Investigating hypergraph-partitioning-based sparse matrix partitioning methods

Bora Uçar

ro:ma, Lyon, France

22 October 2009

Jointly with Ümit V. Çatalyürek (VMWIP)

Outline

Hypergraphs: Definitions

A hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ is a set of vertices $\mathcal{V}$ and a set of hyperedges (nets) $\mathcal{N}$.

A hyperedge $h \in \mathcal{N}$ is a subset of vertices.

A cost $c(h)$ is associated with each hyperedge h.

A weight $w(v)$ is associated with each vertex v.

An undirected graph can be seen as a hypergraph where each hyperedge contains exactly two vertices.

Hypergraphs: Partitioning

Partition

$\Pi = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K\}$ is a K-way vertex partition if

- $\mathcal{V}_k \neq \emptyset$,
- parts are mutually exclusive: $\mathcal{V}_k \cap \mathcal{V}_\ell = \emptyset$,
- parts are collectively exhaustive: $\mathcal{V} = \bigcup \mathcal{V}_k$.

The connectivity $\lambda(h)$ of a hyperedge h is equal to the number of parts in which h has vertices.

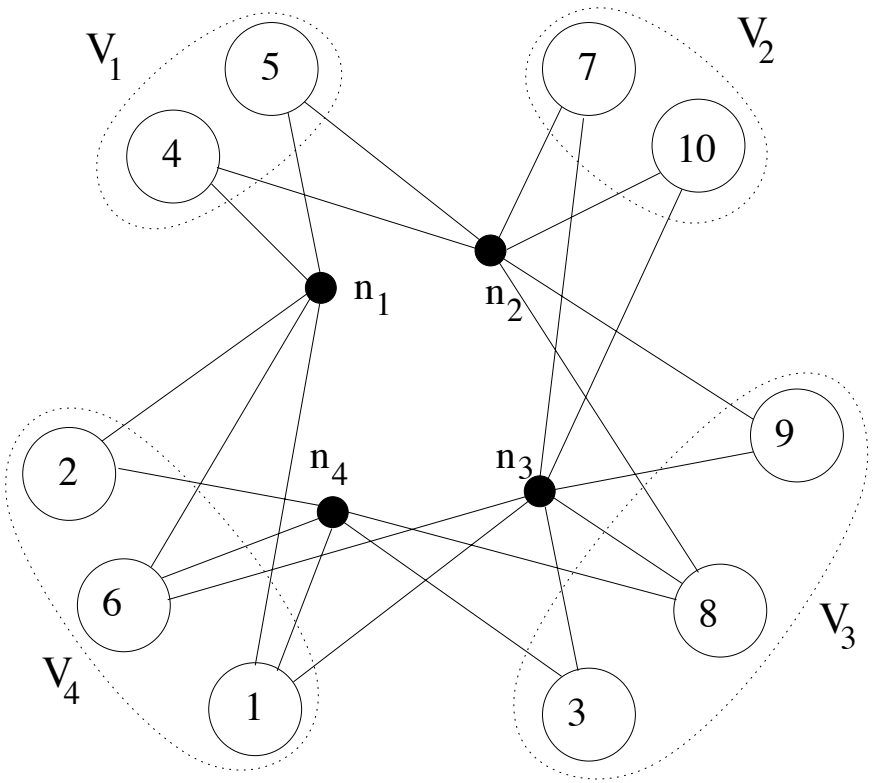
Objective: Minimize cutsize(Π)

$$\sum_h c(h)(\lambda(h) - 1),$$

Constraint: Balanced part weights

$$\sum_{v \in \mathcal{V}_k} w(v) \leq (1 + \varepsilon) \frac{\sum_{v \in \mathcal{V}} w(v)}{K}.$$

Hypergraph partitioning



10 vertices, 4 nets.

Partitioned into 4 parts: $\{4, 5\}$, $\{7, 10\}$, $\{3, 8, 9\}$, and $\{1, 2, 6\}$,

$\lambda(n_1) = 2 \quad \lambda(n_2) = 3$
$\lambda(n_3) = 3 \quad \lambda(n_4) = 2$

$cutsize(\Pi) = c(n_1) + 2c(n_2) + 2c(n_3) + c(n_4)$

(with unit costs 6).

Hypergraphs: Partitioning tools and applications

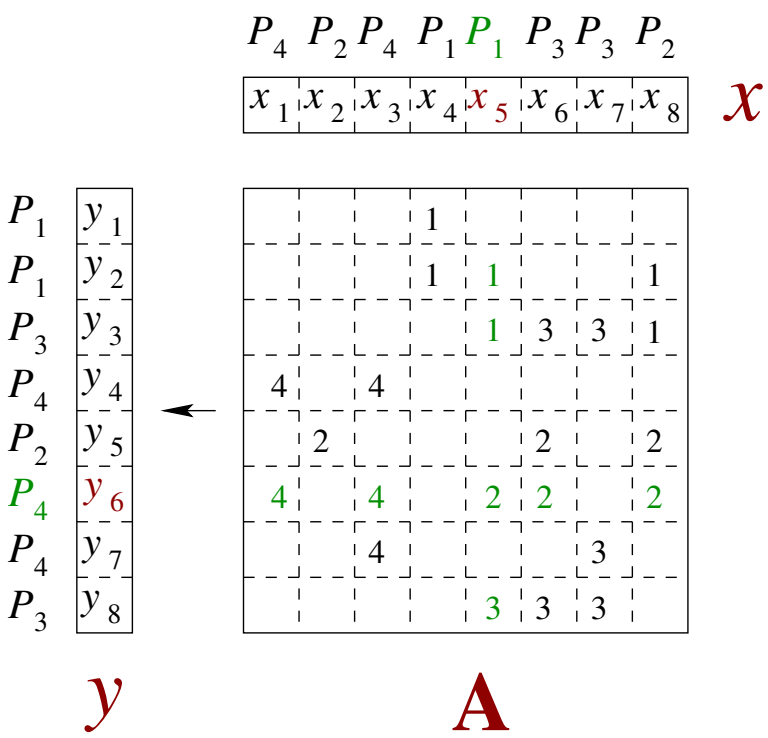
Tools

hMETIS (Karypis and Kumar, Univ. Minnesota),
MLPart (Caldwell, Kahng, and Markov, UCLA/UMich),
Mondriaan (Bisseling and Meesen, Utrecht Univ.),
Par*k*way (Trifunovic and Knottenbelt, Imperial Coll. London),
PaToH (Çatalyürek and Aykanat, Bilkent Univ.),
Zoltan-PHG (Devine, Boman, Heaphy, Bisseling, and Çatalyürek, Sandia National Labs.).

Applications

- VLSI: circuit partitioning,
- Scientific computing: matrix partitioning, ordering, cryptology,
- Parallel/distributed computing: volume rendering, data aggregation, declustering/clustering, scheduling,
- Software engineering, information retrieval, processing spatial join queries, etc.

Parallel sparse matrix-vector multiplies



Row-column-parallel multiplies

To compute $\mathbf{y} \leftarrow \mathbf{Ax}$

1. Expand **x**
   (P_1 sends x_5 to P_2 and P_3.)
2. Scalar multiply and add
   (P_2 computes a partial result
   $y'_6 = a_{65}x_5 + a_{66}x_6 + a_{68}x_8$.)
3. Fold **y**
   (P_2 sends its partial result y'_6 to P_4.)

Parallelization objectives

Achieve load balance

Load of a processor: number of nonzeros.
$\Rightarrow$ assign almost equal number of nonzeros per processor.

Minimize communication cost

Communication cost is a complex function (depends on the machine architecture and the problem size):

- total volume of messages,
- total number of messages,
- max. volume of messages per processor (sends or receives, both?),
- max. number of messages per processor (sends or receives, both?).

The common metric in different works: total volume of communication.

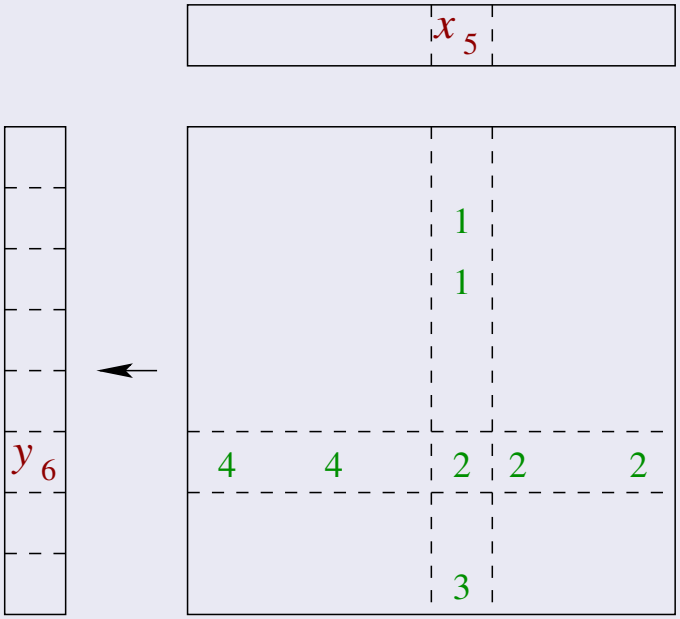
Parallelization problem

Problem definition:

Partition the matrix so that

- processors have equal number of nonzeros,
- minimize the total volume of messages.

Volume of messages



Consider x_5. If assigned to processor:

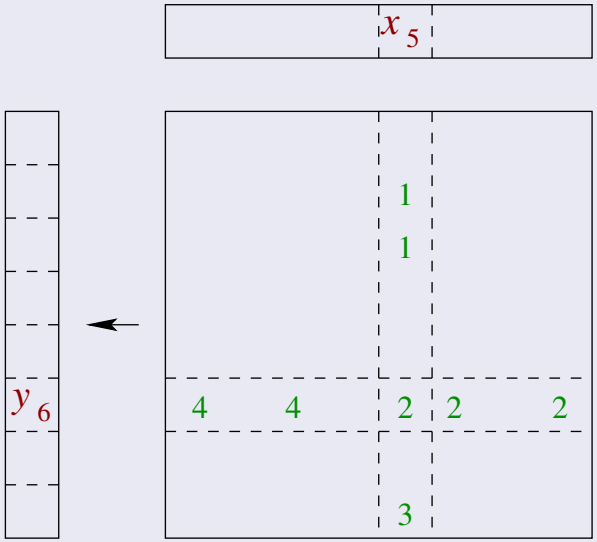
- P_1: 2 units of communication,
- P_2: 2 units of communication,
- P_3: 2 units of communication,
- P_4: 3 units of communication, and does not make sense.

Consider y_6

- Similar

Parallelization problem (cont')

Volume of messages



Nonzeros in column c_j in $s_c(j)$ processors:

- volume of communication is $s_c(j)-1$.

Nonzeros in row r_i in $s_r(i)$ processors:

- volume of communication is $s_r(i)-1$.

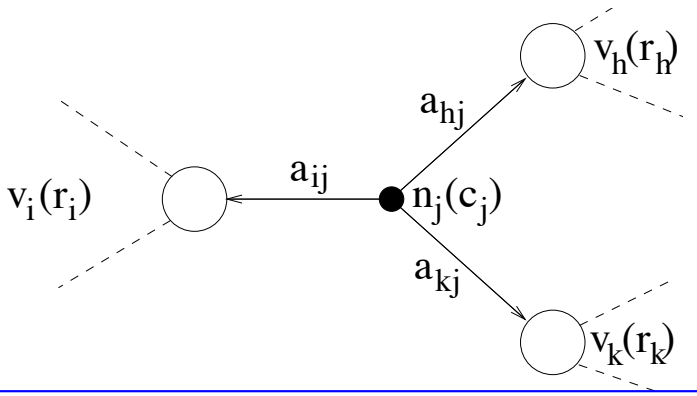
Total volume of communication is $\sum s_c(j)-1 + \sum s_r(i)-1$.

Balance the number of nonzeros per processor and minimize the number of processors sharing a column/row.

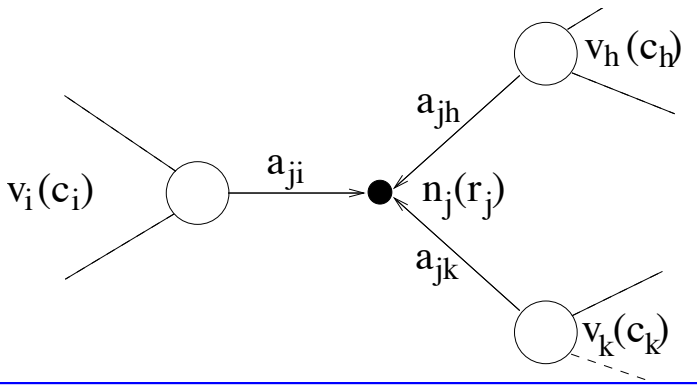
- Equivalent to the hypergraph partitioning problem; it is NP-complete.

Three main models for matrix partitioning

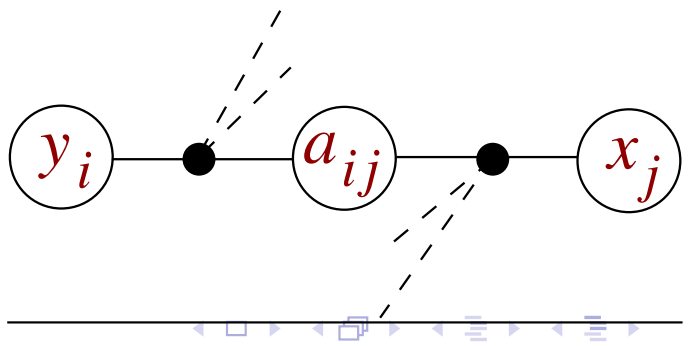
Column-net model: Used for rowwise partitioning. Each column is a net and each row is a vertex.



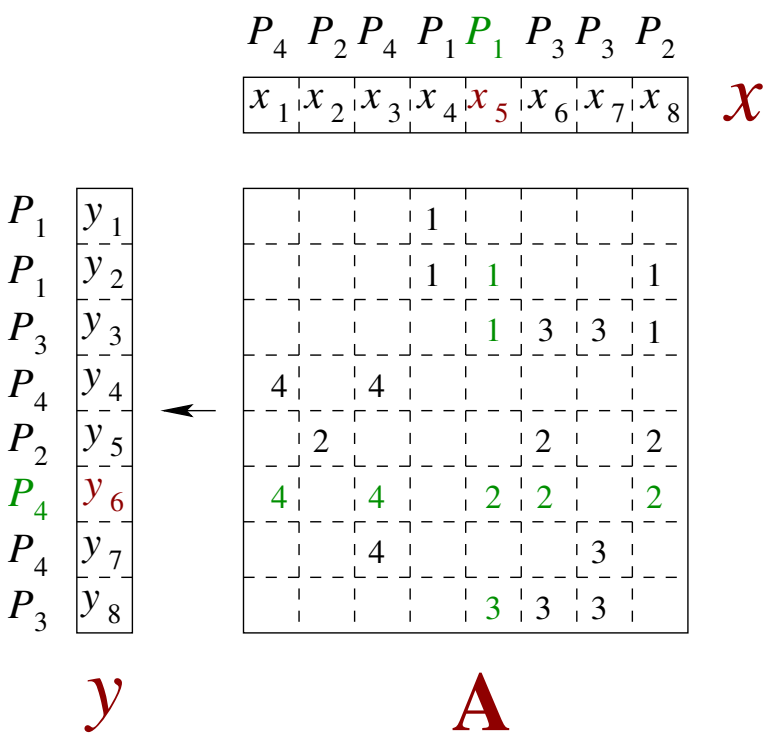
Row-net model: Used for columnwise partitioning. Each row is a net and each column is a vertex.



Fine-grain model: Used for nonzero-based partitioning. Each row is a net, each column is a net, and each nonzero is a vertex.



Parallel sparse matrix-vector multiplies

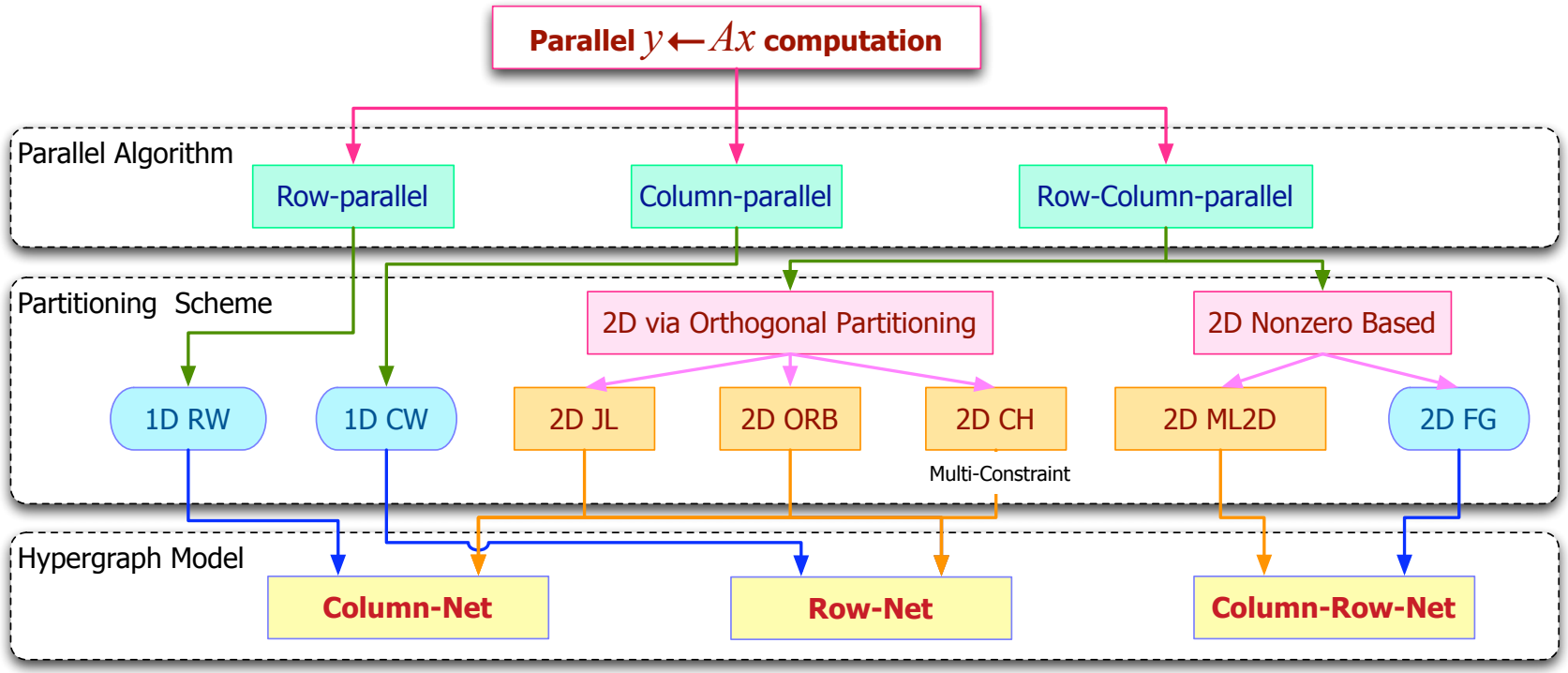


Row-column-parallel multiplies

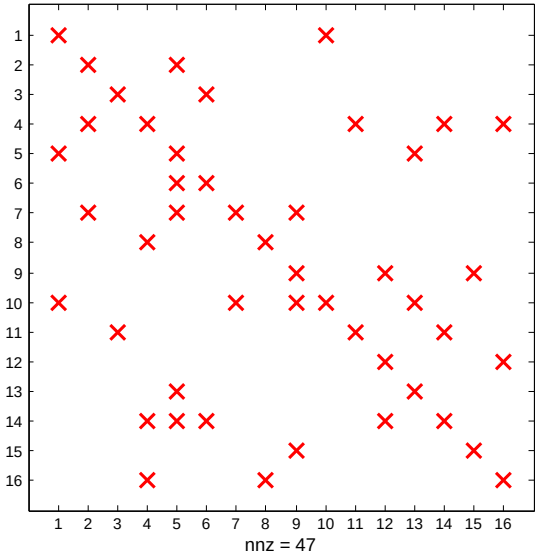
To compute $\mathbf{y} \leftarrow \mathbf{Ax}$

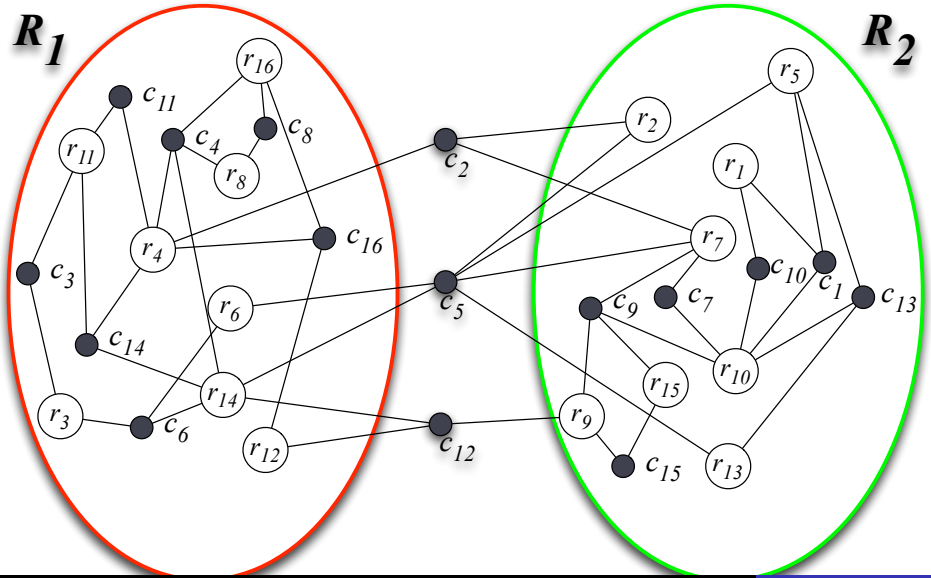
1. Expand $\mathbf{x}$
   (P_1 sends x_5 to P_2 and P_3.)
2. Scalar multiply and add
   (P_2 computes a partial result
   $y'_6 = a_{65}x_5 + a_{66}x_6 + a_{68}x_8$.)
3. Fold $\mathbf{y}$
   (P_2 sends its partial result y'_6 to P_4.)

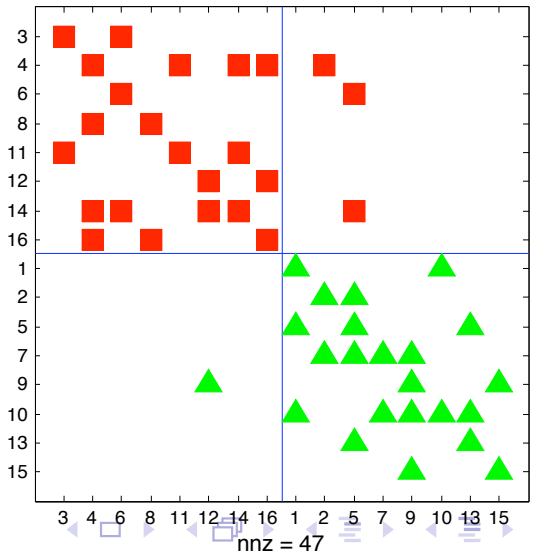
Taxonomy of sparse matrix partitioning methods and models



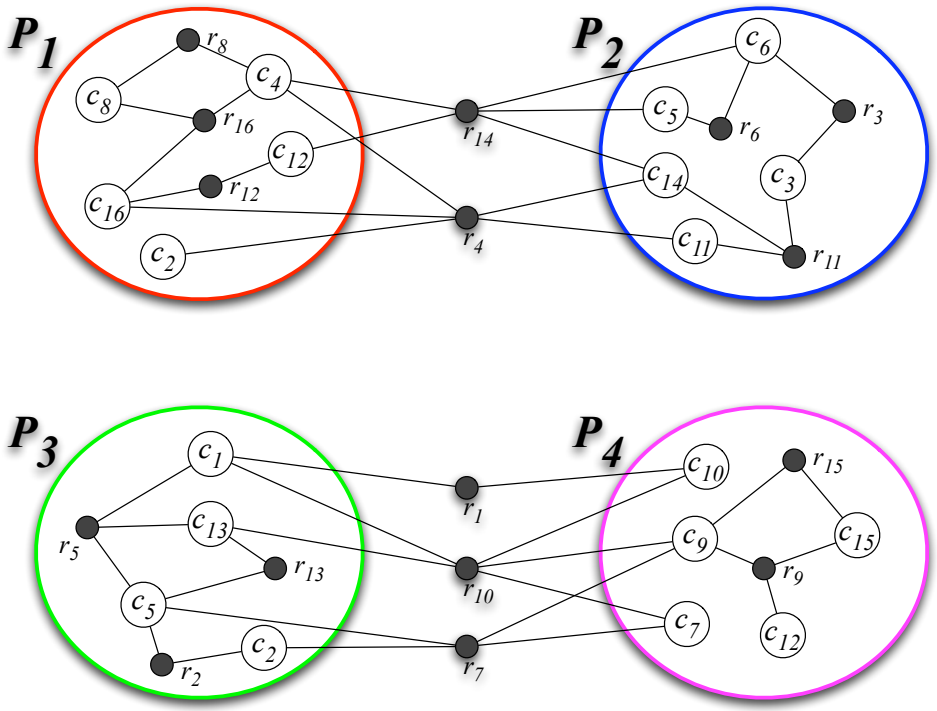
Example: Jagged-like partitioning

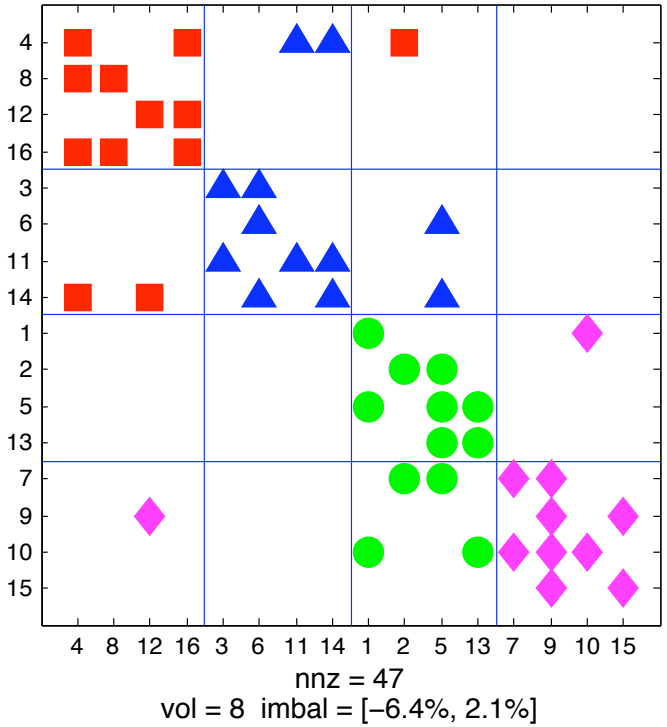






Example: Jagged-like partitioning





Too many alternatives?

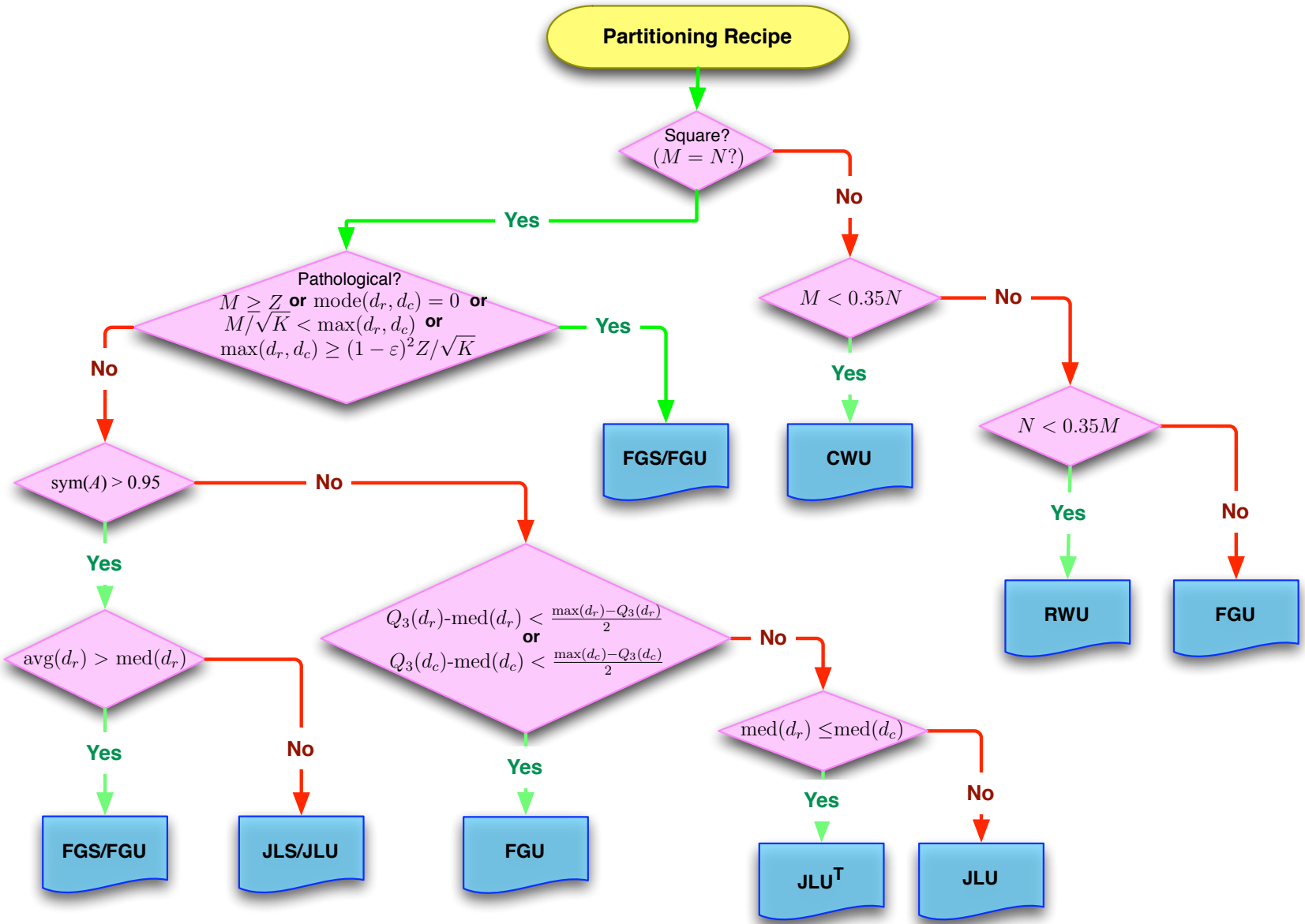
Ideally, for partitioning a given matrix, we would like to choose the best method.

The best is not well-defined. Given that the main objective of the hypergraph-partitioning-based methods is the minimization of the total communication volume, we may content ourselves with the "least total communication volume".

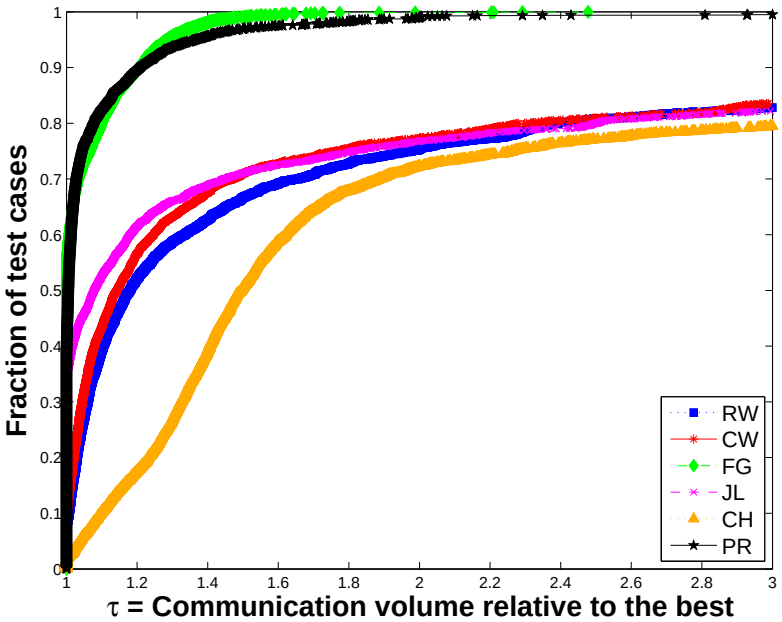
Can we know which method will give the best result, without applying them all?

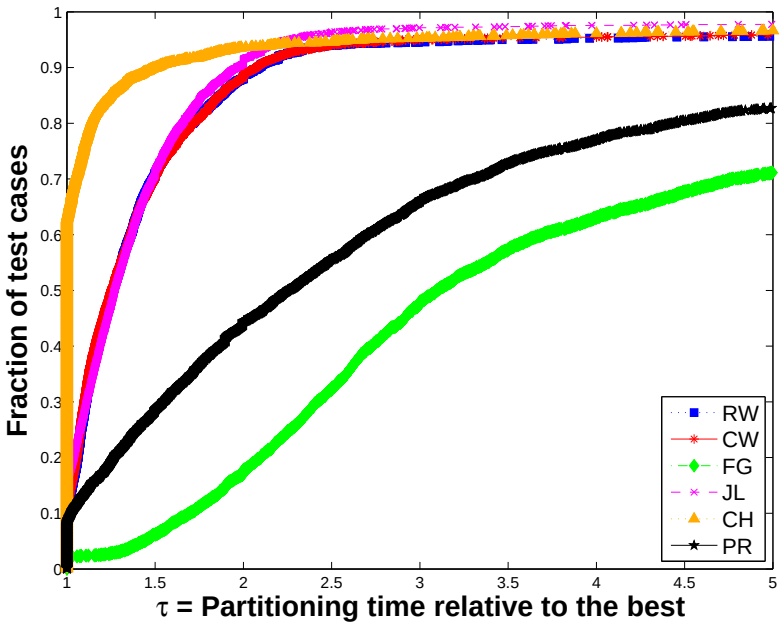
The landscape is not too complicated; the fine-grain method usually obtains the least total communication volume. But, also with the highest run time and, even worse, the highest total number of messages.

A recipe for matrix partitioning



How good is my choice





On the average recipe is faster than the FG method and produces similar results.

How good is my choice

The number of times a specific method has been chosen by the partitioning recipe for 4,100 partitioning instances.

method	$K = 4$	$K = 16$	$K = 64$	$K = 256$
FG sym	350	331	252	148
JL sym	296	272	176	107
RW nonsym	9	8	5	4
CW nonsym	94	82	52	26
FG nonsym	594	567	327	198
JL nonsym	31	23	16	12
JL^T nonsym	43	40	24	13

Would the recipe remain the same for larger K, different matrices?

The holy grail is to be able to tell, at least approximately, the total volume that can be obtained by the methods.

Why do I care?

I do not know any algorithm with approximation guarantee. The most successful and common methods are based on multilevel approach and local search (iterative improvement). Hard to analyze. How good are we doing?

Why do you care?

Suppose you have a parallel algorithm which uses hypergraph models to distribute the matrix and relies on the minimization of the total communication volume.

Can you just run on some instances and report the scalability of your parallel algorithm? How do you know that my/his/our hypergraph partitioning method will do a good job in reducing the communication volume?

Scalability analysis: How

Why do I care?

- I do not know any algorithm with approximation guarantee.
- I did not know any special class of hypergraphs for which any other method than his/my/our software produces considerably better results.
- Graph partitioning is an alternative but not a contender. In reducing the total volume, hypergraphs are much better (besides graph edge cut is not an exact measure of communication).

Why don't I just run on bigger matrices with larger K?

Do I know what to expect?

- There is no approximation guarantee, there is no known optimality results.

Scalability analysis: Proposed methodology

Have expectations

- Examine the models/methods under differing scenarios
- Make educated guess as to what can come out if the initial method were exact.

Special cases

Consider a special class of hypergraphs and develop specialized partitioning algorithms.

Scalability analysis: Scenarios

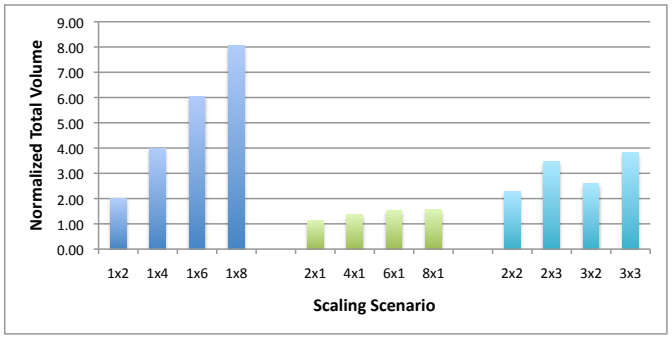
Suppose we have partitioned the matrix A among K processors and obtained the communication volume requirements in fold as V_F and V_E

What can we say about:

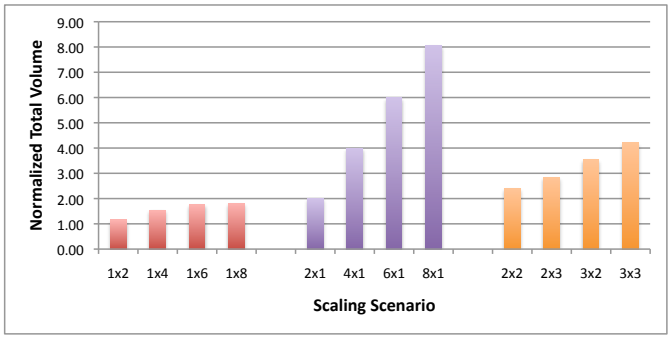
- partitioning of $(A\ A\ A\ \cdots)$
- partitioning of $\begin{pmatrix} A \\ A \\ A \\ \vdots \end{pmatrix}$
- R times replication of the rows, and C times replication of the columns

General formula would look something like $V_F \times (R-1) + V_E \times (C-1)$

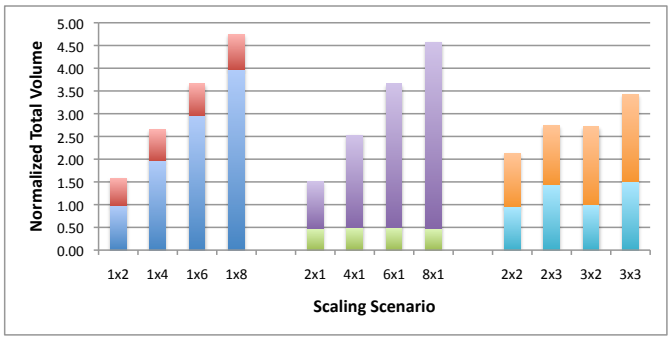
Scalability analysis: Scenarios



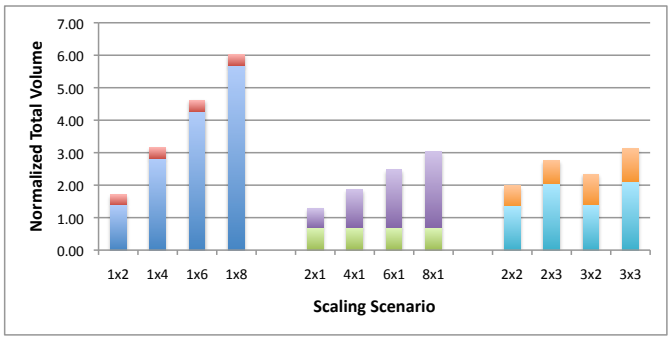
(a) RW



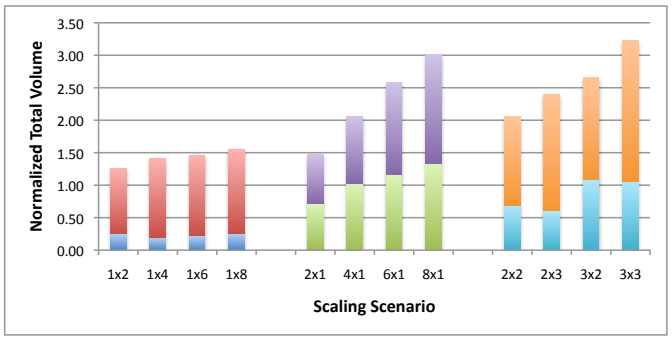
(b) CW

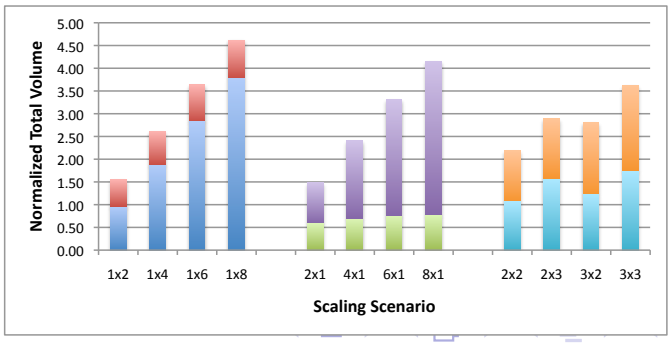


(c) JL



(d) CH





Scalability analysis: Scenarios

The results meet the expectations reasonably, with a few highlights.

Observations

- identical vertices should be detected,
- identical nets should be detected,
- 2D methods scale better than the 1D methods (since the same partitioning algorithms are used for hypergraph partitioning methods, this favors the 2D methods)

Other than these, everything else seems to be reasonable.

Scalability analysis: Special cases

The 2D Laplace equation

- Used in solving the heat equation,
- Had been used to analyze direct methods since 1970s — paving the way for the development of ordering and partitioning
- Came to be known as the "model problem"

The 5-point stencil

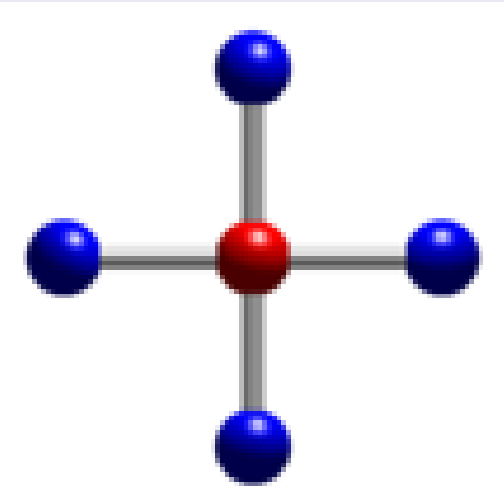
- Discretize a (say square) 2D domain with step size h, resulting in an $n \times n$ mesh of points
- The stencil consist of a point along with four neighbors.
- Used in finite difference approximation of the derivatives at mesh points. In 2D, known as the Laplacian of a function of two variables

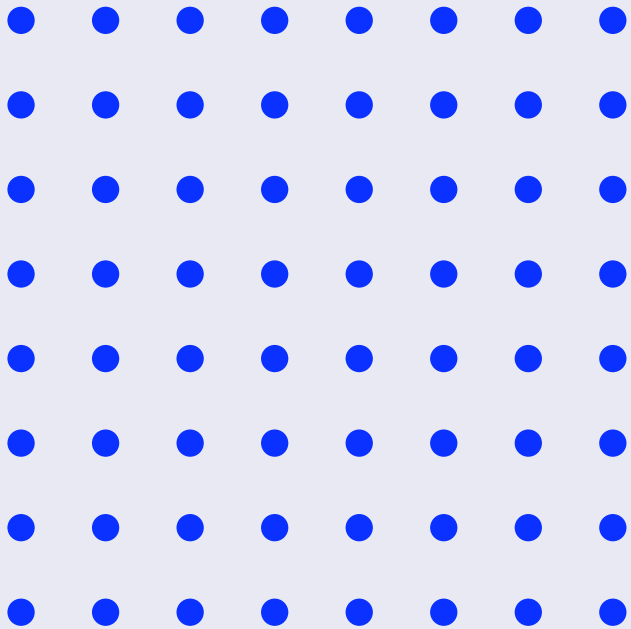
$$\Delta f(x,y) \approx \frac{f(x-h,y) + f(x+h,y) + f(x,y-h) + f(x,y+h) - 4f(x,y)}{h^2}$$

Scalability analysis: Special cases

The 5-point stencil and the 8×8 mesh

$$\Delta f(x,y) \approx \frac{f(x-h,y) + f(x+h,y) + f(x,y-h) + f(x,y+h) - 4f(x,y)}{h^2}$$

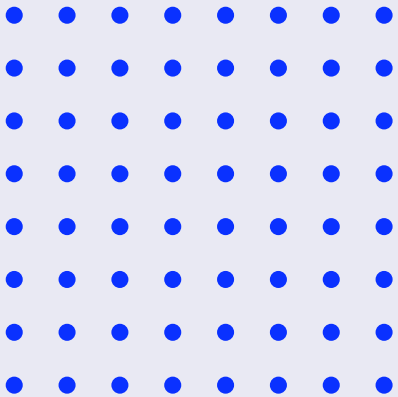


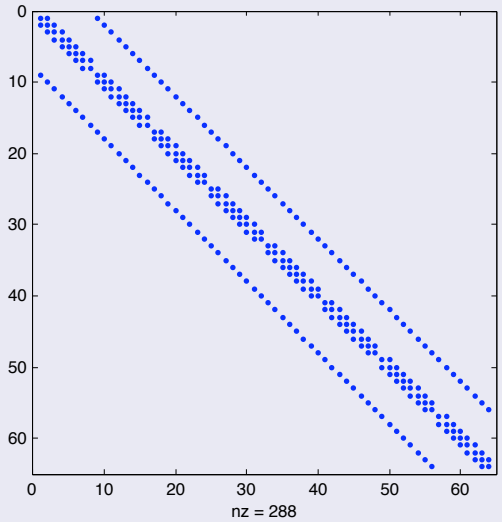


Scalability analysis: Special cases

The 5-point stencil and the 8×8 mesh and the associated matrix

The mesh is of size $n \times n$, the matrix is of size $N \times N$ with $N = n^2$





Calculations with the mesh and the matrix

$$x_{i,j}^{(k+1)} \leftarrow \frac{x_{i-1,j}^{(k)} + x_{i+1,j}^{(k)} + x_{i,j-1}^{(k)} + x_{i,j+1}^{(k)} - 4x_{i,j}^{(k)}}{h^2}$$

$$x^{(k+1)} \leftarrow Ax^{(k)} + b$$

Scalability analysis: Partitioning the 5-point stencil meshes

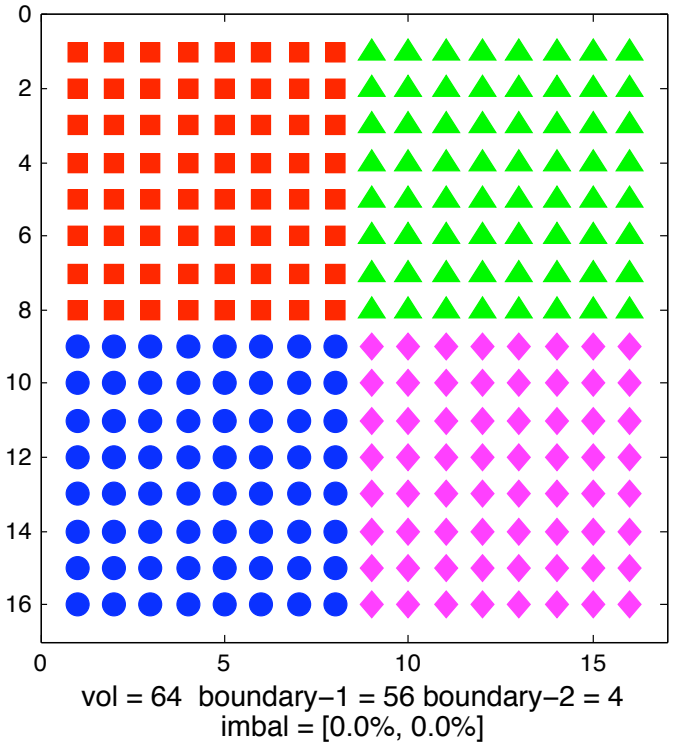
Partitioning the points of the mesh corresponds to partitioning the rows/cols of the associated (symmetric) matrix.

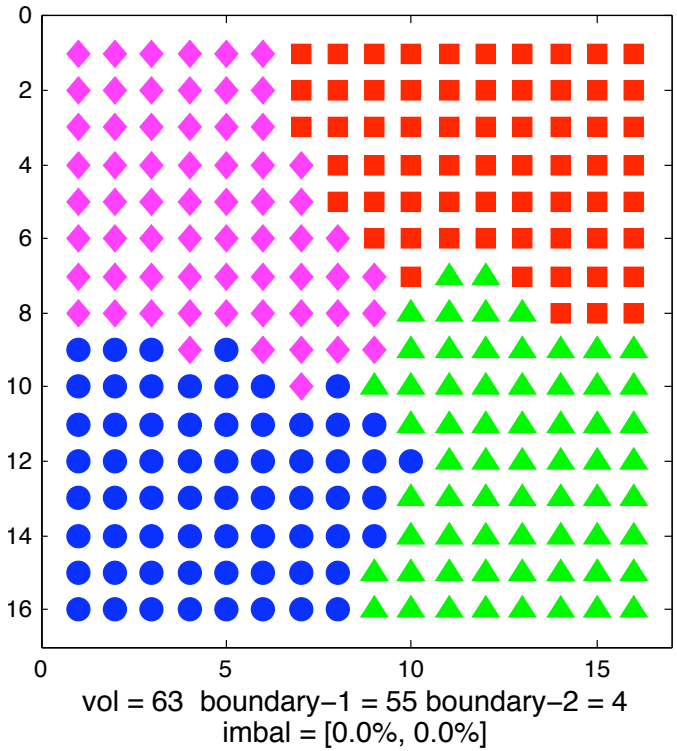
With the objective of minimizing the total communication volume, the problem is equivalent to the hypergraph partitioning problem.

Solving the partitioning problem with some specialized methods will help evaluate the hypergraph partitioning methods/models.

Scalability analysis: Partitioning the 5-point stencil meshes

The experience with the nested-dissection/recursive bisection-based approached suggests the Cartesian distribution:

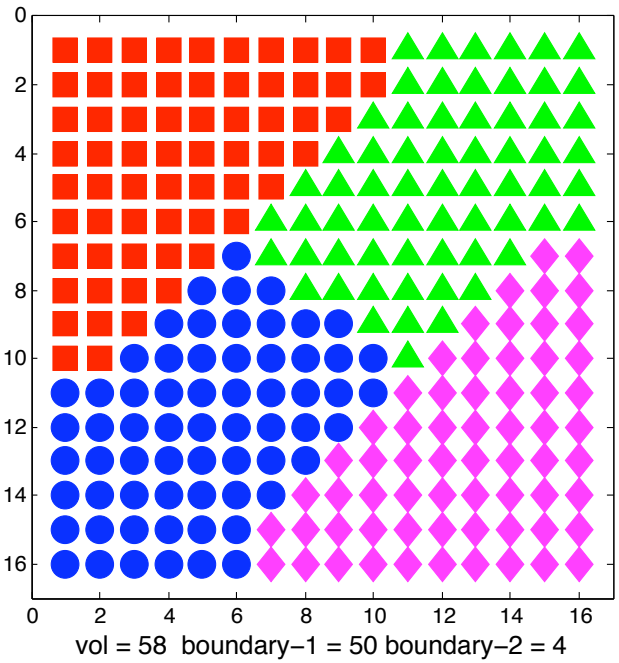




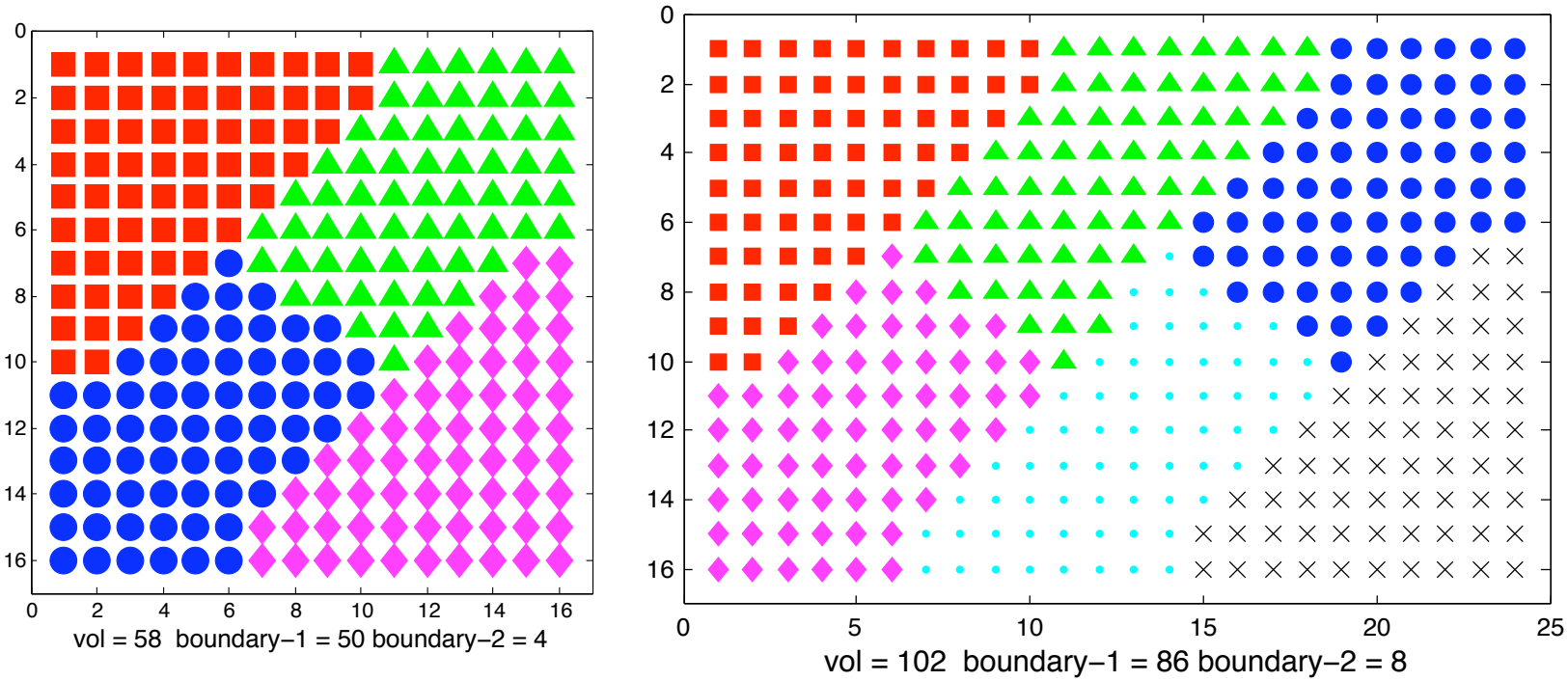
$$vol_{cart}(M, N, P, Q) = 2 \times (P - 1) \times N + 2 \times (Q - 1) \times M$$

Scalability analysis: Quadrisection

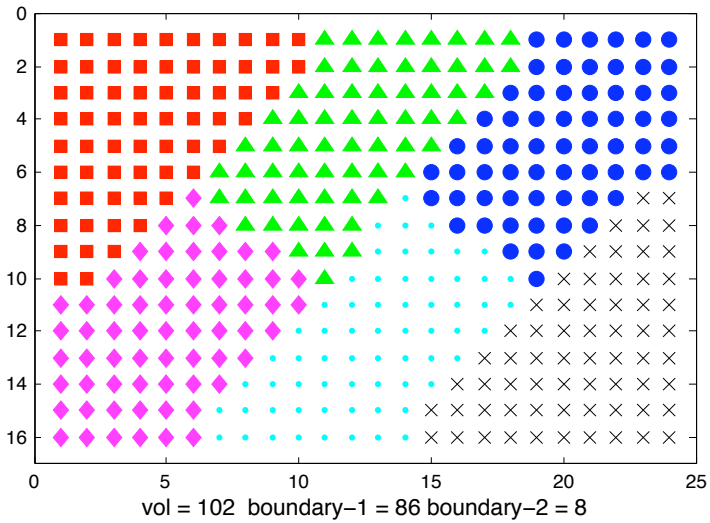
1: $M_{1/8} = \frac{M}{8}$; $M_{1/2} = \frac{M}{2}$; $M_{3/8} = M_{1/2} - M_{1/8}$
2: $M_{+1} = M + 1$
3: $part(M_{3/8}, M_{3/8}) = 1$
4: $part(M_{+1} - M_{3/8}, M_{+1} - M_{3/8}) = 4$
5: target $= M \times M/4 - M_{3/8} \times M_{3/8}$
6: $i = M_{3/8}$; $j = M_{3/8}$
7: **while** target> 0 **do**
8: $\quad i = i + 1$; $j = j - 1$
9: $\quad$ target$=$ target $-2 \times j$
10: $\quad$ **if** target< 0 **then**
11: $\quad\quad j = j +$ target$/2$; target $= 0$
12: $\quad part(i, j) = 1$; $part(M_{+1} - i, M_{+1} - j) = 4$
13: $\quad part(j, i) = 1$; $part(M_{+1} - j, M_{+1} - i) = 4$
14: **for** $k = 1$ **to** j **do**
15: $\quad part(i, k) = 1$; $part(M_{+1} - i, M_{+1} - k) = 4$
16: $\quad part(k, i) = 1$; $part(M_{+1} - k, M_{+1} - i) = 4$
17: **for** $k = M_{3/8} + 1$ **to** $M_{1/2}$ **do**
18: $\quad part(k, k) = 2$; $part(M_{+1} - k, M_{+1} - k) = 3$
19: bfsColor(1, 1, 1); bfsColor(1, M, 2)
20: bfsColor(M, 1, 3); bfsColor(M, M, 4)

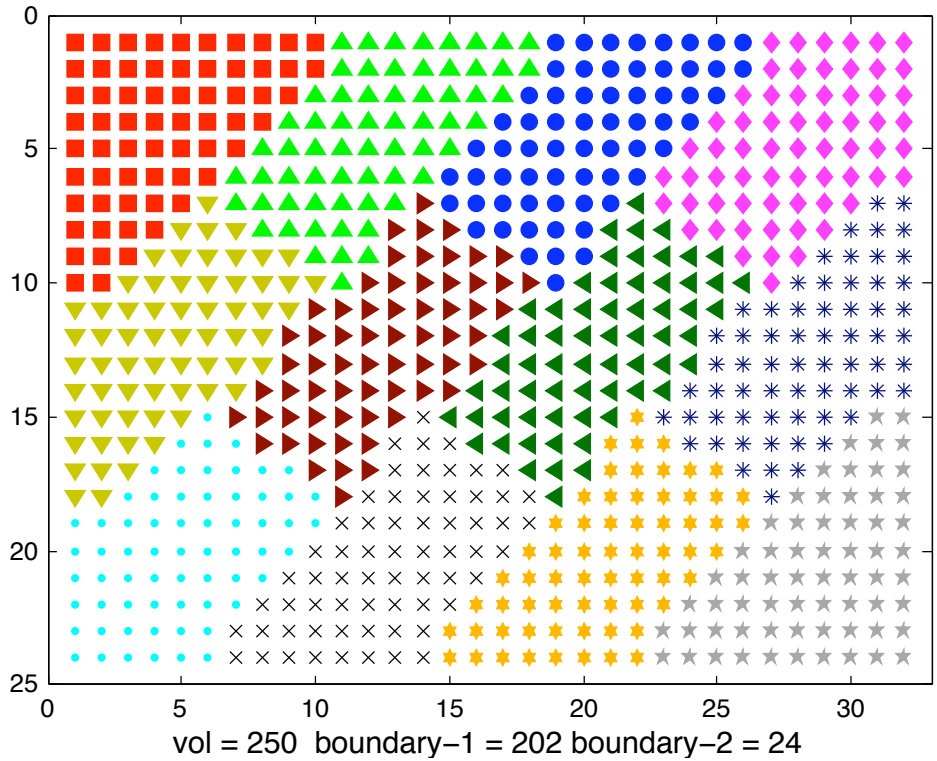


Scalability analysis: Extending the quadrisection to K

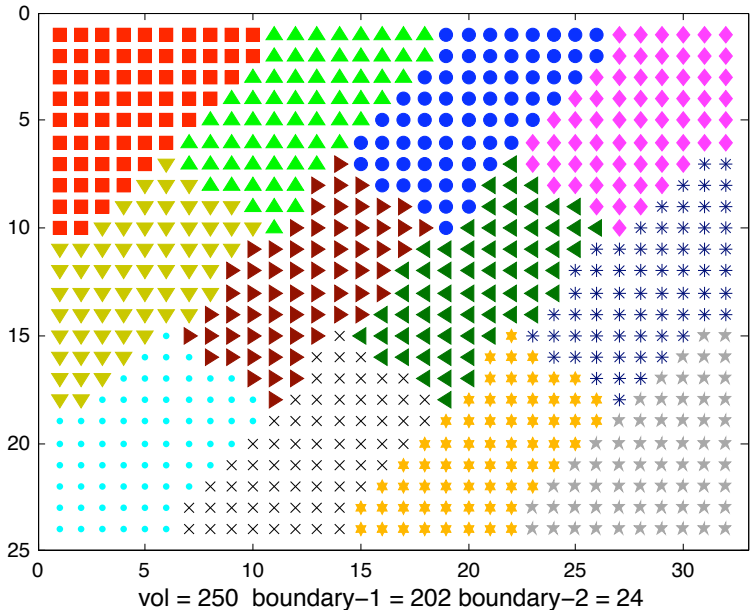


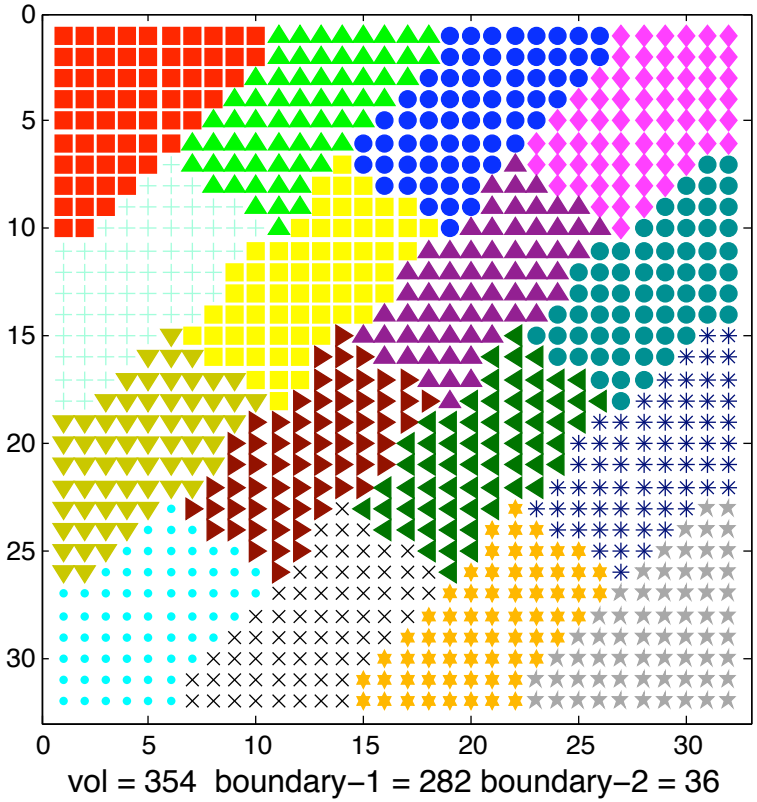
Scalability analysis: Extending the quadrisection to K





Scalability analysis: Extending the quadrisection to K





Scalability analysis: Results

Mesh Size	K	MeshPart	Cartesian Part		1D Hypergraph		2D finegrain
64x64	4	226	256	(1.13)	252	(1.11)	253
64x64	16	666	768	(1.15)	739	(1.11)	713
128x128	16	1290	1536	(1.19)	1475	(1.14)	1473
128x128	64	3066	3584	(1.17)	3353	(1.09)	3247
256x256	4	898	1024	(1.14)	1015	(1.13)	1007
256x256	16	2538	3072	(1.21)	2979	(1.17)	3038
256x256	256	13050	15360	(1.18)	13893	(1.06)	13236
512x512	4	1794	2048	(1.14)	2051	(1.14)	2018
512x512	16	5034	6144	(1.22)	6272	(1.25)	6114
512x512	64	11466	14336	(1.25)	13648	(1.19)	13843
512x512	1024	53754	63488	(1.18)	56306	(1.05)	56314
1024x1024	16	10026	12288	(1.23)	12251	(1.22)	
1024x1024	64	22666	28672	(1.26)	28279	(1.25)	
1024x1024	256	48330	61440	(1.27)	58598	(1.21)	
1024x1024	1024	101866	126976	(1.25)	114223	(1.12)	
2048x2048	16	20010	24576	(1.23)	24382	(1.22)	
2048x2048	64	45066	57344	(1.27)	56890	(1.26)	
2048x2048	256	95370	122880	(1.29)	117996	(1.24)	
2048x2048	1024	198090	253952	(1.28)	234477	(1.18)	
average				(1.21)		(1.16)	

MeshPart: Analysis

$$vol(M, M, 2, 2) = \frac{7 \times M}{2} + 2\,. \tag{1}$$

This can be derived by tracing the algorithm.

$$\begin{aligned} vol(M, N, P, Q) = {} & (3 \times P \times Q - (P + Q) - 1) \times R \\ & + (P - 1) \times (3 \times Q - 5) \\ & + (Q - 1) \times (3 \times P - 5)\,, \end{aligned} \tag{2}$$

where $R = M/P = N/Q$.

How did we obtain these formulas?

Compare with
$vol_{cart}(M, N, P, Q) = 2 \times (P - 1) \times N + 2 \times (Q - 1) \times M$

Plans

Prove or disprove:

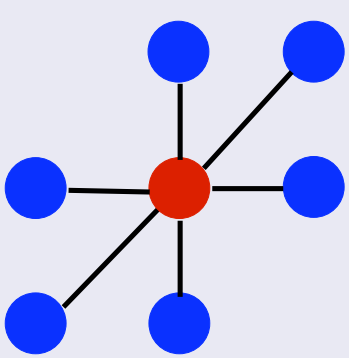
The proposed quadrisection algorithm for partitioning the $M \times M$, 5-point stencil mesh is exact, i.e., the minimum total volume is given by $\frac{7M}{2} + 2$.

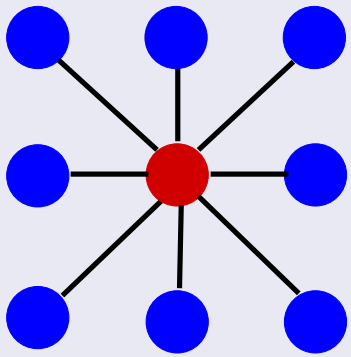
Can we claim

the same for general K?

Je n'ai aucune idée. Si vous en avez, je suis preneur.

Extension to other type of meshes





And the corresponding structures in 3D.