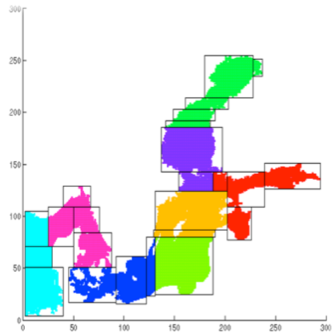
UPPSALA UNIVERSITET

Data partitioning and load balancing

Jarmo Rantakokko

2014-02-06



UPPSALA UNIVERSITET

Case study:

Consider 7 independent tasks with estimated work loads: 5, 2, 3, 4, 5, 2, 10

Assign the tasks to 2 processors and compute the work load for each processor.

(How would you do with 10000 tasks and 100 processors?)

Institutionen för informationsteknologi | www.it.uu.se | Jarmo Rantakokko

Linear partitioning (array of N tasks)

1.Static partitioning

set n=floor(N/p), m=mod(N/p)
Assign the m first proc n+1 tasks
and the other n tasks
Ex. N=10, p=4 => n=2, m=2
tasks: 3,3,2,2

1	1	1	2	2	2	3	3	4	4

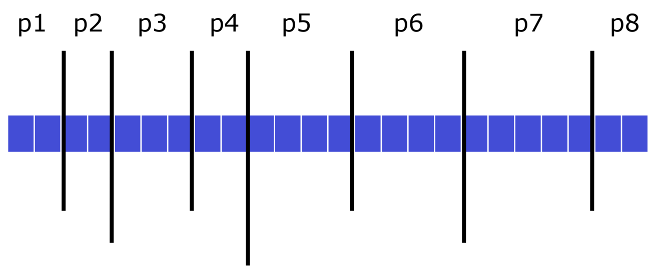
2. Cyclic partitioning

Assign tasks cyclicly (Round Robin)
proc(i) <= task(i+n*p), n=1,2,…
LU-factorization, Gram-Schmidt, etc

1	2	3	4	1	2	3	4	1	2

3. Recursive bisection

Divide the array into two halves with equal work loads. Proceed recursively with each half until we have p parts.

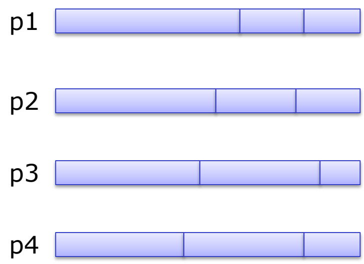


Can easily be extended to arbitrary number of processors by assigning the work load proportional to the processors in each cut.

UPPSALA UNIVERSITET

4. Bin-packing

Assign tasks in size order, largest first, to the processor with least work in each step.



UPPSALA UNIVERSITET

5. Greedy (variant of Bin-packing)

1. Set an upper limit C for the work load in each bin.
2. For each bin grab as many tasks as possible (load ≤ C).
3. If we could not assign all tasks, increase C otherwise decrease C.

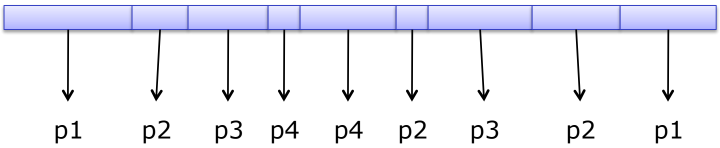
Repeat 1-3 searching for the smallest possible C for which it is possible to assign all tasks.

(Lower complexity than for Bin-packing)

UPPSALA UNIVERSITET

6. Dynamic

Set up a task-queue and assign tasks from the queue to the processors as soon as they are ready processing a previous task.



(See example 7.4 and 7.6 for Pthreads implementation, OpenMP supports schedule(dynamic) and task queues.)

UPPSALA UNIVERSITET

Case study: Tasks: 5, 2, 3, 4, 5, 2, 10

Static: 5+2+3+4=14,
5+2+10=17

Cyclic: 5+3+5+10=23
2+4+2=8

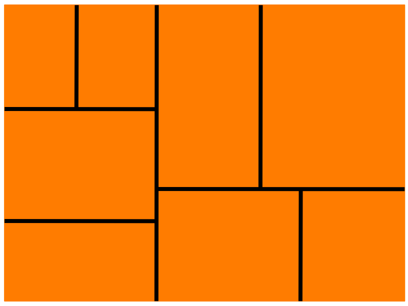
Bisect: 5+2+3+4=14
5+2+10=17

Bin-pack: 10+4+2=16
5+5+3+2=15

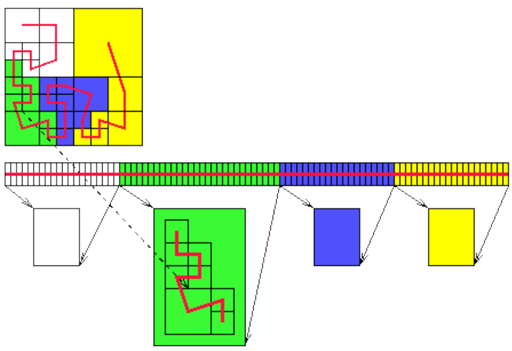
Dynamic: 5+4+2=11
2+3+5+10=20

Note: The algorithms can easily be extended to multi-dimensional arrays (for partitioning grids or matrices).

Ex Recursive bisection:



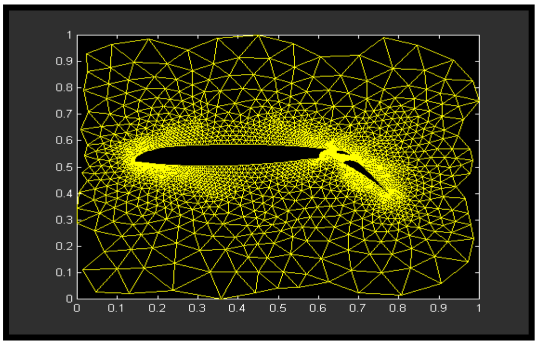
Or by mapping to a one dimensional array with Space Filling Curves (SFC), e.g., Hilbert or Morton curve.



SFCs has a locality preserving property giving geometrically collected partitions.

Graph partitioning methods

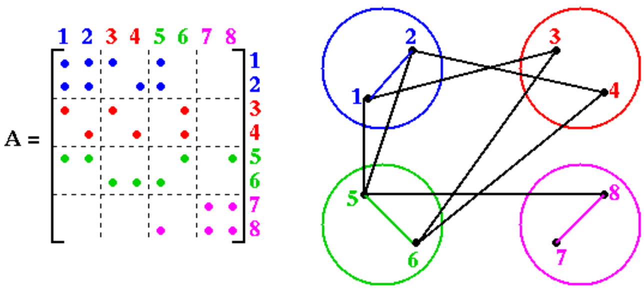
Assume that we have data with dependencies to nearest neighbors. These dependencies can then be represented with a graph, e.g., consider a FEM-mesh or sparse matrix mult.



Nodes - data, Edges - data dependencies

We want to partition the data equally (load balance) with minimal edge-cut (minimizing communication).

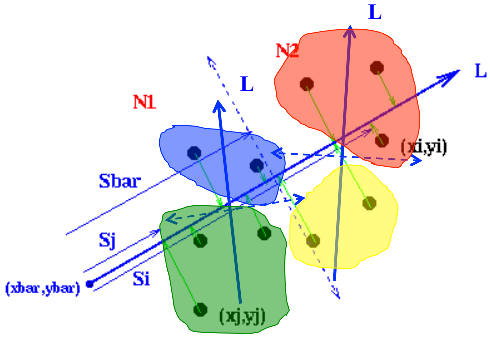
Consider Matrix-Vector Multiplication



Is there a better partitioning than the straight forward with less edge-cut?

1. *Recursive Inertial partitioning*

Find the axis of minimal inertia and divide into two halves perpendicular to the axis.



Orthogonal cut gives “best” edge-cut. Proceed in the same way with each half.

2. *Recursive Spectral Bisection*

Assume that we have a graph with nodes V_I and edge weights W_{IJ} between nodes V_I and V_J. The graph can be represented with the Laplacian matrix L.

$$L_{IJ} = \begin{cases} -W_{IJ} & \text{if an edge } V_I \text{ to } V_J \\ \Sigma_K(W_{IK}) & \text{if } I=J \\ 0 & \text{otherwise} \end{cases}$$

Theorem by Fiedler:

The second smallest eigenvalue to L satisfies

$$\lambda_2 = \min_{|x|\neq 0} \Sigma_{i,k} W_{ik}(x_i-x_k)^2/\Sigma_i x_i^2$$

and the minimum is attained for the corresponding eigenvector $x=[x_1,x_2,\ldots]$.

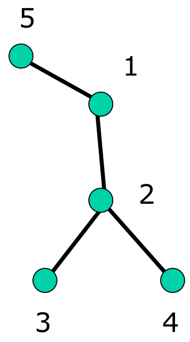
Note, to attain minimum x_i and x_k must be close if edge weight W_{ik} is large.

$\Rightarrow$ The second eigenvector gives geometrical information about the graph. Use this for partitioning of the graph.

Algorithm RSB:

1. Set up the Laplacian matrix.
2. Compute the eigenvector corresponding to the second smallest eigenvalue.
3. Sort the nodes according to the elements in eigenvector, then nodes with heavy edge weights will be close.
4. Divide the nodes in two halves according to the eigenvector.
5. Repeat 1-4 recursively with the two sets.

Example:



Laplacian matrix =

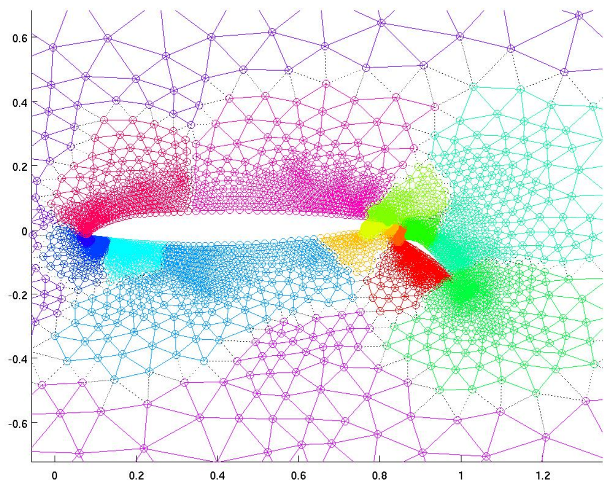
2	-1	0	0	-1
-1	3	-1	-1	0
0	-1	1	0	0
0	-1	0	1	0
-1	0	0	0	1

Eigenvalues =

1.0000	**0.5188**	0.0000	2.3111	4.1701

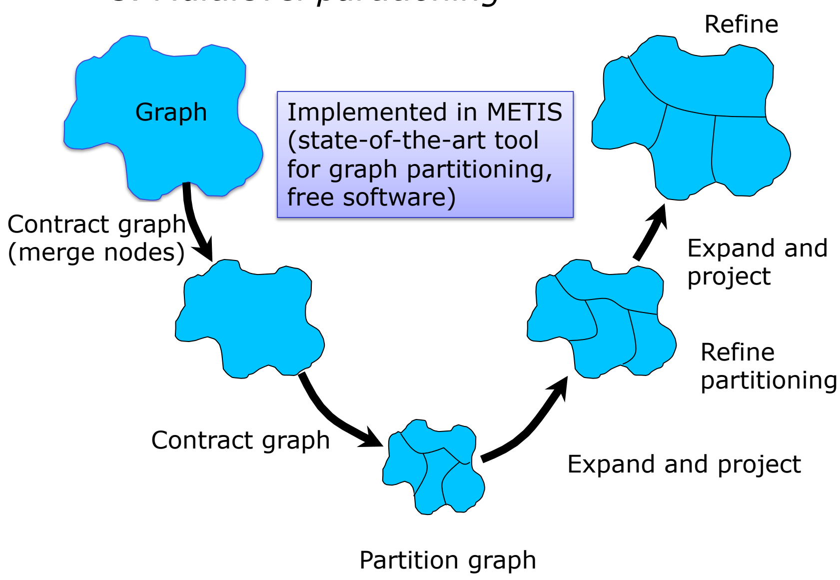
Eigenvectors =

0	**-0.3380**	-0.4472	0.7031	-0.4375
-0.0000	**0.2018**	-0.4472	0.3175	0.8115
0.7071	**0.4193**	-0.4472	-0.2422	-0.2560
-0.7071	**0.4193**	-0.4472	-0.2422	-0.2560
0.0000	**-0.7024**	-0.4472	-0.5362	0.1380



Example: Partitioning a FEM mesh
(Note, the graph can be *very* large)

3. Multilevel partitioning

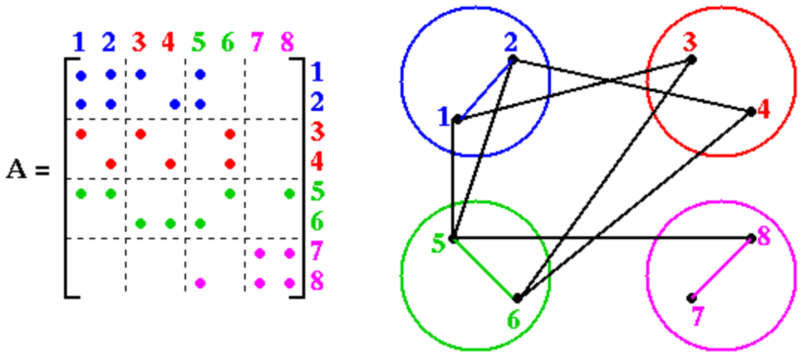


Bandwidth minimization

FEM → Ax=b where A is sparse (<1% non-zero) and very large (~10^6 rows).

Ax=b is usually solved with an iterative method based on a sequence of Matrix-vector multiplications (e.g. Conjugate gradient method). Need to parallelize the MxV row-wise. Block off-diagonal elements requires communication.

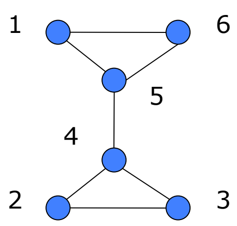
UPPSALA UNIVERSITET



Minimize bandwidth => minimize communication and improve cache locality!

UPPSALA UNIVERSITET

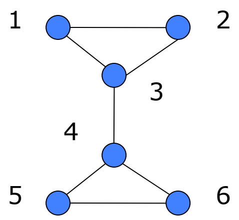
The bandwidth depends on the numbering of the nodes, consider the following graph and the corresponding matrix:



X				X	X
	X	X	X		
	X	X	X		
	X	X	X	X	
X			X	X	X
X				X	X

Bandwidth = 6

Then re-number the nodes (unknowns):



X	X	X			
X	X	X			
X	X	X	X		
		X	X	X	X
			X	X	X
			X	X	X

Bandwidth = 4

UPPSALA UNIVERSITET

Reverse Cuthill-McKee algorithm:

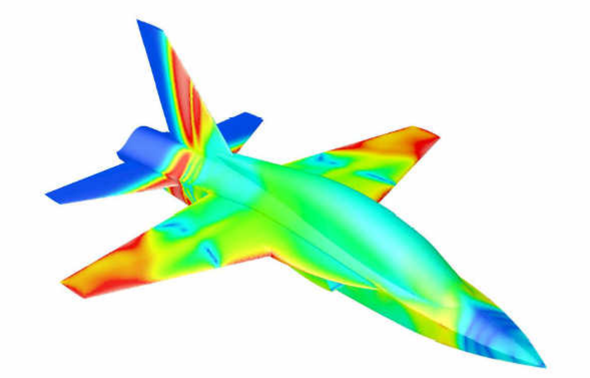
1. Find a root node, i.e., a node with a small number of edges => First node.
2. Number its neighbors, starting with fewest edges, then next fewest, etc.
3. Continue with lowest numbered node, with unnumbered neighbors, and number its neighbors as above.
4. When all nodes are numbered, reverse the ordering.

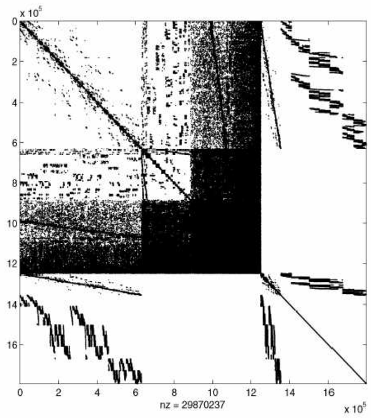
Implemented as p=symrcm(A) in Matlab, takes a matrix A and returns a permutation vector p minimizing the bandwith of A(p,p).

Institutionen för informationsteknologi | www.it.uu.se | Jarmo Rantakokko

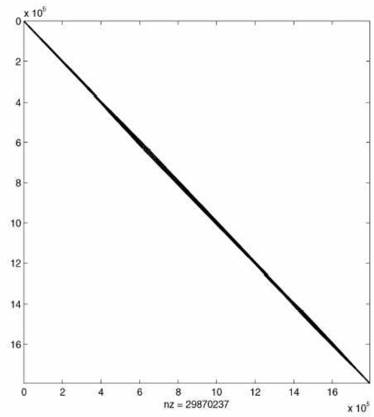
Real applications: GEMS-project

Maxwell's equations discretized with FEM-grid around a fighter jet => Ax=b with 1.8 million unknowns, solved with the CG method .





Original matrix



Bandwidth minimized

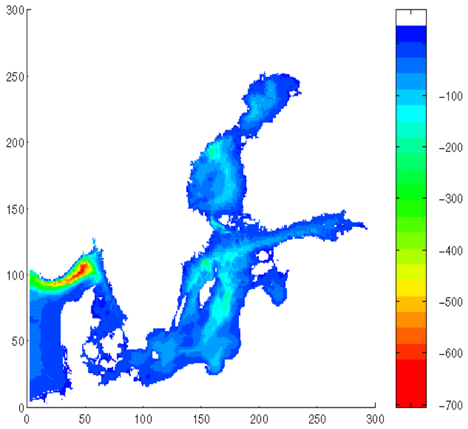
[Ref: H. Löf, J. Rantakokko, *Algorithmic Optimization of a Conjugate Gradient Solver on Shared memory systems*, International Journal of Parallel, Emergent and Distributed Systems, Vol 21, 2006.]

Ocean modeling, SMHI

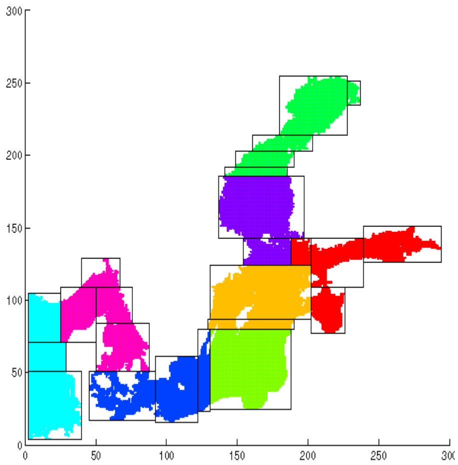
Baltic Sea, 300x300 structured grid varying sea depth

Need a block partition but have varying work load per grid point (zero for land points)

Minimize:
– load imbalance
– communication
– land points
– blocks



Hybrid block-structured & graph partitioning method:



1. Cover the domain with mxm blocks.
2. Remove blocks completely on land.
3. Shrink blocks to sea boundary.
4. Split blocks with large fraction of land and repeat 2-4.
5. Set up a graph for remaining blocks.
6. Compute Fiedler vector and sort.
7. Divide into two sets and split a block in division if necessary.
8. Proceed with steps 5-7 recursively.

[Ref: T. Wilhelmsson, J. Schüle, J. Rantakokko, L. Funkquist, *Increasing Resolution and Forecast Length with a Parallel Ocean Model, In: Operational Oceanography, Implementation at the European and Regional Scales, editor N.C. Fleming, Elsevier Oceanography Series, 2002.*]