Structured Data Partitioning

Fredrik Manne

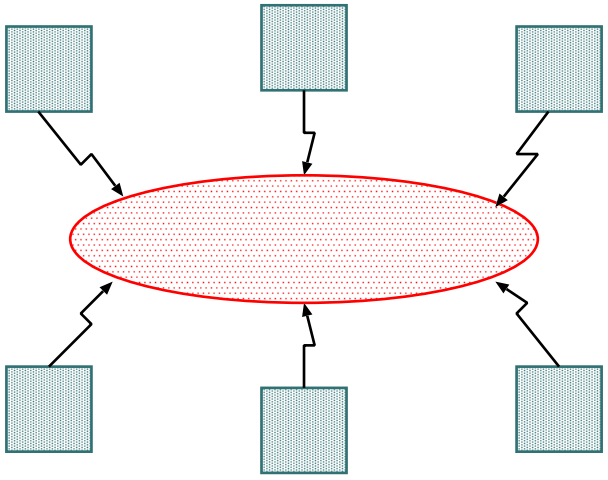
Department of Informatics
University of Bergen

Outline

- Introduction
- Different ways to partition data
- Structured partitioning
- 1D partitioning
- 2D partitioning
- Conclusion

Parallel Processing

Parallel Computer: Several processors connected together (hypercube, grid, etc.)



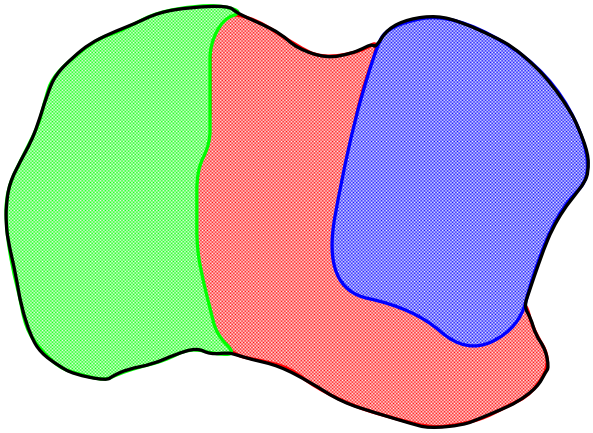
To parallelize a large task it must be split into (at least partly) independent tasks.

Data-parallelism: Same set of instructions on different parts of the data.

Control parallelism: Different instruction set on different processors.

Data model

- Data describes a "connected" physical area.
- Data locality - What happens in one place influences only its close vicinity.

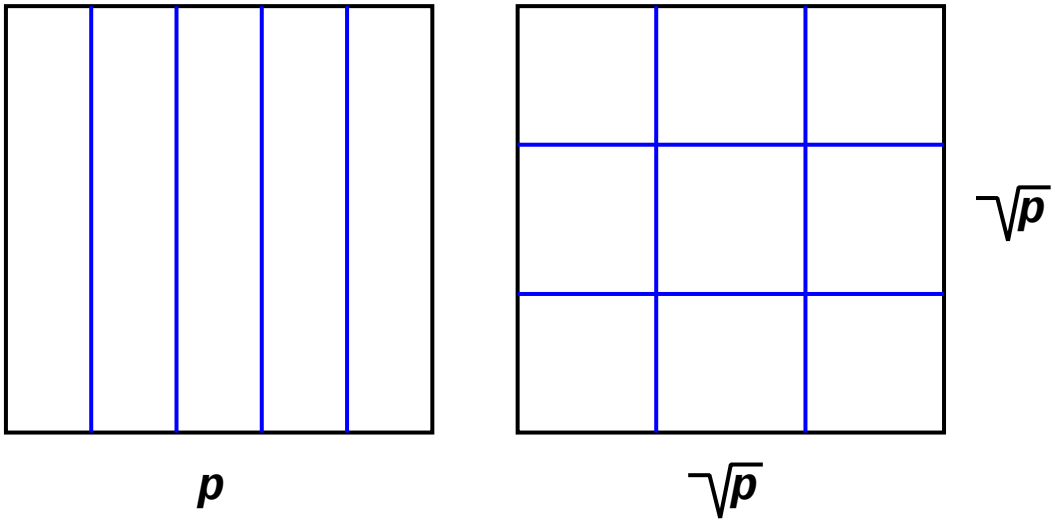


- Divide data to perform tasks in parallel.
- Want even load balance with as little communication as possible.

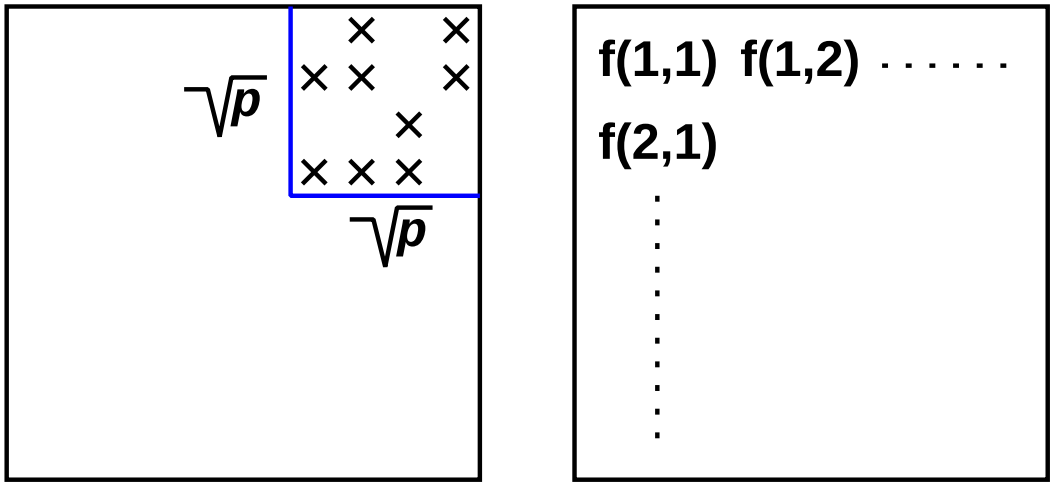
Data dependencies + Partitioning = Parallel Algorithm

Regular Data Partitioning

Full regular systems can easily be divided into equally sized parts.



Irregular systems without data locality can be partitioned in several ways.



Will be looking at problems with irregular load where data locality is of importance.

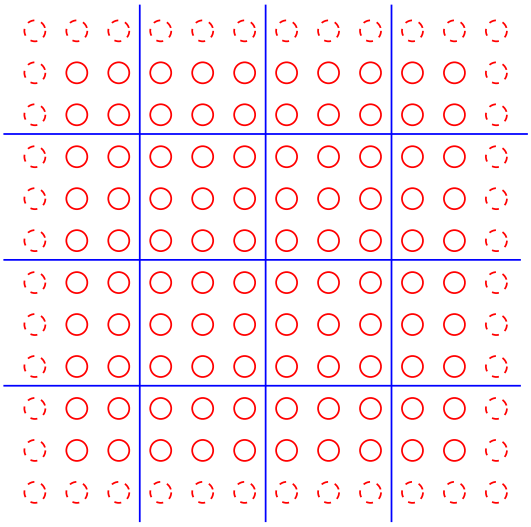
High Performance Fortran

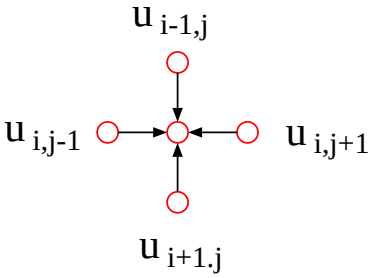
High Performance Fortran has support both for block and cyclic partitioning.

```
real u(0:n+1,0:n+1)

!HPF$ PROCESSORS pros(4,4)
!HPF$ DISTRIBUTE u(BLOCK,BLOCK) ONTO pros

  u(1:n,1:n) = (u(0:n-1,1:n)+u(2:n+1,1:n)+      &
                u(1:n,0:n-1)+u(1:n,2:n+1))/4
```

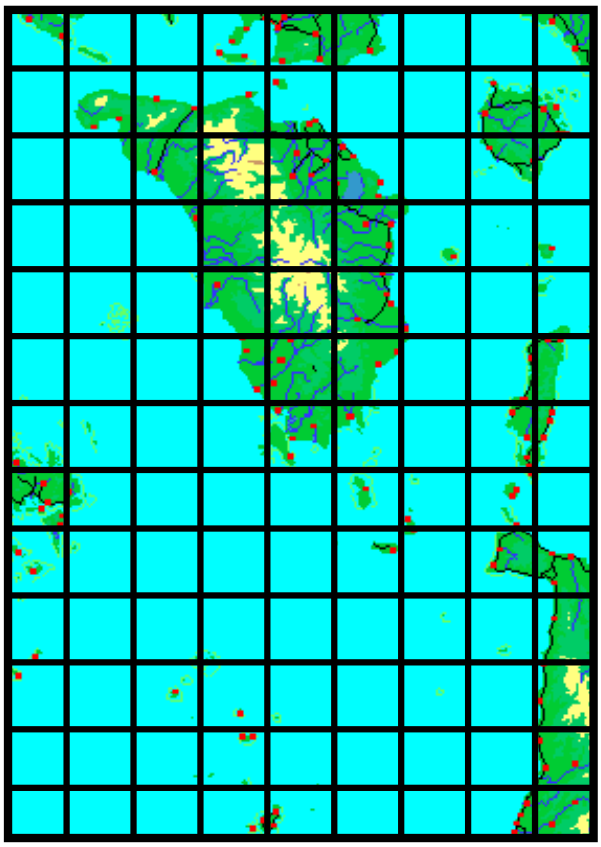




Ocean modeling

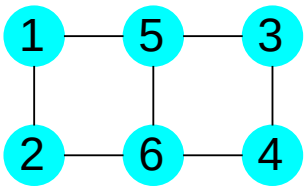
Miami Isopycnic Coordinate Ocean Model (MICOM)

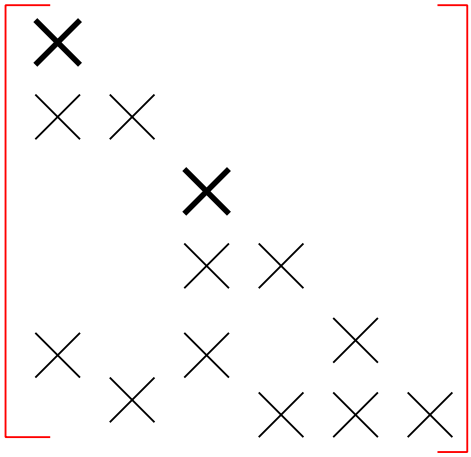
- Simulates currents in the sea
- Computations are very parallel
- Divides data into many small rectangles
- Areas with water are spread on the fly among the processors

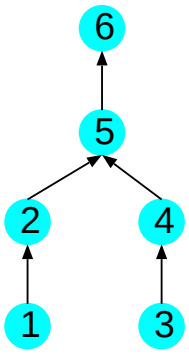


Sparse Matrix Computations

Wish to solve $Ax = b$. A is sparse, symmetric, and positive definite.
Direct method: Parallel Cholesky-factorization



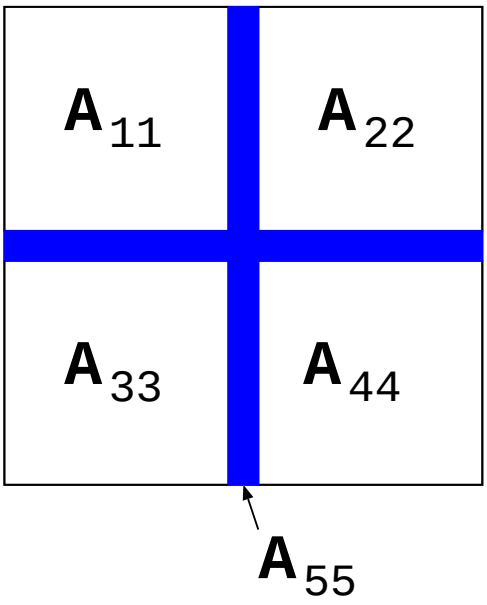




- Vertices 5 og 6 make up a vertex separator.
- Can solve for x_1 and x_3 simultaneously
- Recursive partitioning of the graph gives the needed parallelism

Iterative methods

Domain decomposition



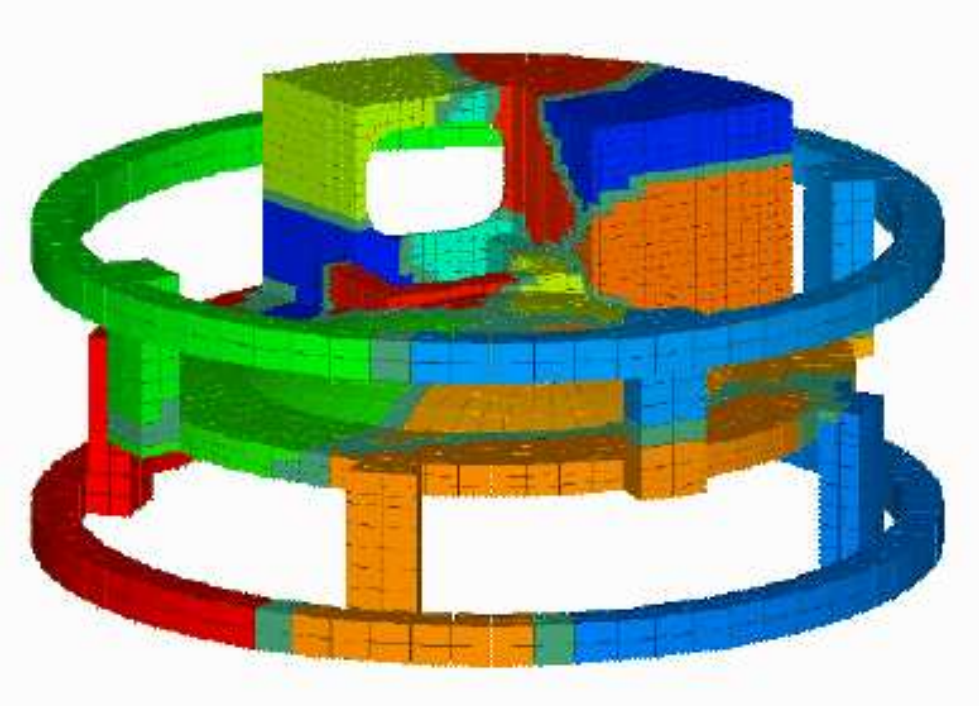
$$\begin{bmatrix} A_{11} & & & & A_{51}^T \\ 0 & A_{22} & & & A_{52}^T \\ 0 & 0 & A_{33} & & A_{53}^T \\ 0 & 0 & 0 & A_{44} & A_{54}^T \\ A_{51} & A_{52} & A_{53} & A_{54} & A_{55} \end{bmatrix}$$

- Divide the system
- Eliminate $A_{5,1}$ through $A_{5,4}$
- Solve for x_5 using iterative (parallel) method
- Perform back substitution and solve for x_1 through x_4 using direct method

Graph Partitioning

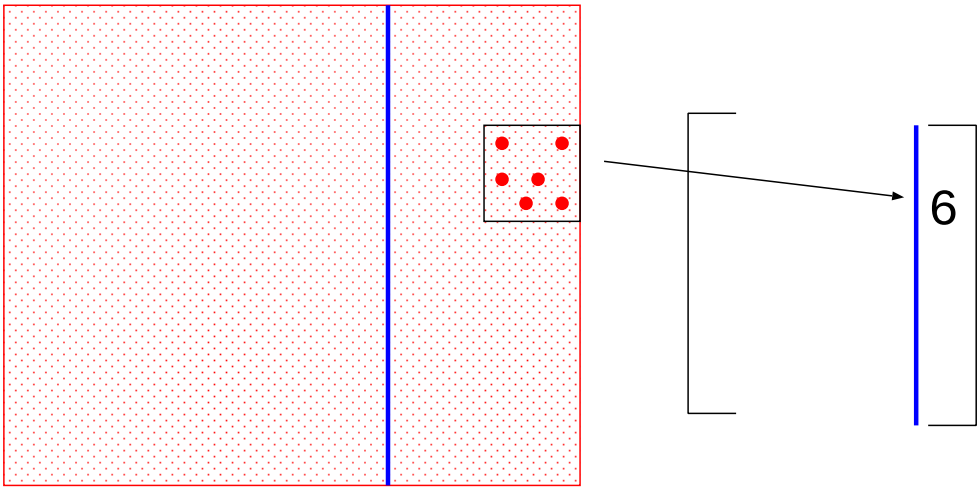
Both direct and iterative methods requires partitioning of a graph into smaller parts of equal size.

- Gives a good partitioning of complex systems
- Even load, good data locality



- Requires separate (complex) graph partitioning software
- The programmer must exploit the parallelism herself
- Must keep track of communication

Structured Data Partitioning

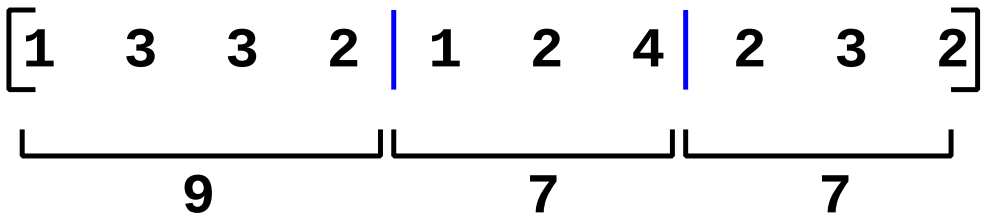


- Each data object has a coordinate in a n dimensional space (n=1,2 or 3)
- The data space is partitioned using vertical and horizontal lines.
- The goal is to get an even load (largest load as small as possible).
- Can aggregate data items into a table
- The cutting pattern determines the communication pattern

1D-Partitioning

Divide a sequence of n positive numbers into p connected components such that the most expensive interval is as cheap as possible.

$n = 10$
$p = 3$



- Cost is (usually) measured using summation
- Can include cost of nearest neighbor communication
- Supported in High Performance Fortran 2
- Several exact algorithms

Optimal 1D-partitioning, History

Time complexity	**Author**
$O(n^3p)$	Bokhari
$O(p^3n)$	Becker, Perl and Schach
$O(n^2p)$	Nicol and Hallaron
$O(n^2p)$	Anily and Federgruen
$O(n^2p)$	Hansen and Lih
$O(np^2)$	Perl and Vishkin
$O(np^2)$	Pinar and Aykanat
$O(np\log n)$	Iqbal and Bokhari
$O(p(n-p)\log p)$	Manne and Sørevik
$O(np)$	Choi and Narahari
$O(p(n-p))$	Olstad and Manne
$O(n+p^2\log^2 n)$	Nicol
$O(n+p^2\log^2 n)$	Charikar, Chekuri, and Motwani
$O(n+p^{1+\epsilon})$	Han, Narahari, and Choi
$O(n\log n)$	Khanna, Muthukrishnan, and Skiena
$O(n)$	Frederickson

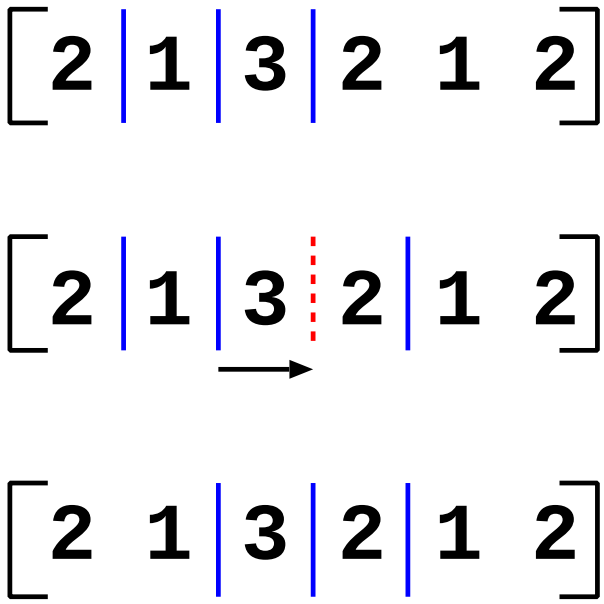
1D-partitioning, Chronological

Time complexity	**Year**
$O(p^3 n)$	1982
$O(np^2)$	1985
$O(n^3 p)$	1988
$O(n^2 p)$	1991
$O(n^2 p)$	1991
$O(np)$	1991
$O(n + p^2 \log^2 n)$	1991
$O(n)$	1991
$O(n^2 p)$	1992
$O(n + p^{1+\epsilon})$	1992
$O(np \log n)$	1995
$O(p(n-p) \log p)$	1995
$O(p(n-p))$	1995
$O(n + p^2 \log^2 n)$	1996
$O(np^2)$	1997
$O(n \log n)$	1997

Shift Algorithms

Place the dividers to the left of their optimal placement.

Repeat until 1. intervall is the most expensive:
- Find most expensive interval i
- Shift i's left divider one place to the right



- Will generate the optimal solution at some stage.
- Time complexity $O(p(n-p)\log p)$

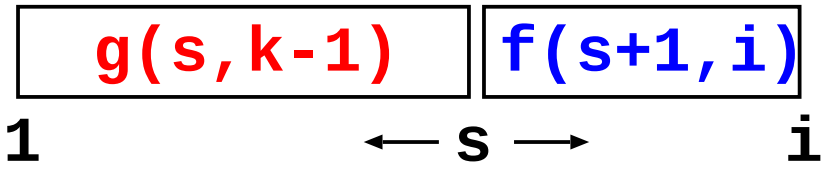
Dynamic Programing

Solve smal problems in order to build larger solutions.
$g(i,k)$ = Optimal cost when partitioning $[a_1, a_2, ..., a_i]$ into k intervals.

Want to find $g(n,p)$.

Initial values:
$g(i,1) = \Sigma_{j=1}^{i} a_j$

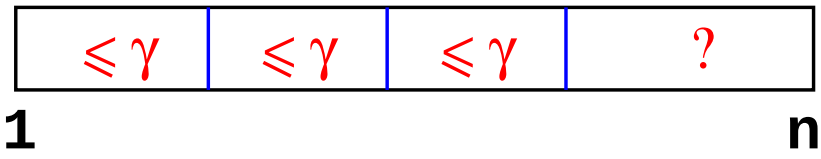
Recursion:
$g(i,k) = \min_s\{\max\{g(s,k-1), f(s+1,i)\}\}$



- Time complexity: $O((n-p)p)$

Probe Algorithms

$P(\gamma, p)$ = True if $[a_1, a_2, ..., a_n]$ can be partitioned into p intervals of cost $\leq \gamma$.



Greedy algorithm calculates $P(\gamma, p)$ in $O(n)$ time.

- Observation: In the optimal solution some interval must the most expensive.
- Search for the most expensive interval such that $P(\gamma, p)$ is true.
- Time complexity: $O(n)$ (Asymptotically)

Experiments 1D

Can bound the search area for an optimal solution:

W = Total weight of all the elements

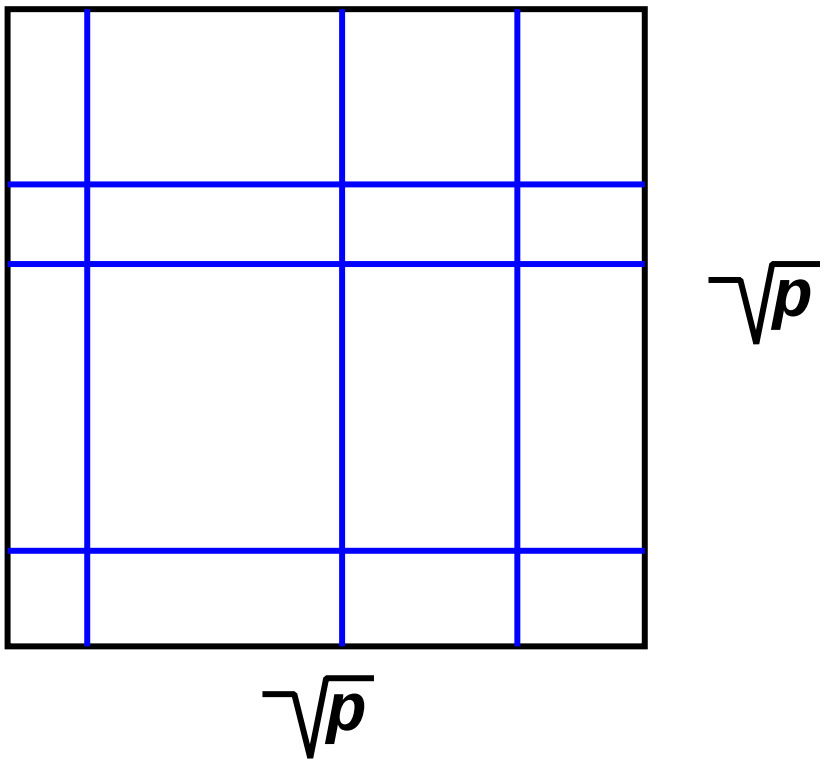
$$\frac{W}{p} \leq Opt \leq \frac{2W}{p}$$

- (Pinar and Aykanat 97) Shift algorithm with smart initialization
  runs 50 to 100 times faster than $O(n + p^2 \log^2 n)$ probe algorithm.

- Shift algorithm is also approximately 2 times faster than dynamic programming algorithm.
- Still unclear which algorithm to choose
  (Have master student working on this...)

2D: Generalized block partitioning

- Perform 1D partitioning in both dimensions
- Gives variable block size
- Each processor still has 4 neighbors

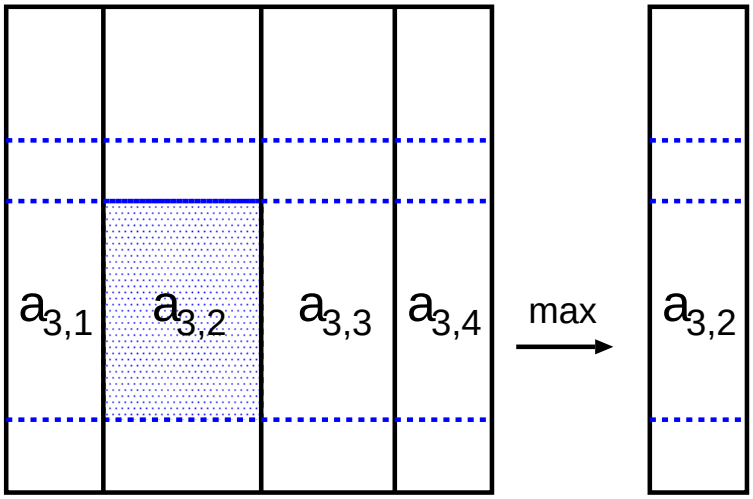


- More even load balance
- Can give rise to more communication than ordinary block partitioning
- Supported in High Performance Fortran 2

Generalized block partitioning, Theory

- 2D problem is NP-hard (Grigni and Manne 96)

- Heuristic (Nicol 94): Keep the vertical dividers fixed
  Find optimal placement of the horizontal dividers using a 1D algorithm. Repeat iteratively with vertical dividers.



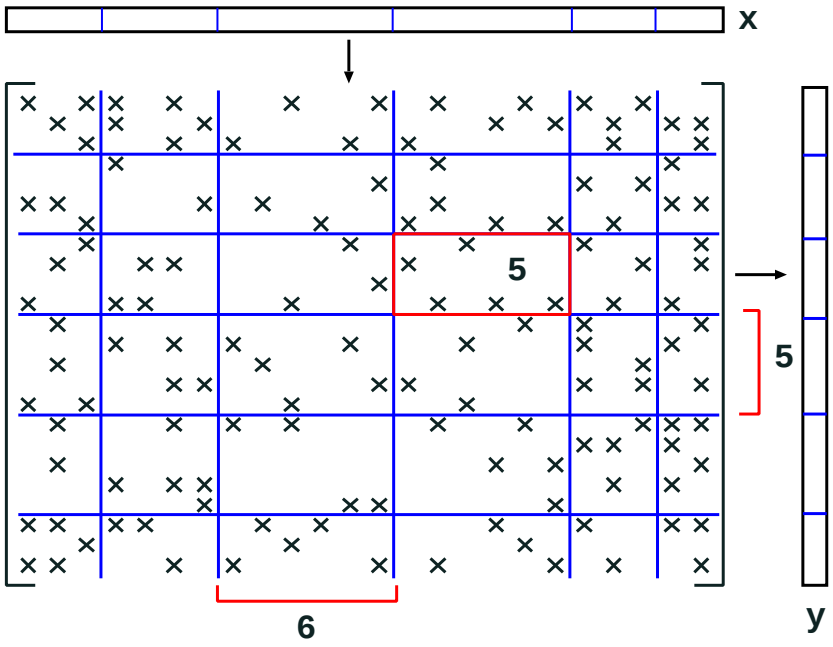
  - Is $O(p^{1/4})$ times optimal after at most three iterations (Aspvall, Halldorsson, Manne 97).
  - Experimental results gives solutions < 2 times optimal (Manne and Sørevik 96)

- New heuristic (Khanna et.al): Gives solution which is $O(1)$ times optimal! But
  - The constant has three digits
  - The time complexity is $O(n^4)$

Sparse Matrix-Vector multiplication

(Manne 93) Calculate $y = Ax$ where A is sparse using a grid connected parallel computer (Maspar MP2).

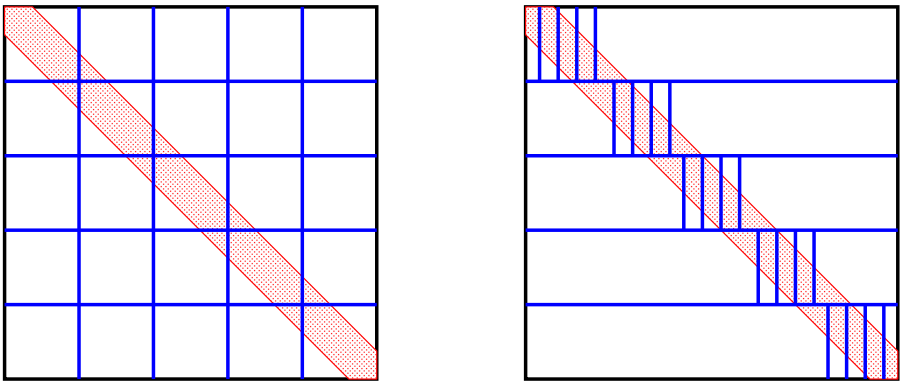
- Perform a random permutation of A in order to get an even load.
- Find a generalized block partitioning



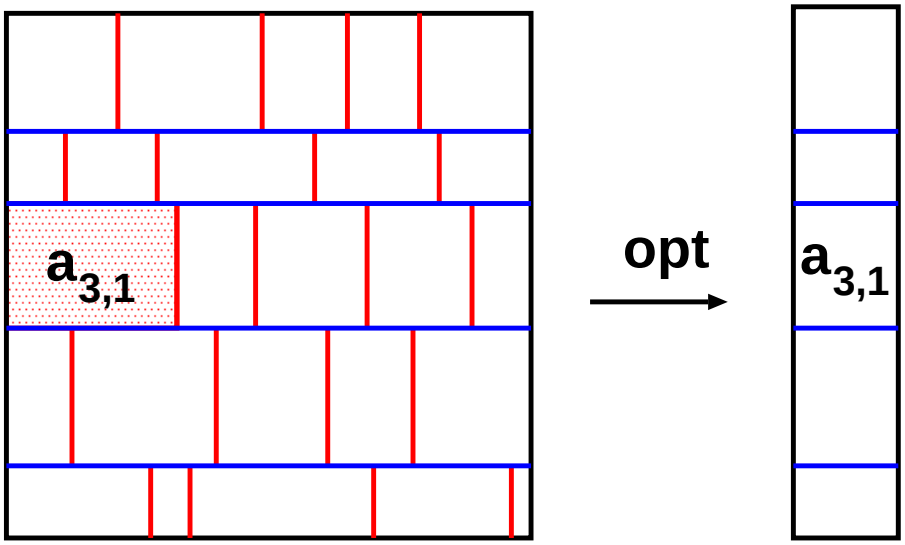
Time is given by
α |max column| + β |max block| + γ |max row|

Gave performance increase of up to 40%.

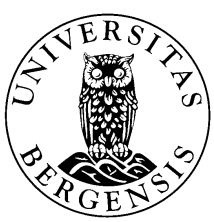
2D: Semi-Generalized Block Partitioning



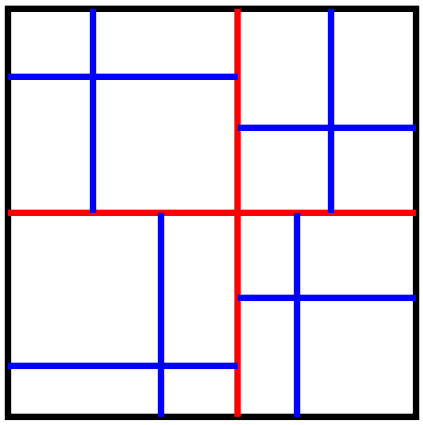
- More even load than Generalized block partitioning
- Can have as many as $\sqrt{p}$ neighbors north and south



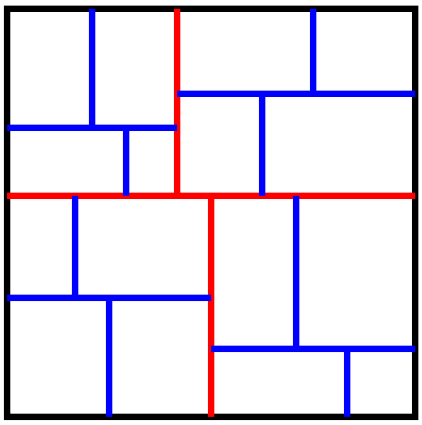
Calculates optimal partitioning using 1D algorithm.



2D: Recursive partitioning



Quad-tree partitioning



Binary recursive partitioning

Quad-tree partitioning = Recursive Generalized block partitioning
($p = 4$)

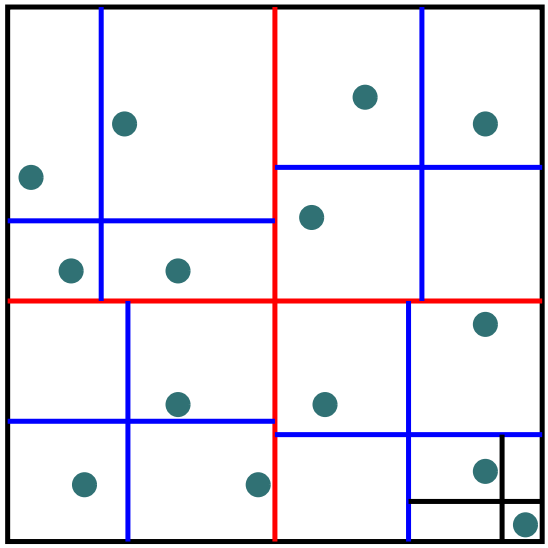
Binary recursive partitioning = Recursive Semi-generalized block
partitioning ($p = 4$)

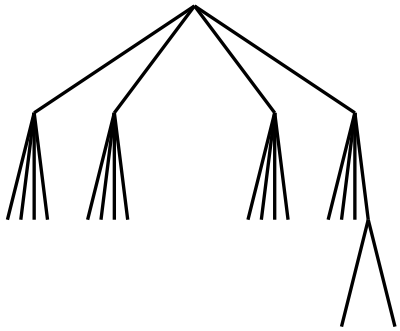
- Difficult to calculate optimal partitioning
- Never backtrack when performing partitioning
- Can stop partitioning when an area is "small" enough
- Each block can have several neighbors

N-body simulations

Simulate movement of objects that influence each other

- Planets
- Particles



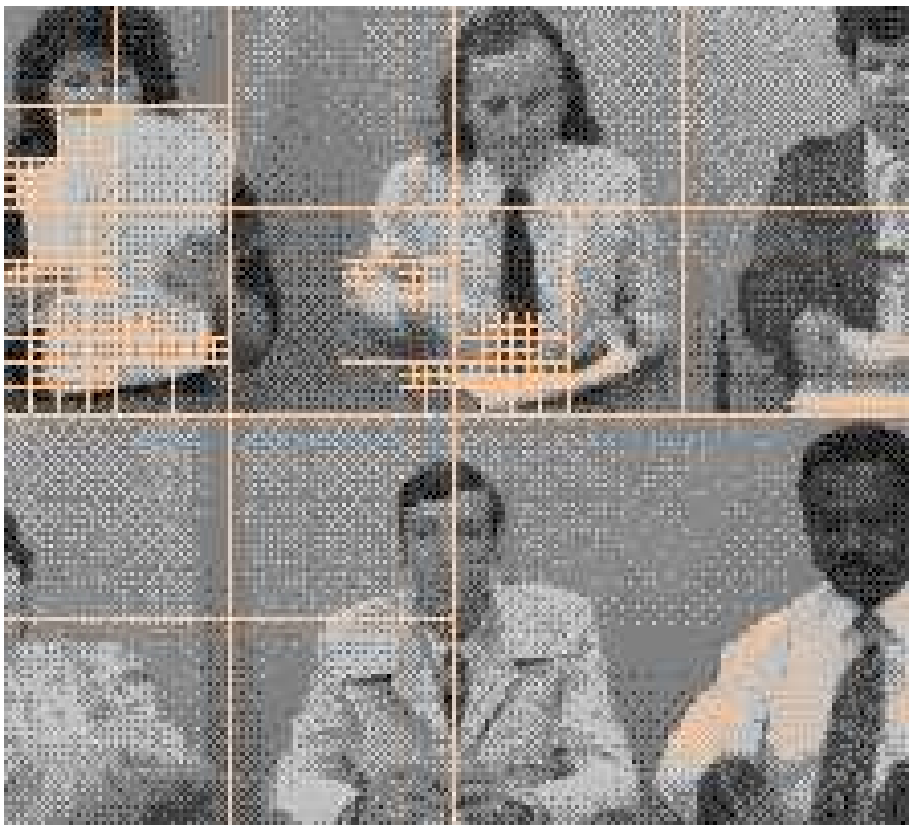


- Each object influences each other object
- Divide until each block contains exactly one object
- Accumulate force in each vertex
- Regard four vertices as one object

Video compression

Digital video

- Transmit the change between two consecutive frames.
- Divides a frame into blocks.
- Search in previous frame for similar block.
- Use small blocks for complex movements and large blocks for unchanged areas.

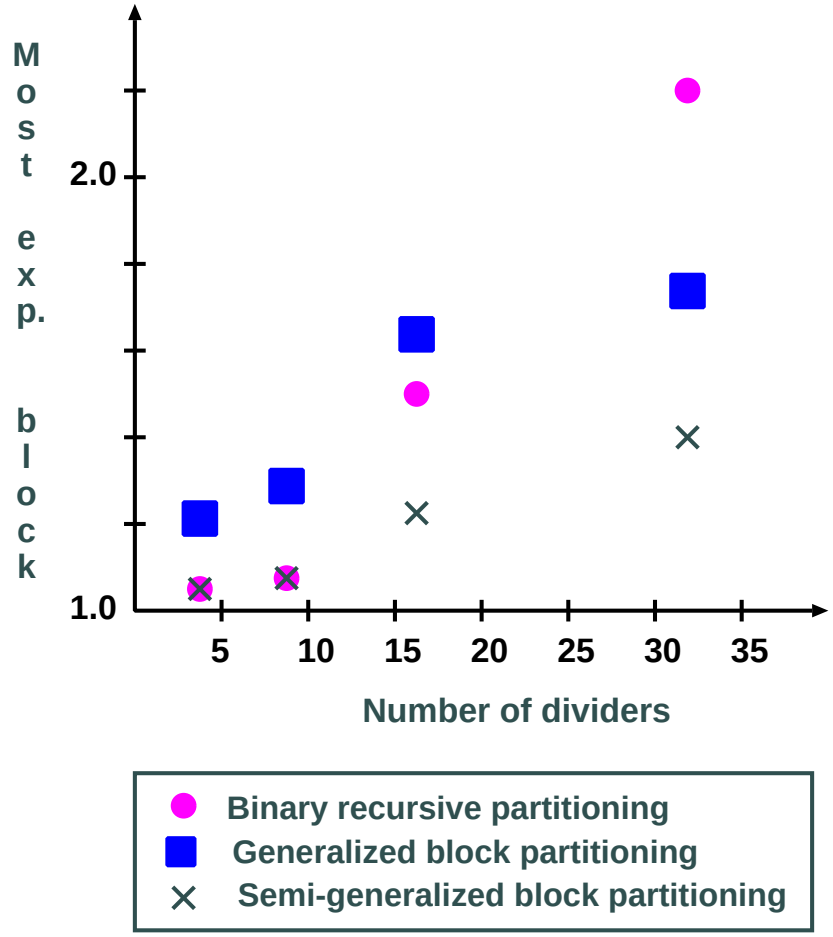


Transmit movement vector and change for each block.

2D Experiments

(Nicol 94) Graph partitioning using Generalized block partitioning: Better result for more structured graphs.

(Manne and Sørevik 96) Matrix with two distinct peaks.



Conclusion

Structured data partitioning

- Relatively simple to compute partitioning
- Can give worse load balance than graph partitioning
- Somewhat more limited applications than graph partitioning
- Easier to handle communication
- Can be handled by compiler
- Easier to develop code

For the future

- Parallel algorithms for load balancing
- Theoretical results Vs. practical experiences
- Other cost functions and applications