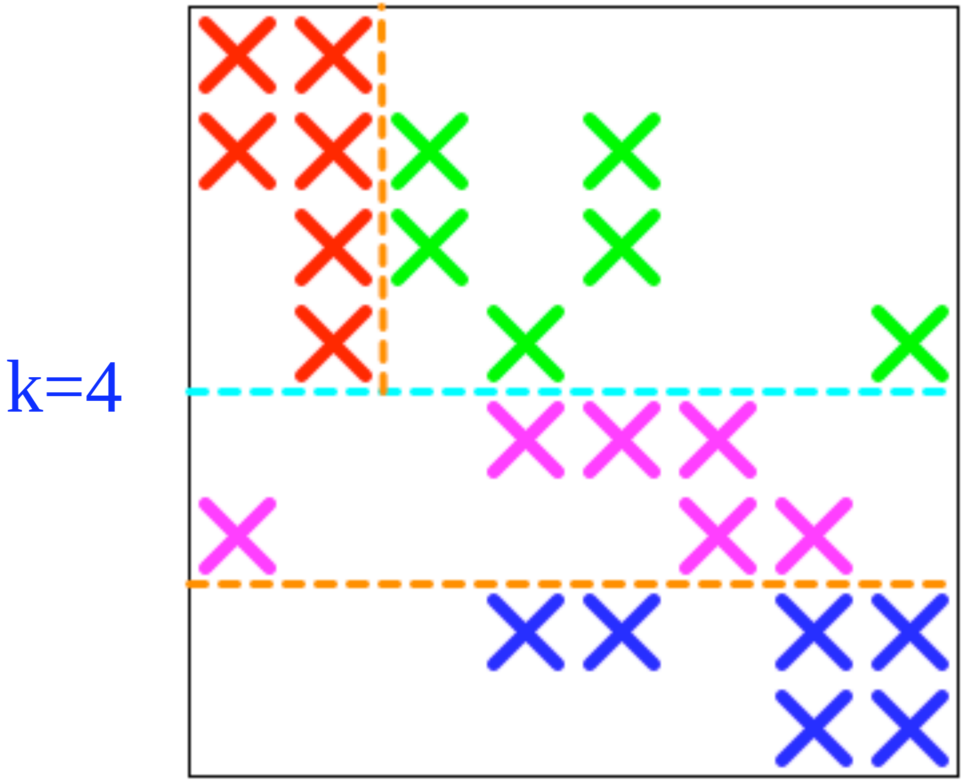
- Level 2 -- upper partition
- Column partitioning

2-D Partitioning Methods: Mondriaan



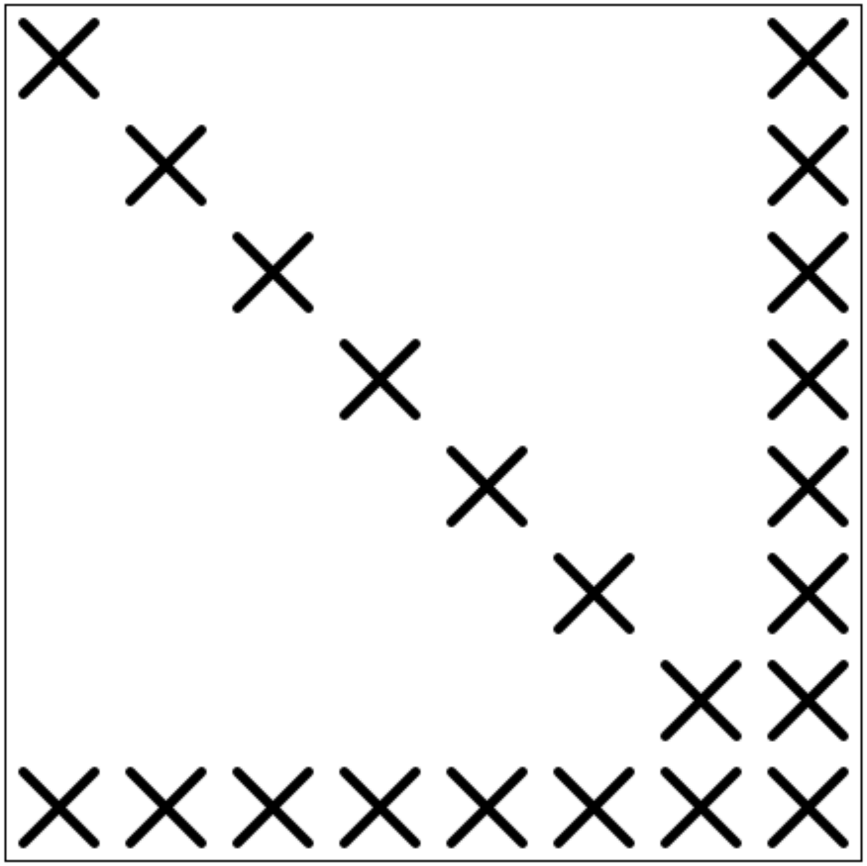
- Level 2 -- lower partition
- Row partitioning (balance)

2-D Partitioning Methods: Mondriaan



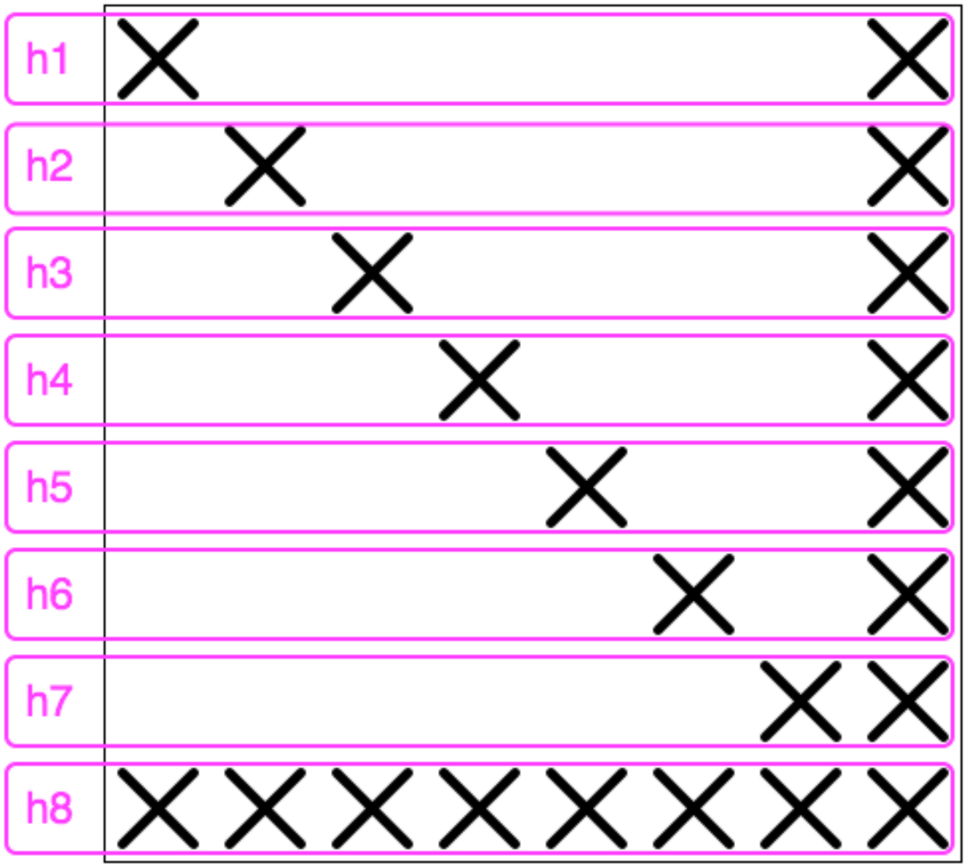
- Mondriaan
  - Fairly fast
  - Generally yields good partitions
  - Does not suffer from poor load-balancing

2-D Method: Fine-Grain Hypergraph Model



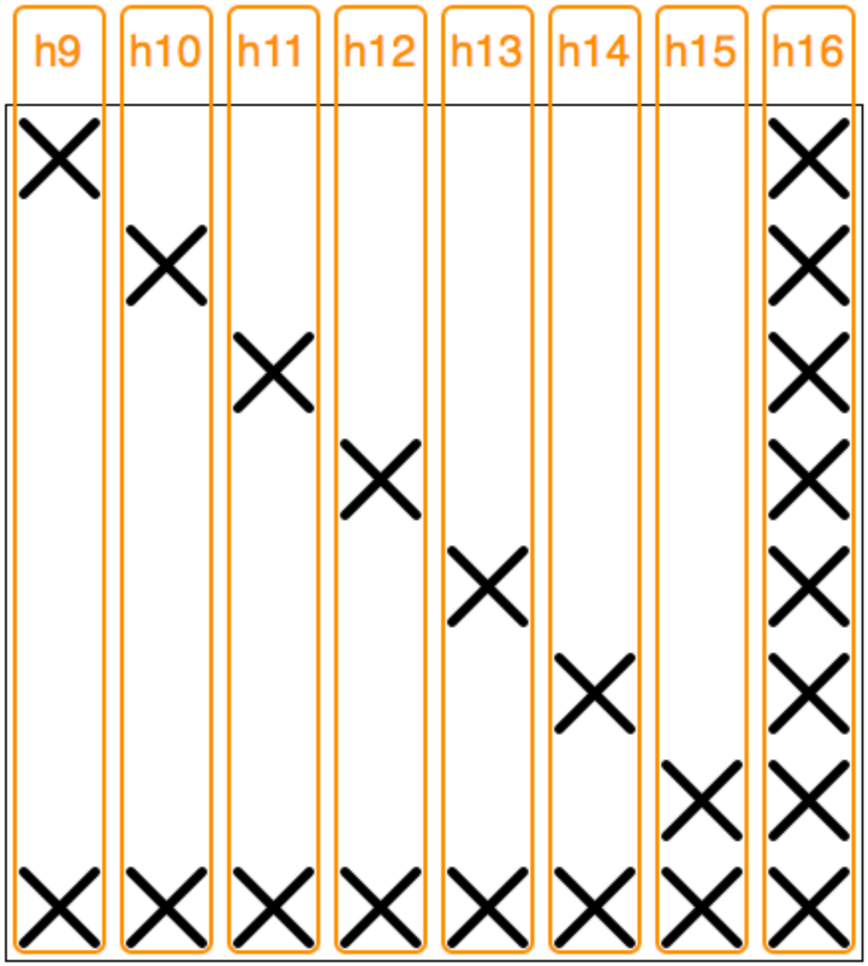
- Catalyurek and Aykanat (2001)
- Assign each nz separately
- Nonzeros represented by vertices in hypergraph

2-D Method: Fine-Grain Hypergraph Model



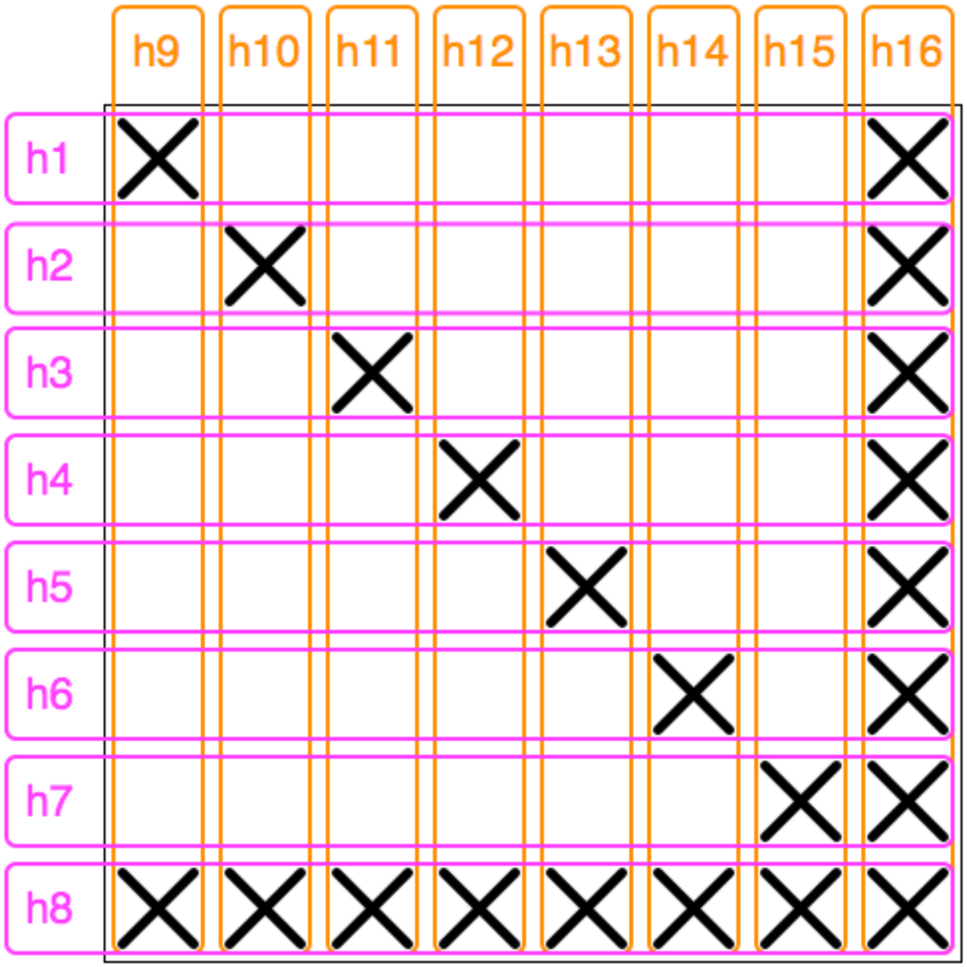
- Rows represented by hyperedges

2-D Method: Fine-Grain Hypergraph Model



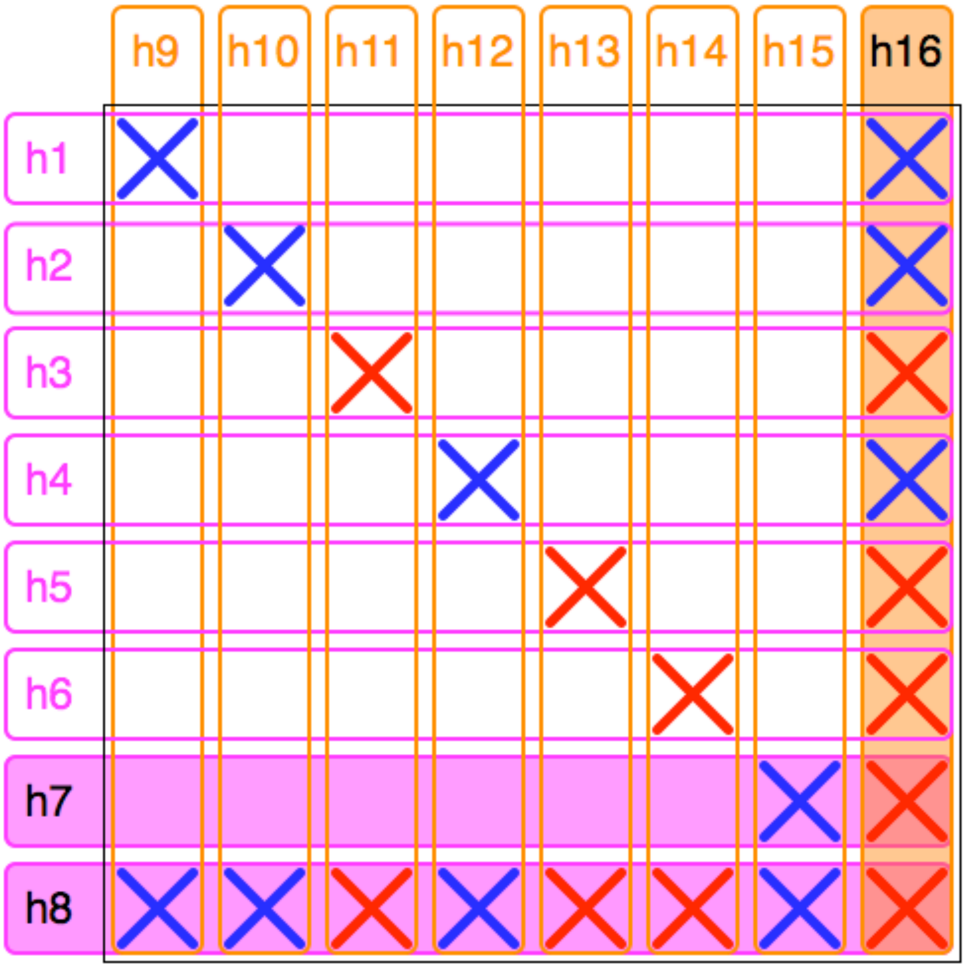
- Columns represented by hyperedges

2-D Method: Fine-Grain Hypergraph Model



- 2n hyperedges

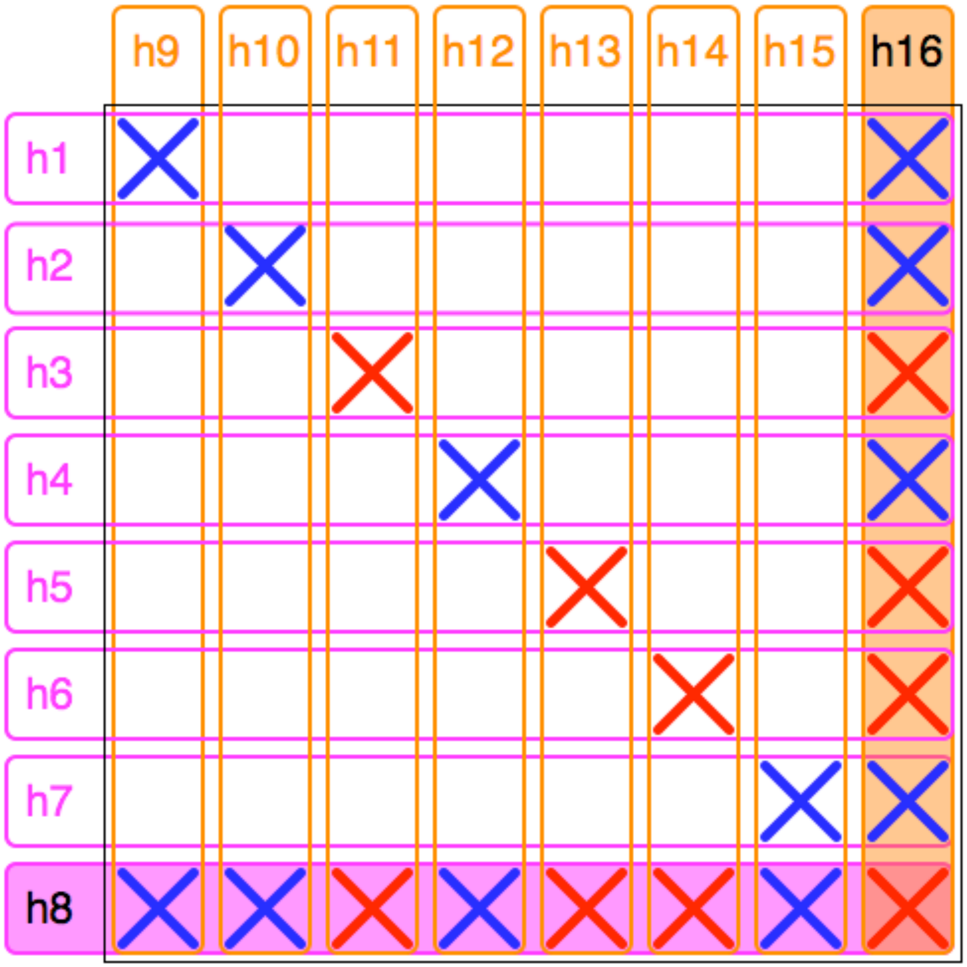
2-D Method: Fine-Grain Hypergraph Model



k=2, volume = 3

- Partition vertices into k equal sets
- Volume = hypergraph cut
- Minimum volume partitioning when optimally solved
- Larger NP-hard problem

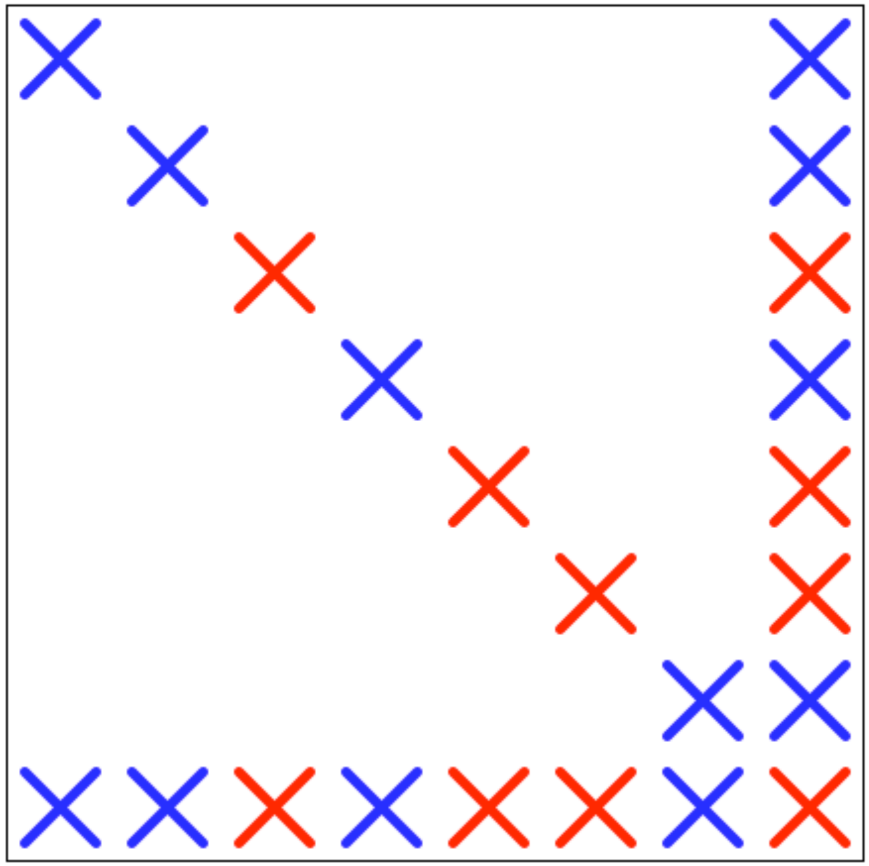
2-D Method: Fine-Grain Hypergraph Model



Volume = 2

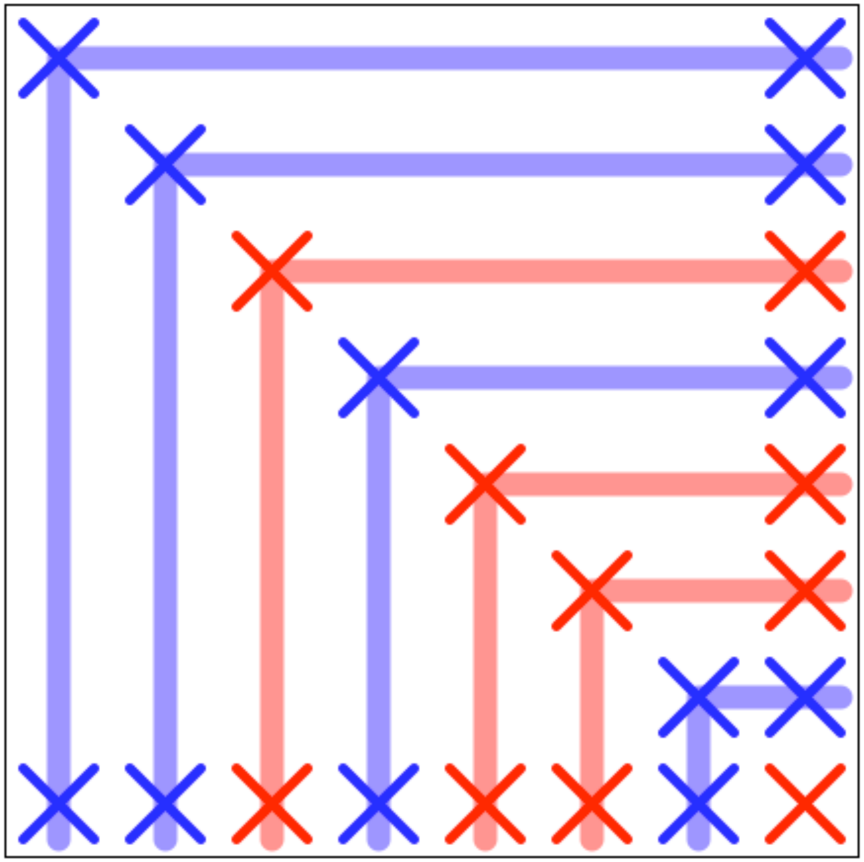
- Loosening load-balancing restriction we can obtain minimum cut (for non-trivial partitioning)

New 2-D Method: “Corner” Partitioning



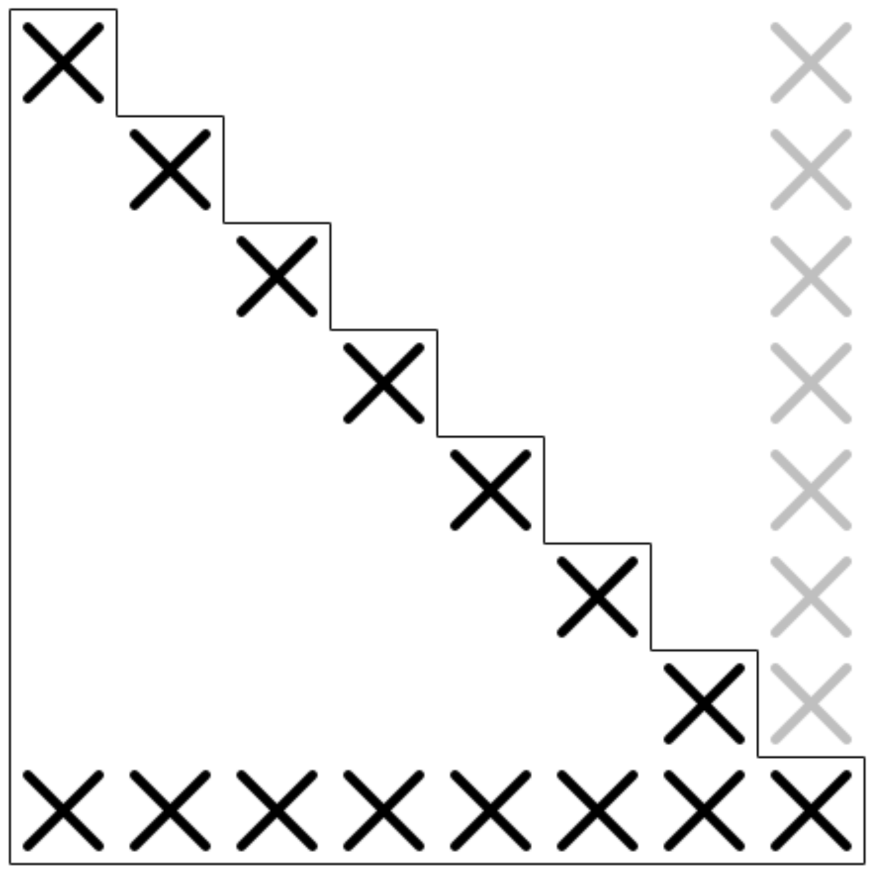
- Optimal partitioning of arrowhead matrix suggests new partitioning method

New 2-D Method: "Corner" Partitioning



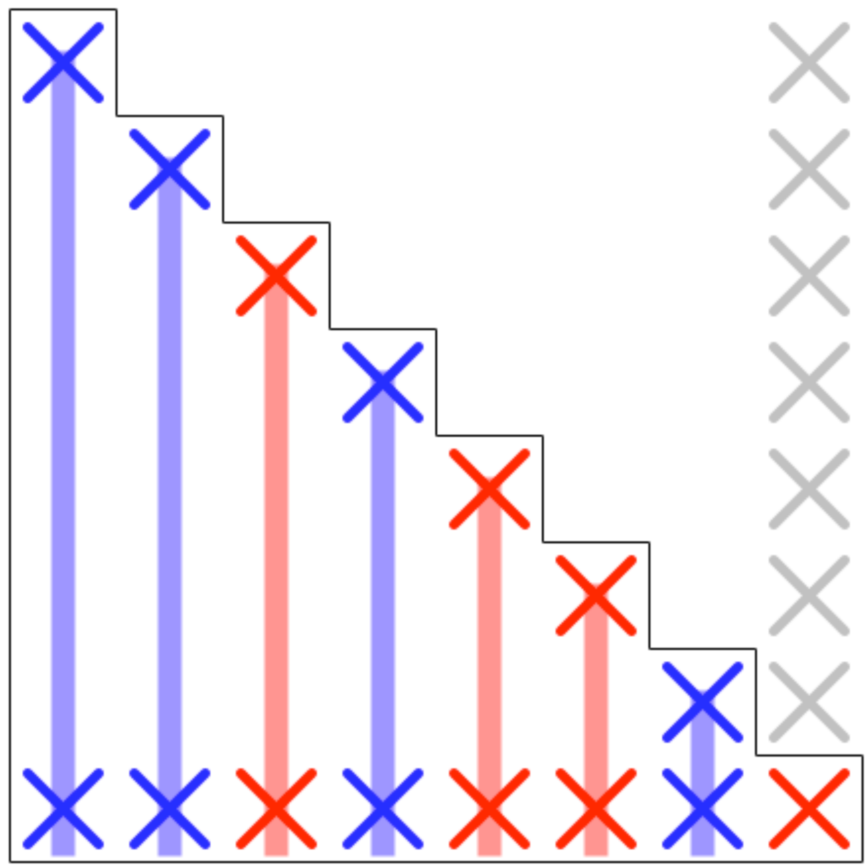
- 1-D partitions reflected across diagonal

New 2-D Method: "Corner" Partitioning



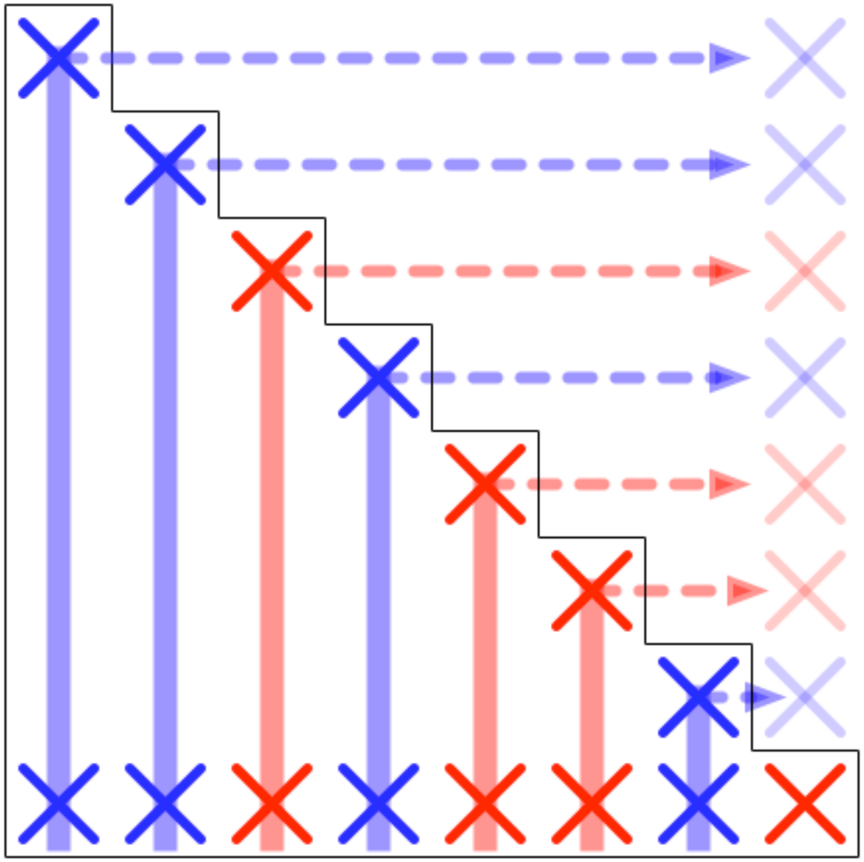
- Take lower triangular part of matrix

New 2-D Method: "Corner" Partitioning



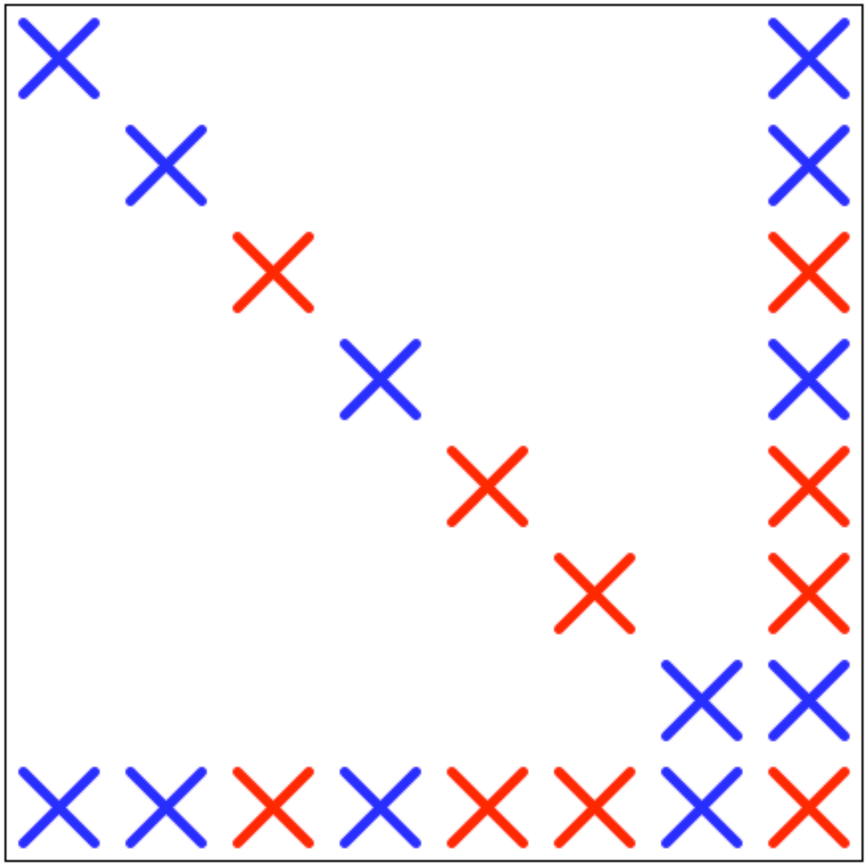
- 1-D (column) hypergraph partitioning of lower triangular matrix

New 2-D Method: “Corner” Partitioning



- Reflect partitioning symmetrically across diagonal

New 2-D Method: "Corner" Partitioning



Volume = 2

- Optimal (non-trivial) partitioning

Comparison of Methods -- Arrowhead Matrix

k	1D Column	Mondriaan	Corner	Fine-Grain
2	29101	29102	**2***	**2***
4	40001	29778	**6***	**6***
16	40012	37459	**30***	**30***
64	40048	39424	**126***	**126***

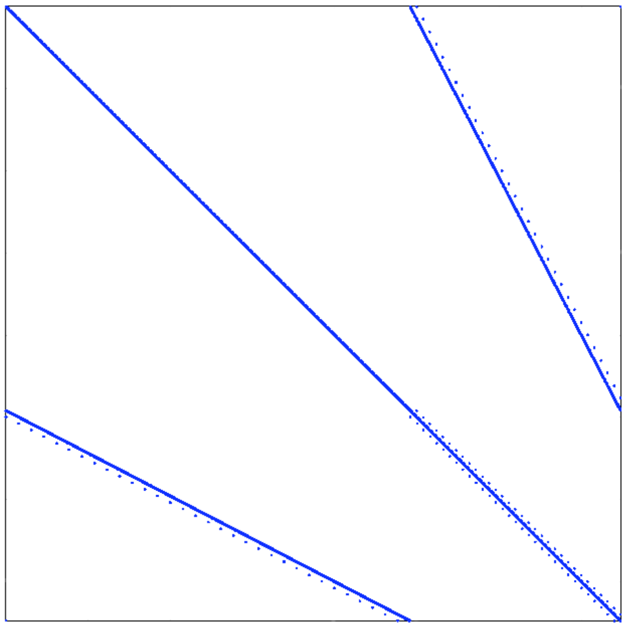
Order n

2(k-1)

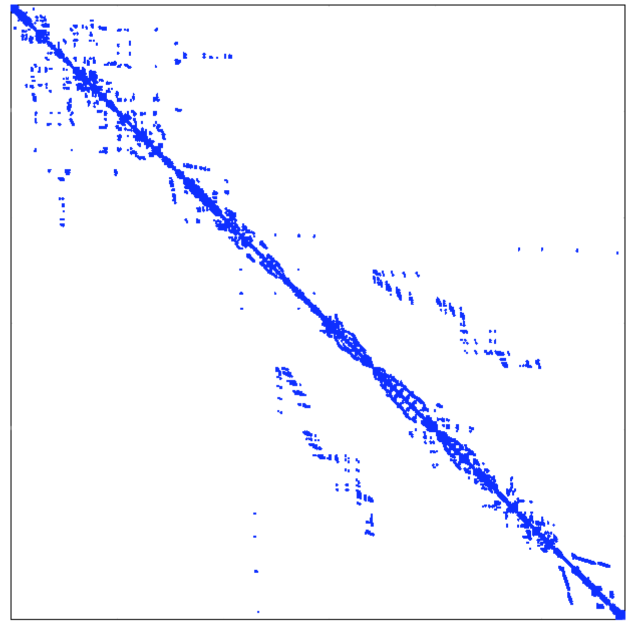
- n = 40,000
- nnz = 119,998

*optimal

Comparison of Methods -- “Real” Matrices



finan512



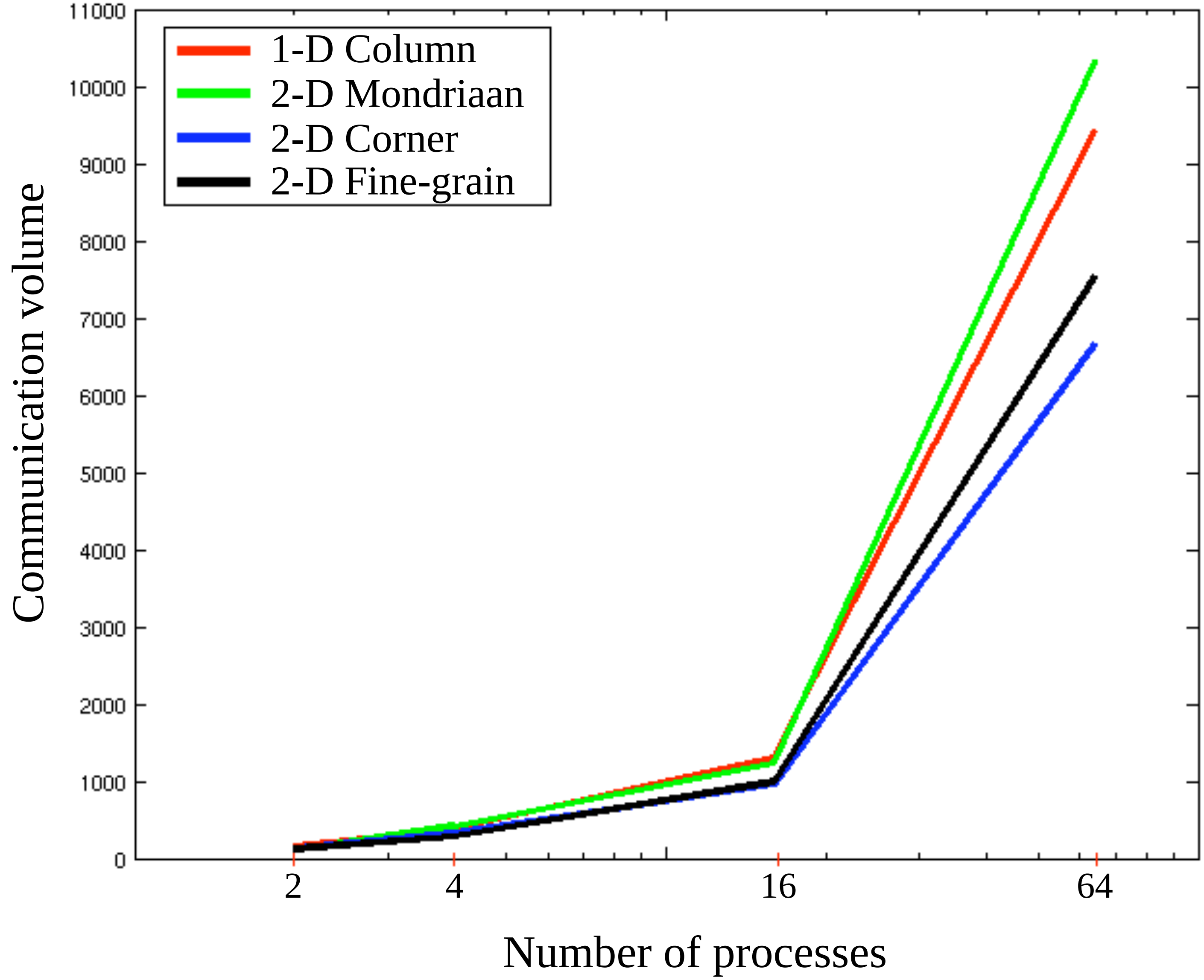
bcsstk30

Portfolio optimization

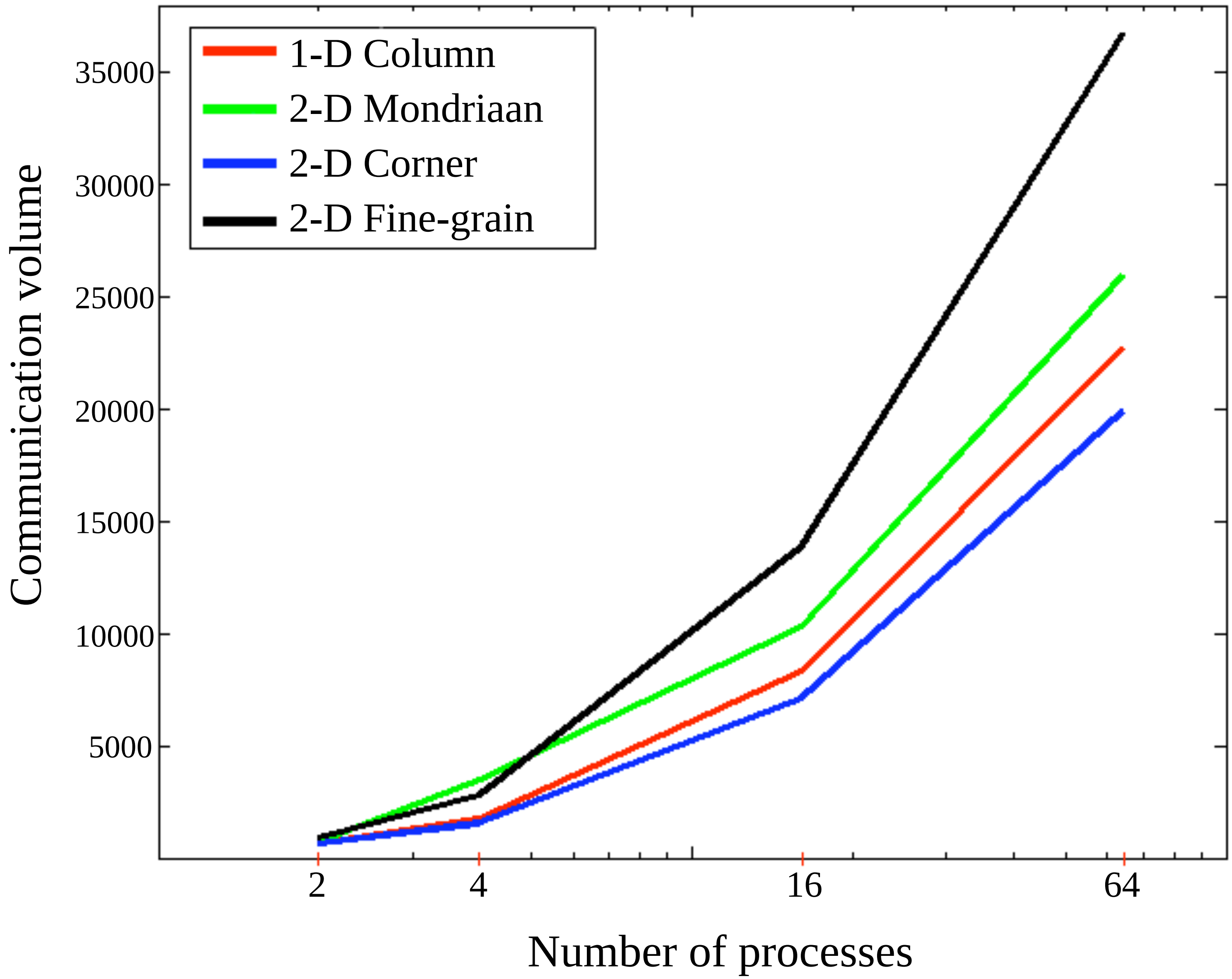
Structural Engineering

matrix	rows	nonzeros
finan512	74,752	596,992
bcsstk30	28,924	2,043,492

Comparison of Methods -- finan512 Matrix



Comparison of Methods -- bcsstk30 Matrix



Summary

- Many models for reducing communication in matrix-vector multiplication
- 1-D partitioning inadequate for many partitioning problems
- New method of 2-D matrix partitioning
  - Improvement for some matrices
  - Faster than fine-grain method

Future Work

- Better intuition for “corner” partitioning method
  - Optimal for arrowhead matrix
  - Good for finan512, bcsstk30 matrices
  - When effective?
- Reordering of matrix rows/columns for “corner” partitioning method
  - Unlike 1-D graph/hypergraph, dependence on ordering
  - Find optimal ordering/partition
  - Extend utility of method

Acknowledgements

- Work at Sandia National Laboratories
  - CSCAPES SciDAC project
- Dr. Erik Boman (SNL)
  - Technical advisor
- Dr. Bruce Hendrickson (SNL)
  - Row/column reordering work
- Zoltan Team (SNL)
  - Used Zoltan for 1-D hypergraph partitioning