Partitioning for applications: using Mondriaan in mesh-based computations

Rob H. Bisseling, Albert-Jan Yzelman, Bas Fagginger Auer

Mathematical Institute, Utrecht University

Albert-Jan

Bas

Deltares Lunch Seminar Delft, November 8, 2010

Universiteit Utrecht

Mesh partitioning

Laplacian operator

Bulk synchronous parallel communication cost

Diamond-shaped subdomains

3D partitioning

Matrix partitioning

Parallel sparse matrix–vector multiplication (SpMV)

Visualisation by MondriaanMovie

Hypergraphs

Ordering matrices for faster SpMV

Separated Block Diagonal structure

Where meshes meet matrices

Conclusions and future work

Motivation: CFD and other applications

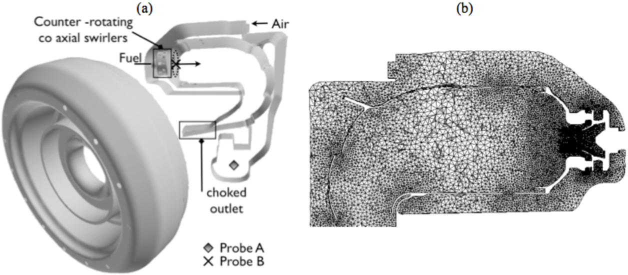
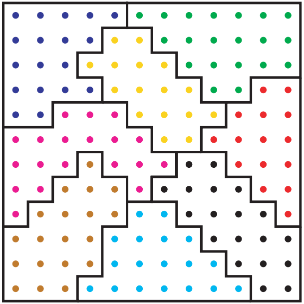


Fig. 7 Full annular aeronautical burner computed with AVBP (LES) for a thermo-acoustic analysis of the burner: (a) computational domain and (b) typical mesh resolution in the injector region.

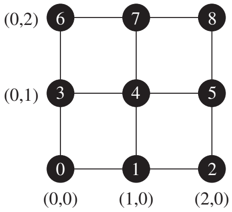
- Source: N. Gourdain et al. 'High performance Parallel Computing of Flows in Complex Geometries. Part 2: Applications' *Computational Science and Discovery* 2009.

2D rectangular mesh partitioned over 8 processors



- In many applications, a physical domain can be partitioned naturally by assigning a contiguous subdomain to every processor.
- Communication is only needed for exchanging information across the subdomain boundaries.
- Grid points interact only with a set of immediate neighbours, to the north, east, south, and west.

2D Laplacian operator for $k \times k$ grid



Compute

$$\Delta_{i,j} = x_{i-1,j} + x_{i+1,j} + x_{i,j+1} + x_{i,j-1} - 4x_{i,j}, \quad \text{for } 0 \leq i,j < k,$$

where $x_{i,j}$ denotes e.g. the temperature at grid point (i,j).
By convention, $x_{i,j} = 0$ outside the grid.

- $x_{i+1,j} - x_{i,j}$ approximates the derivative of the temperature in the i-direction.
- $(x_{i+1,j} - x_{i,j}) - (x_{i,j} - x_{i-1,j}) = x_{i-1,j} + x_{i+1,j} - 2x_{i,j}$ approximates the second derivative.

Relation operator–matrix

$$A = \begin{bmatrix} -4 & 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & -4 & 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & -4 & \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\ 1 & \cdot & \cdot & -4 & 1 & \cdot & 1 & \cdot & \cdot \\ \cdot & 1 & \cdot & 1 & -4 & 1 & \cdot & 1 & \cdot \\ \cdot & \cdot & 1 & \cdot & 1 & -4 & \cdot & \cdot & 1 \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot & -4 & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & -4 & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & -4 \end{bmatrix}$$

$\mathbf{u} = A\mathbf{v} \Longleftrightarrow$

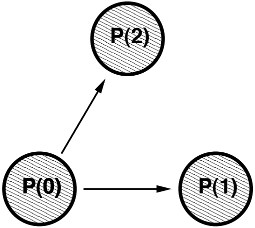
$$\Delta_{i,j} = x_{i-1,j} + x_{i+1,j} + x_{i,j+1} + x_{i,j-1} - 4x_{i,j}, \quad \text{for } 0 \leq i,j < k.$$

Finding a mesh partitioning

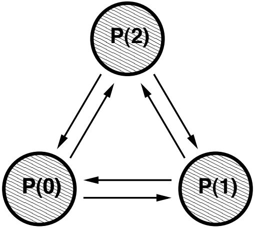
- We must assign each grid point to a processor.
- We assign the values $x_{i,j}$ and $\Delta_{i,j}$ to the owner of grid point (i,j).
- Each point of the grid has an amount of computation associated with it determined by the operator.
- Here, an interior point has 5 flops; a border point 4 flops; a corner point 3 flops.

Our parallel cost model: BSP

2-relations:



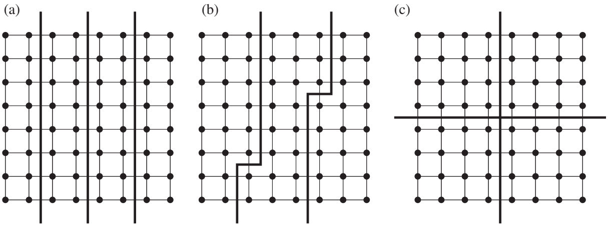
(a)



(b)

- Bulk synchronous parallel (BSP) model by Valiant (1990): a bridging model for parallel computing
- An *h*-relation is a communication phase (superstep) in which every processor sends and receives at most h data words: $h = \max\{h_{\mathrm{send}}, h_{\mathrm{recv}}\}$
- $T(h) = hg + l$, where g is the time per data word and l the global synchronisation time

Partition into strips and blocks



- (a) Partition into strips: long Norwegian borders,

$$T_{\text{comm, strips}} = 2kg.$$

- (b) Boundary corrections improve load balance.
- (c) Partition into square blocks: shorter borders,

$$T_{\text{comm, squares}} = \frac{4k}{\sqrt{p}} g \qquad (\text{for } p > 4).$$

Surface-to-volume ratio

- The communication-to-computation ratio for square blocks is

$$\frac{T_{\mathrm{comm,\ squares}}}{T_{\mathrm{comp,\ squares}}} = \frac{4k/\sqrt{p}}{5k^2/p}g = \frac{4\sqrt{p}}{5k}g.$$

- This ratio is often called the surface-to-volume ratio, because in 3D the surface of a domain represents the communication with other processors and the volume represents the amount of computation of a processor.

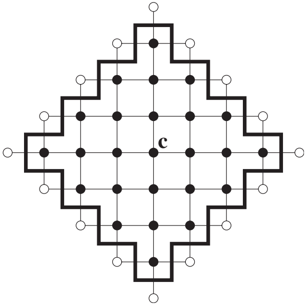
What do we do at scientific workshops?

Participants of HLPP 2001, International Workshop on High-Level Parallel Programming, Orléans, France, June 2001, studying Château de Blois.

The high-level object of our study

Universiteit Utrecht

Blocks are nice, but diamonds . . .



$r = 3$

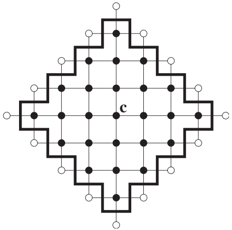
- Digital diamond, or closed l_1-sphere, defined by

$$B_r(c_0, c_1) = \{(i, j) \in \mathbf{Z}^2 : \ |i - c_0| + |j - c_1| \le r\},$$

for integer radius $r \ge 0$ and centre $\mathbf{c} = (c_0, c_1) \in \mathbf{Z}^2$.

- $B_r(\mathbf{c})$ is the set of points with Manhattan distance $\le r$ to the central point $\mathbf{c}$.

Points of a diamond



$r = 3$

- The number of points of $B_r(\mathbf{c})$ is

$$\begin{aligned} & 1 + 3 + 5 + \cdots + (2r-1) + (2r+1) + (2r-1) + \cdots + 1 \\ = \; & 2r^2 + 2r + 1. \end{aligned}$$

- The number of neighbouring points is $4r + 4$.
- This is also the number of ghost cells needed in a parallel grid computation.

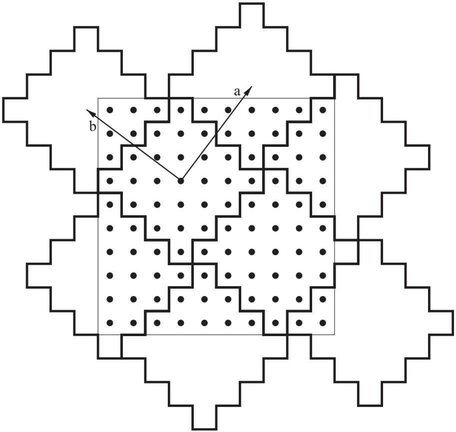
Diamonds are forever

- For a $k \times k$ grid and p processors, we have
$$k^2 = p(2r^2 + 2r + 1) \approx 2pr^2.$$
- Just on the basis of $4r + 4$ receives from neighbour points, we have
$$\frac{T_{\text{comm, diamonds}}}{T_{\text{comp, diamonds}}} = \frac{4r+4}{5(2r^2+2r+1)} g \approx \frac{2}{5r} g \approx \frac{2\sqrt{2p}}{5k} g.$$
- Compare with value $\frac{4\sqrt{p}}{5k} g$ for square blocks: factor $\sqrt{2}$ less.
- This gain was caused by reuse of data: the value at a grid point is used twice but sent only once.
- Also $\sqrt{2}$ less memory for ghost cells.

Alhambra: tile the whole space

(2001)

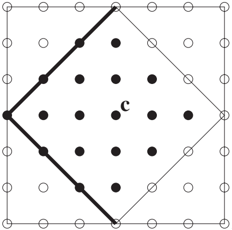
Tile the whole sky with diamonds



$r = 3$

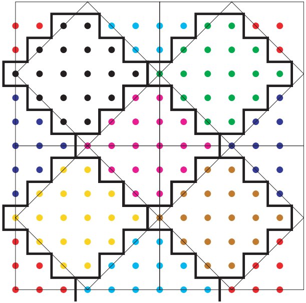
Diamond centres at $\mathbf{c} = \lambda\mathbf{a} + \mu\mathbf{b}$, $\lambda, \mu \in \mathbf{Z}$,
where $\mathbf{a} = (r, r+1)$ and $\mathbf{b} = (-r-1, r)$.
Good method for an infinite grid.

Practical method for finite grids



- Discard one layer of points from the north-eastern and south-eastern border of the diamond.
- For $r = 3$, the number of points decreases from 25 to 18.

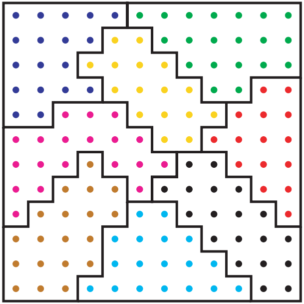
12×12 computational grid: periodic partitioning



8 processors

- Total computation: 672 flops. Avg 84. Max 90.
- Communication: 104 values. Avg 13. Max 14.
- Total time: $90 + 14g = 90 + 14 \cdot 10 = 230$ (ignoring $2l$).
- 8 rectangular blocks of size 6×3 blocks:
  time is $87 + 15 \cdot 10 = 237$.

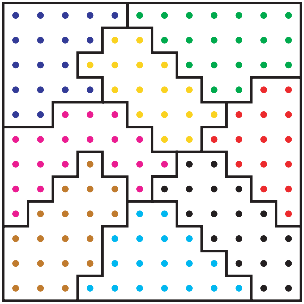
12×12 computational grid: Mondriaan partitioning



8 processors

- Partitioning obtained by translating into a sparse matrix. This treats the structured grid as unstructured.
- Total computation: 672 flops. Avg 84. Max 91. (allowed imbalance $\epsilon = 10\%$.)
- Communication: 85 values. Avg 10.525. Max 16.
- Total time: $91 + 16g = 91 + 16 \cdot 10 = 251$.

12×12 computational grid: challenge



8 processors

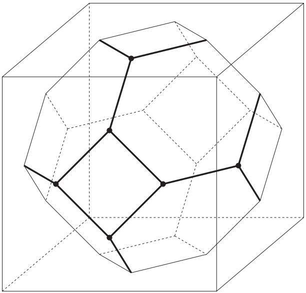
- Find a better solution than can be obtained manually, using ideas from both solutions shown. Current best known solution is 199 (Bas den Heijer 2006).

Three dimensions

- If a processor has a cubic block of $N = k^3/p$ points, about $\frac{6k^2}{p^{2/3}} = 6N^{2/3}$ are boundary points. In 2D, only $4N^{1/2}$.
- If a processor has a $10 \times 10 \times 10$ block, 488 points are on the boundary. About half!
- Thus, communication is important in 3D.
- Based on the surface-to-volume ratio of a 3D digital diamond, we can aim for a reduction by a factor $\sqrt{3} \approx 1.73$ in communication cost.
- The prime application of diamond-shaped distributions will most likely be in 3D.

Universiteit Utrecht

Basic cell for 3D



- Basic cell: grid points in a truncated octahedron.
- For load balancing, take care with the boundaries.
- What You See, Is What You Get (WYSIWYG):
  4 hexagons and 3 squares visible at the front are included.
  Also 12 edges, 6 vertices.
- Gain factor of 1.68 achieved for $p = 2q^3$.

Comparing partitioning methods in 2D and 3D

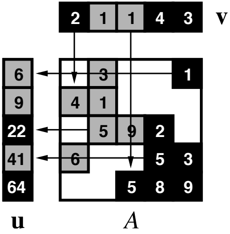
Grid	p	Rectangular	Mondriaan	Diamond
1024×1024	2	1024	1024	2046
	4	1024	1240	2048
	8	1280	1378	1026
	16	1024	1044	1024
	32	768	766	514
	64	512	548	512
	128	384	395	258
$64 \times 64 \times 64$	16	4096	2836	2402
	128	1024	829	626

Communication cost (in g) for a Laplacian operation on a grid.
Mondriaan with $\epsilon = 10\%$.

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

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

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



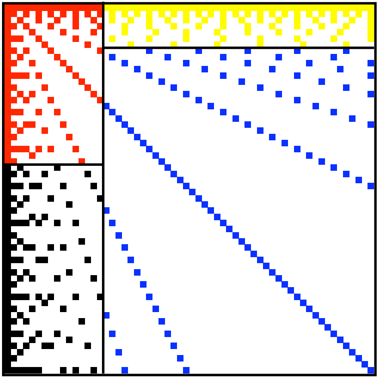
$p = 2$

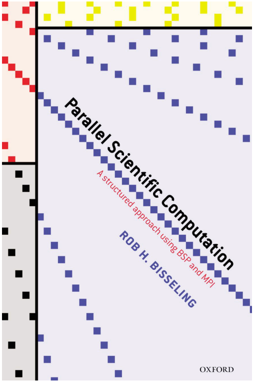
4 supersteps: communicate, compute, communicate, compute

Universiteit Utrecht

Divide evenly over 4 processors

Matrix prime60





- Mondriaan block partitioning of 60×60 matrix `prime60` with 462 nonzeros, for $p = 4$
- $a_{ij} \neq 0 \iff i|j \text{ or } j|i \qquad (1 \leq i, j \leq 60)$

Avoid communication completely, if you can

All nonzeros in a row or column have the same colour.

Permute the matrix rows/columns

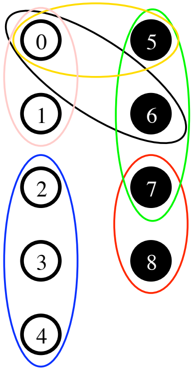
First the green rows/columns, then the blue ones.

Combinatorial problem: sparse matrix partitioning

Problem: Split the set of nonzeros A of the matrix into p subsets, $A_0, A_1, \ldots, A_{p-1}$, minimising the communication volume $V(A_0, A_1, \ldots, A_{p-1})$ under the load imbalance constraint

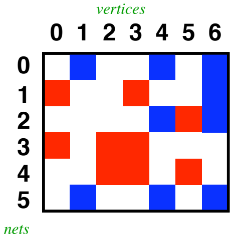
$$nz(A_i) \leq \frac{nz(A)}{p}(1+\epsilon), \quad 0 \leq i < p.$$

The hypergraph connection



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

1D matrix partitioning using hypergraphs



- Hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N}) \Rightarrow$ exact communication volume in sparse matrix–vector multiplication.
- Columns $\equiv$ Vertices: $0, 1, 2, 3, 4, 5, 6$.
  Rows $\equiv$ Hyperedges (nets, subsets of $\mathcal{V}$):

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

$(\lambda - 1)$-metric for hypergraph partitioning

- 138×138 symmetric matrix `bcsstk22`, $nz = 696$, $p = 8$
- Reordered to Bordered Block Diagonal (BBD) form
- Split of row i over λ_i processors causes a communication volume of $\lambda_i - 1$ data words

Cut-net metric for hypergraph partitioning

- Row split has unit cost, irrespective of λ_i

Mondriaan 2D matrix partitioning

- $p = 4$, $\epsilon = 0.2$, global non-permuted view

Fine-grain 2D matrix partitioning

- Each individual nonzero is a vertex in the hypergraph
  Çatalyürek and Aykanat, 2001.

Mondriaan 2.0, Released July 14, 2008

- New algorithms for vector partitioning.
- Much faster, by a factor of 10 compared to version 1.0.
- 10% better quality of the matrix partitioning.
- Inclusion of fine-grain partitioning method
- Inclusion of hybrid between original Mondriaan and fine-grain methods.
- Can also handle $p \neq 2^q$.

Matrix `lns3937` (Navier–Stokes, fluid flow)

Splitting the 3937×3937 sparse matrix `lns3937` into 5 parts.

Recursive, adaptive bipartitioning algorithm

MatrixPartition(A, p, ϵ)

input: p = number of processors, $p = 2^q$

ϵ = allowed load imbalance, $\epsilon > 0$.

output: p-way partitioning of A with imbalance $\leq \epsilon$.

if $p > 1$ **then**

$q := \log_2 p$;

$(A_0^{\mathrm{r}}, A_1^{\mathrm{r}}) := h(A, \mathrm{row}, \epsilon/q)$; hypergraph splitting

$(A_0^{\mathrm{c}}, A_1^{\mathrm{c}}) := h(A, \mathrm{col}, \epsilon/q)$;

$(A_0^{\mathrm{f}}, A_1^{\mathrm{f}}) := h(A, \mathrm{fine}, \epsilon/q)$;

(A_0, A_1) := best of $(A_0^{\mathrm{r}}, A_1^{\mathrm{r}})$, $(A_0^{\mathrm{c}}, A_1^{\mathrm{c}})$, $(A_0^{\mathrm{f}}, A_1^{\mathrm{f}})$;

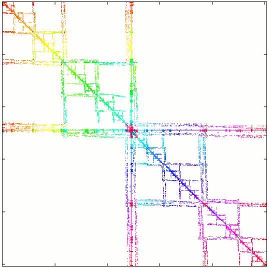
$maxnz := \frac{nz(A)}{p}(1+\epsilon)$;

$\epsilon_0 := \frac{maxnz}{nz(A_0)} \cdot \frac{p}{2} - 1$; MatrixPartition$(A_0, p/2, \epsilon_0)$;

$\epsilon_1 := \frac{maxnz}{nz(A_1)} \cdot \frac{p}{2} - 1$; MatrixPartition$(A_1, p/2, \epsilon_1)$;

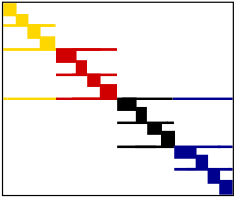
else output A;

Mondriaan 3.0, Released July 27, 2010



- Ordering of matrices to SBD and BBD structure: cut rows are placed in the middle, and at the end, respectively.
- Visualisation through Matlab interface, MondriaanPlot, and MondriaanMovie
- Library-callable, so you can link it to your own program
- Hypergraph metrics: $\lambda - 1$ for parallelism, and cut-net for other applications
- Interface to PaToH hypergraph partitioner

Separated block-diagonal (SBD) structure

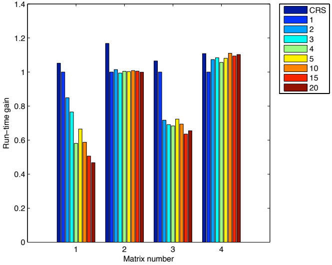


- SBD structure is obtained by recursively partitioning the columns of a sparse matrix, each time moving the cut (mixed) rows to the middle. Columns are permuted accordingly.
- The cut rows are sparse and serve as a gentle cache transition between accesses to two different vector parts.
- Mondriaan is used in one-dimensional mode, splitting only in the column direction.

Partition the columns till the end, $p = n = 59$

- The recursive, fractal-like nature makes the ordering method work, irrespective of the actual cache characteristics (e.g. sizes of L1, L2, L3 cache).
- The ordering is cache-oblivious.

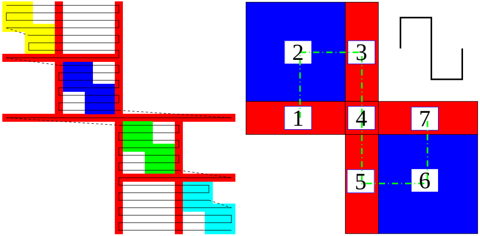
Wall clock timings of SpMV on Huygens



Splitting into 1–20 parts

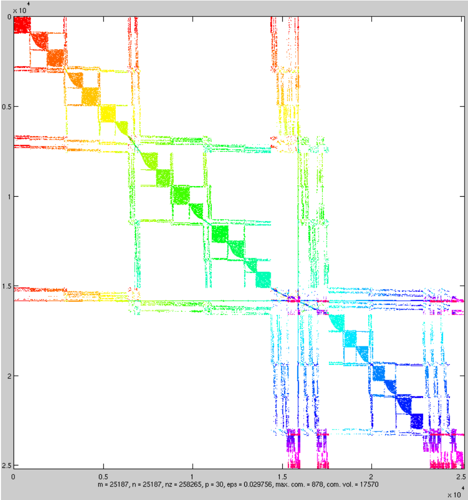
- Experiments on 1 core of the dual-core 4.7 GHz Power6+ processor of the Dutch national supercomputer Huygens.
- 64 kB L1 cache, 4 MB L2, 32 MB L3.
- Test matrices: 1. `stanford`; 2. `stanford_berkeley`; 3. `wikipedia-20051105`; 4. `cage14`

Doubly Separated block-diagonal (DSBD) structure



- Zig-zag Compressed Row Storage (ZZ-CRS) data structure must be adapted for 2D matrix reordering into DSBD form.
- Split the matrix into blocks, each with its own ZZ-CRS data structure.
- The blocks are processed in ZZ-CCS order.
- This gives a factor of 2.7 speedup on the sparse matrix `wikipedia-20060925` due to better cache-use.

Screenshot of Matlab interface



- Matrix `rhpentium`, split over 30 processors

Where meshes meet matrices

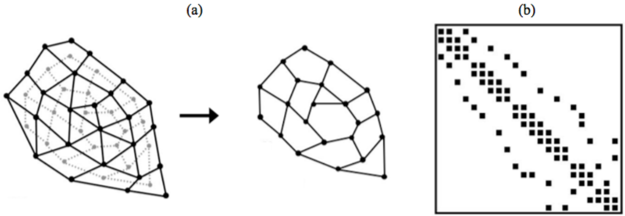


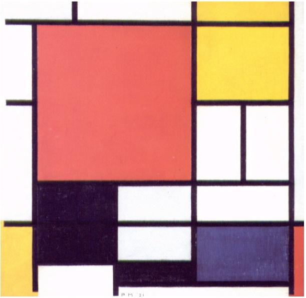
Fig. 8 Example of an unstructured grid with its associated dual graph and partitioning process (a) and the related sparse matrix (b).

- Unstructured grid and its sparse matrix
- Source: N. Gourdain et al. 'High performance Parallel Computing of Flows in Complex Geometries. Part 1: Methods' *Computational Science and Discovery* 2009.

Apply Mondriaan matrix partitioning

- Use Mondriaan in 1D mode, not in full 2D mode.
- Advantage: no need to change data structure, while still giving almost the same communication volume (for FEM matrices).
- Advantage: hypergraph partitioning leads to less ghost cells, and less communication, especially in 3D.

Apply Mondriaan matrix partitioning



- Advantage: Mondriaan is open-source, can be changed by yourself or by us for your needs, and is an ongoing research project with much attention for software engineering.
- Disadvantage: hypergraph partioner Mondriaan itself takes more time and memory than graph partitioners (such as Scotch or Metis).

Conclusions on regular meshes

- To achieve a good partitioning with a low surface-to-volume ratio, all dimensions must be cut. For regular grids in 2D, this gives square subdomains; in 3D, cubic.
- In 2D, an even better method is to use digital diamonds. This basic cell tiles a rectangular domain in a straightforward manner. Best performance is obtained for $p = 2q^2$.
- In 3D, the best method is to use truncated octahedra with WYSIWYG tie breaking at the boundaries. Best performance is obtained for $p = 2q^3$.

Conclusions on irregular meshes

- For unstructured grids, the same gains can be obtained by using hypergraph partitioning, which minimises the exact amount of communication and number of ghost cells.
- This saves a factor of $\sqrt{3}$ in communication and ghost memory usage compared to standard graph partitioning with the edge-cut metric.

Future work

- Converter for reading meshes directly, translating them to matrices, partitioning them, and writing the result back as a mesh.
- Extended Matrix Market format (EMM) to handle distributed and reordered sparse matrices in an easy way.
- PMondriaan (Parallel Mondriaan) for a multicore computer.
- We welcome use of the Mondriaan partitioner in applications of Deltares and its partners and are happy to collaborate on this.