# Parallel sampling of Gaussian Markov random fields


Ingelin Steinsland

ingelins@math.ntnu.no

Norwegian University of Science and Technology

# Outline

- Parallelisation, why, how and how good.
- Parallel sampling of GMRFs
  - How
  - Performance
  - Analysis of Lancaster Campylobacter data.

# Why use parallel computers?

- Well known problems:
  - “Slow programs”
  - Not enough memory to solve a large problem.

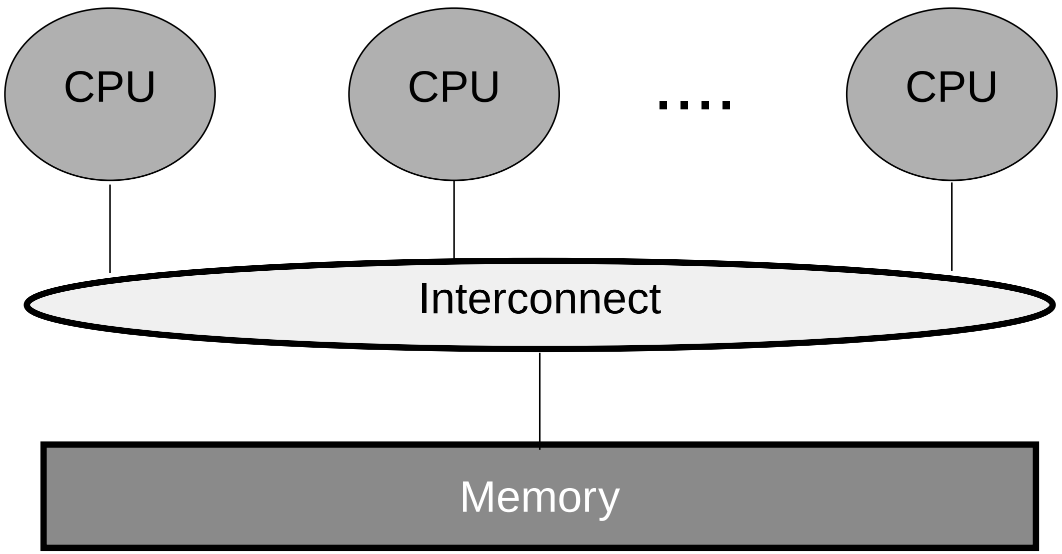
# Why use parallel computers?

- Well known problems:
  - “Slow programs”
  - Not enough memory to solve a large problem.
- Speed - faster programs
- Size - enabled to solve larger problems

# Parallel computer model

Multiple Instructions Multiple Data (MIMD)

- MIMD-SM (Shared Memory):

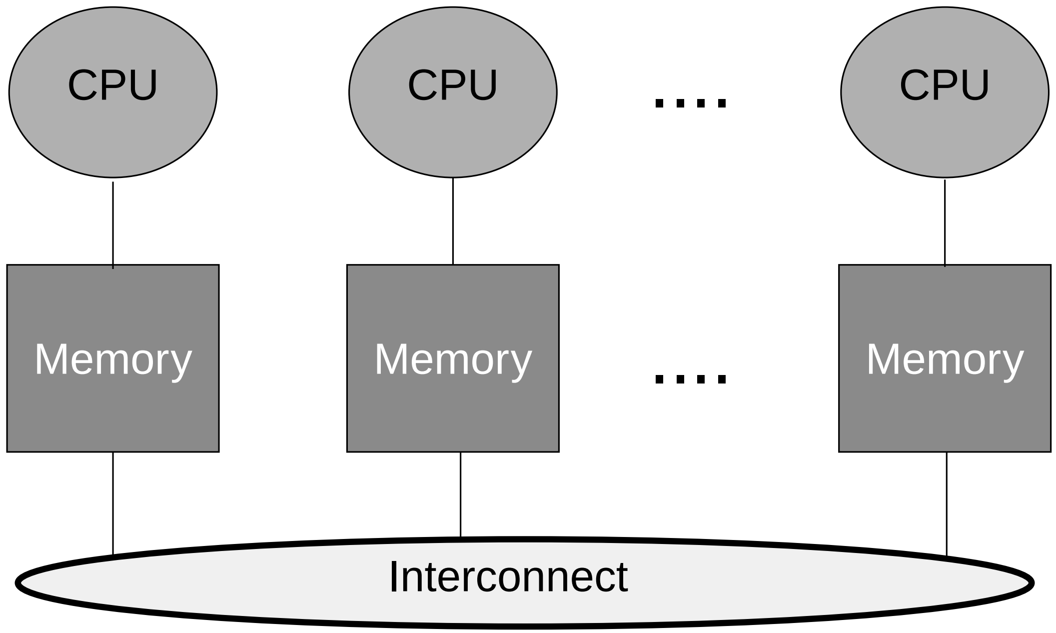



Communication: shared address space.

# Parallel computer model

Multiple Instructions Multiple Data (MIMD)

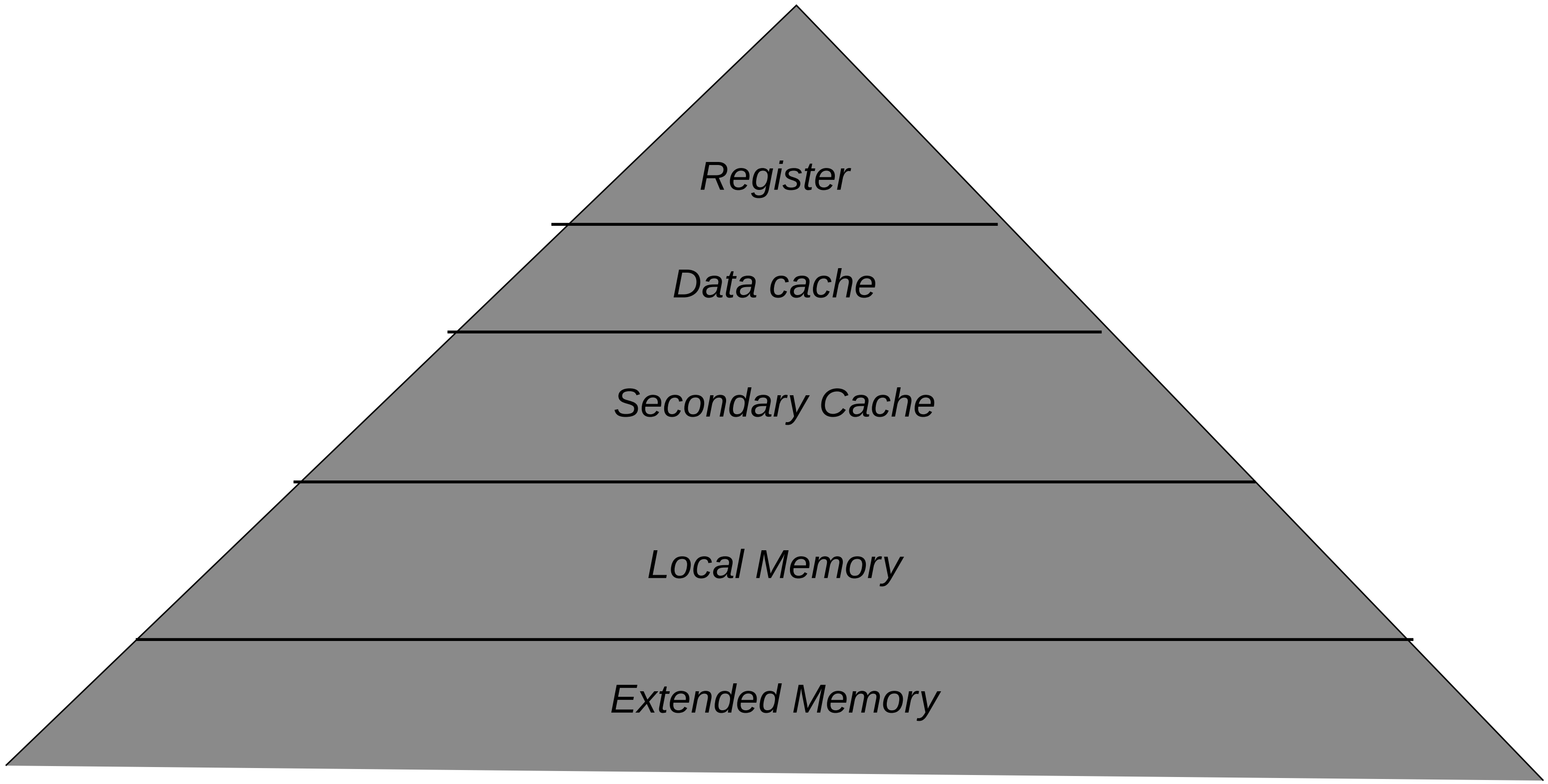
- MIMD-SM (Shared Memory):
  Communication: shared address space.
- MIMD-DM (Distributed Memory):




Communication: message passing

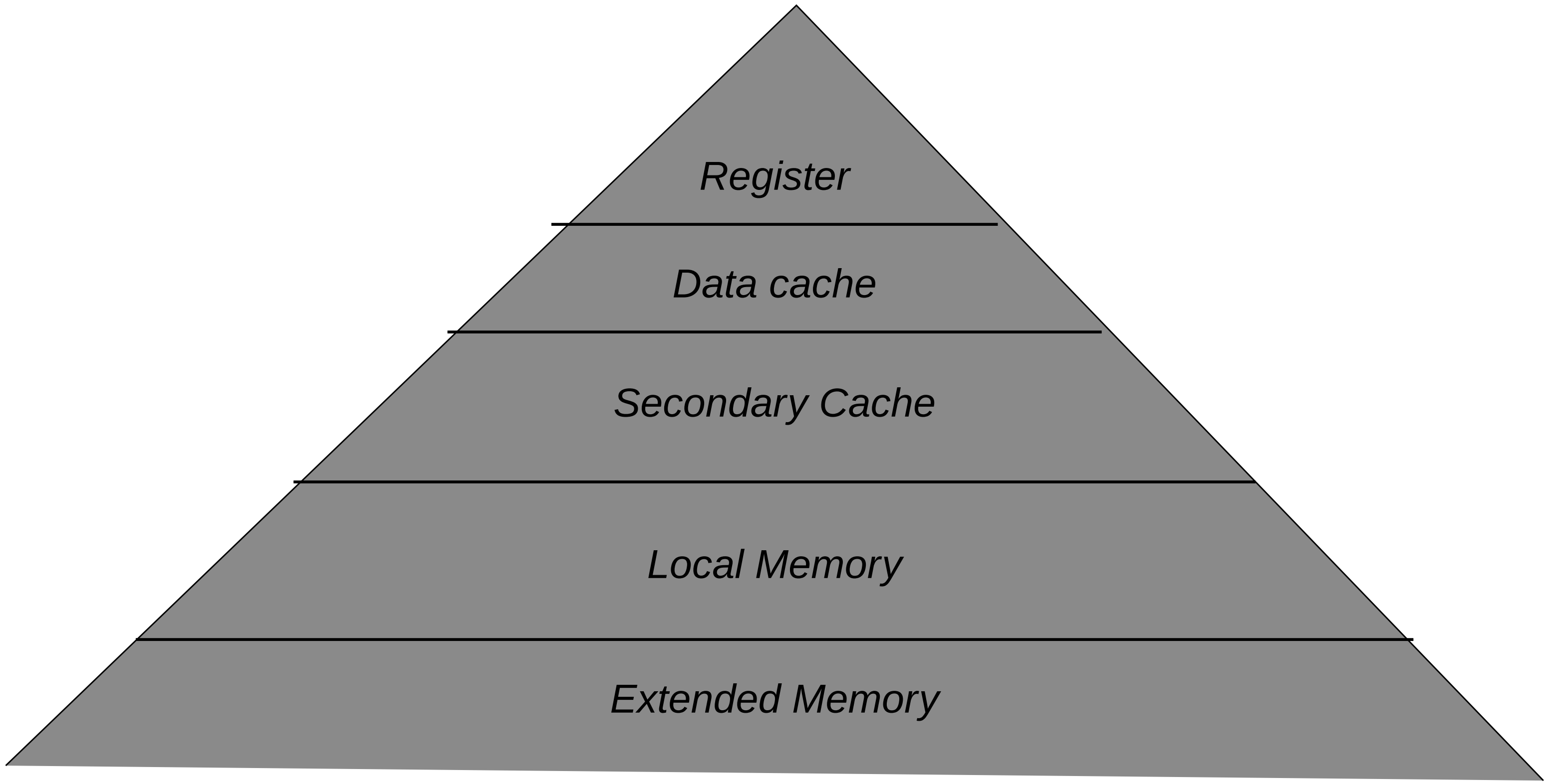
# Communication

- Can be thought of as extended memory;



# Communication

- Can be thought of as extended memory;




- Communication tine ($T_C$) model:
  $T_C = T_{IC} + \frac{n_{Data}}{B}$
  $T_{IC}$: initial cost, $n_{Data}$: amount of data and $B$: bandwidth

# Designing parallel algorithms

*"Parallel algorithm design (...) it requires the sort of integrative thought that is commonly referred as "creativity"."*

We often need to change our approach to the problem.
Main gold: Speed-up and/or handle larger problems.

# Designing parallel algorithms

*"Parallel algorithm design (...) it requires the sort of integrative thought that is commonly referred as "creativity"."*

We often need to change our approach to the problem.
Main gold: Speed-up and/or handle larger problems.

- Scalability: Should be able to benefi t from more computers.
- Unique solution: Should exist a sequential program that always gives the same result.

# Designing strategies

- Functional decomposition,
  - same data, different functions.
- Domain decomposition,
  - different data, same functions.

# Performance measures

$p$: number of processors, $A$ problem size.

- Speed-up$(p) = \dfrac{\text{time}_1(A)}{\text{time}_p(A)}$

# Performance measures

$p$: number of processors, $A$ problem size.

- Speed-up$(p) = \frac{\text{time}_1(A)}{\text{time}_p(A)}$
- Scaled speed-up$(p) = \frac{p \cdot \text{time}_1(A)}{\text{time}_p(p \cdot A)}$

# Performance measures

$p$: number of processors, $A$ problem size.

- Speed-up$(p) = \frac{\text{time}_1(A)}{\text{time}_p(A)}$
- Scaled speed-up$(p) = \frac{p \cdot \text{time}_1(A)}{\text{time}_p(p \cdot A)}$
- Amdahl's law (f; parallisable fraction):

$$\text{Speed-up} = \frac{1}{(1-f) + \frac{f}{p}}$$

20% sequential $\Rightarrow$ max speed-up $5$.

# Performance measures

$p$: number of processors, $A$ problem size.

- Speed-up$(p) = \frac{\text{time}_1(A)}{\text{time}_p(A)}$
- Scaled speed-up$(p) = \frac{p \cdot \text{time}_1(A)}{\text{time}_p(p \cdot A)}$
- Amdahl's law (f; parallisable fraction):

$$\text{Speed-up} = \frac{1}{(1-f) + \frac{f}{p}}$$

  20% sequential $\Rightarrow$ max speed-up $5$.
- Super linear speed-up.

# Communication overhead and load balance

- Load balance: Computers should not be idle.
- Communication overhead: Communication is expensive, often the major part of the extra cost.

# Parallel exact sampling of GMRFs

- Computational benefits of GMRFs (sequential).
- Parallelisation of GMRFs
- Use methods from numerical linear algebra.
- Use GMRFs in Markov chain Monte Carlo simulation.

# Exact sampling from multivariate Gaussian distribution

- $x \sim N(0, Q^{-1}) \Rightarrow \pi(x) \propto \exp(-\frac{1}{2}x^T Q x)$
- $Q = LL^T$, $L$ is the Choleskey factor, lower triangular
- $\pi(x) \propto \exp(-\frac{1}{2}x^T LL^T x)$
- $z \sim N(0, I)$ i.i.d. standard Gaussian.
- $\pi(z) \propto \exp(-\frac{1}{2}z^T z)$
- The solution of $L^T x = z$ is a sample from our Gaussian distribution.

# Exact sampling from multivariate Gaussian distribution

- $x \sim N(0, Q^{-1}) \Rightarrow \pi(x) \propto \exp(-\frac{1}{2}x^T Q x)$
- $Q = LL^T$, $L$ is the Choleskey factor, lower triangular
- $\pi(x) \propto \exp(-\frac{1}{2}x^T LL^T x)$
- $z \sim N(0, I)$ i.i.d. standard Gaussian.
- $\pi(z) \propto \exp(-\frac{1}{2}z^T z)$
- The solution of $L^T x = z$ is a sample from our Gaussian distribution.

Computational complexity: $\mathcal{O}(n^3)$.

# Why GMRF

- The Markov property makes $Q$ sparse.
- **Precision matrix:** $Q_{ij} = 0 \Rightarrow x_i$ and $x_j$ are conditional independent given $\forall x_k, k \neq i, j$
- **Choleskey factor:** $L^T_{ij} = 0 \; (i < j) \Rightarrow x_i$ and $x_j$ are conditional independent given $\forall x_k \mid k \neq j \wedge k > i$.

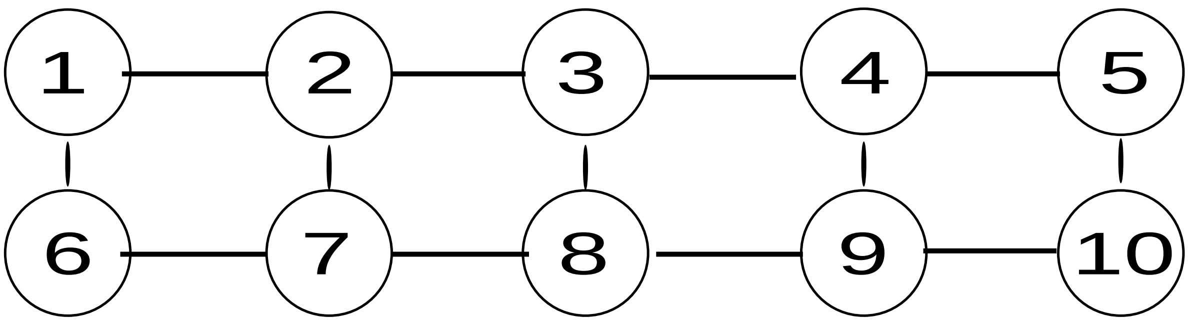
# Why GMRF

- The Markov property makes $Q$ sparse.
- **Precision matrix:** $Q_{ij} = 0 \Rightarrow x_i$ and $x_j$ are conditional independent given $\forall x_k, k \neq i, j$
- **Choleskey factor:** $L_{ij}^T = 0 \; (i < j) \Rightarrow x_i$ and $x_j$ are conditional independent given $\forall x_k \mid k \neq j \wedge k > i$.

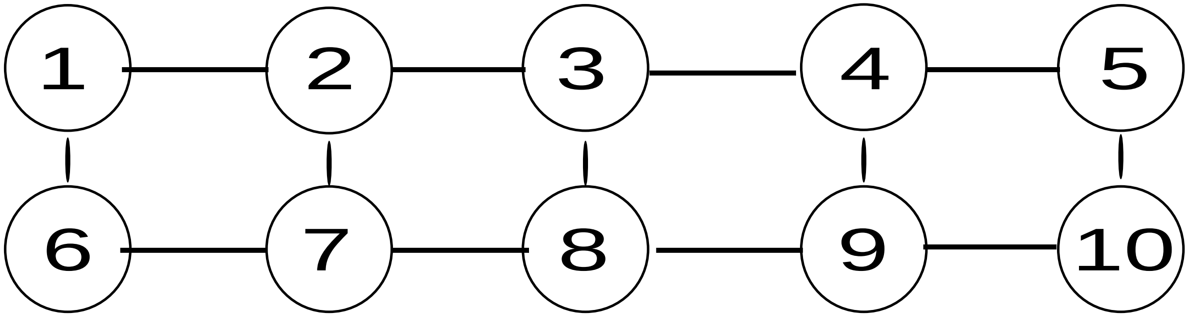
Sparse $Q \Rightarrow$ sparse $L$?

# Graph

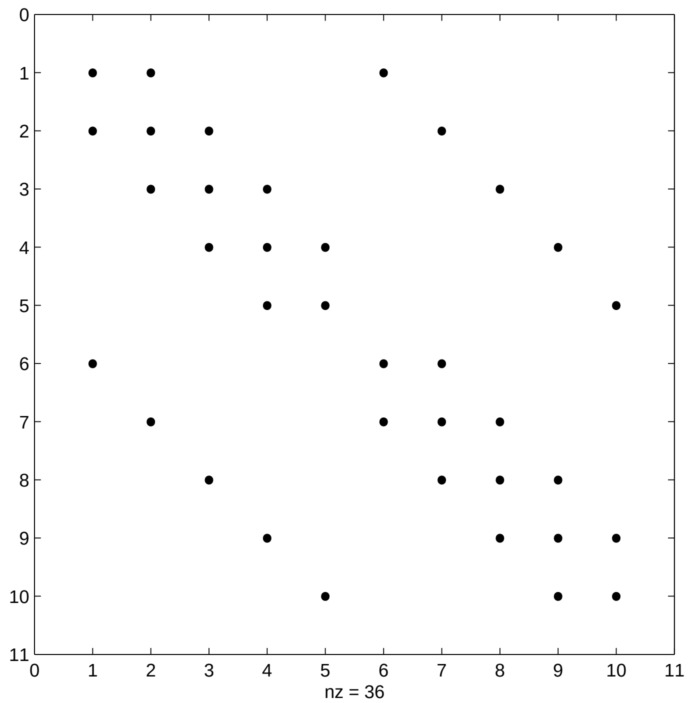
- Each variable is a node.
- Edges between neighbours



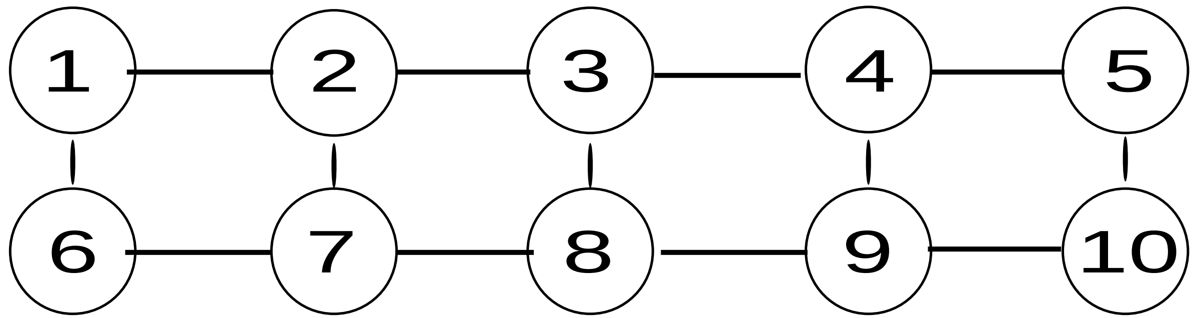
# Graph



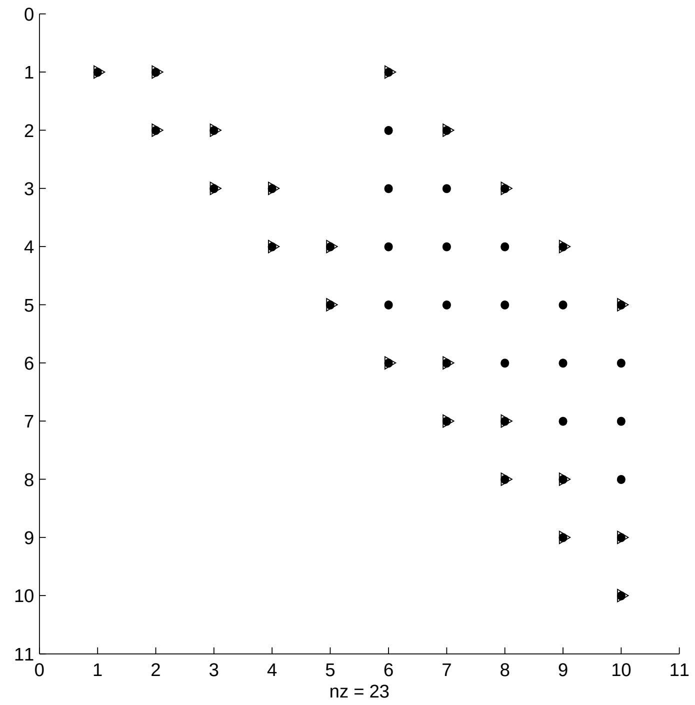

Precision matrix:



# Graph

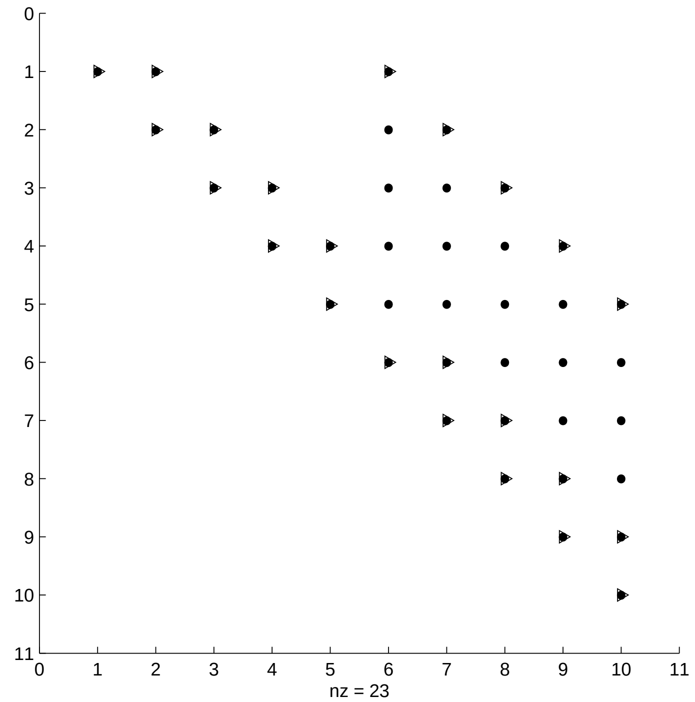



Choleskey factor $L^T$, $\triangle$ non-zeros in $Q$.



# Graph

Choleskey factor $L^T$, $\triangle$ non-zeros in $Q$.




**Fill-in:** The elements that are zero in $Q$, but non-zero in $L$.

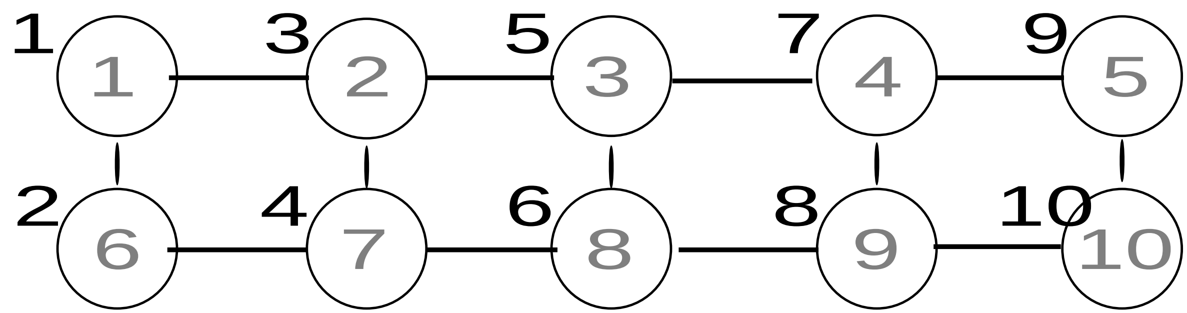
Reduce fill-in $\Rightarrow$ cheaper calculations.

# Reordering

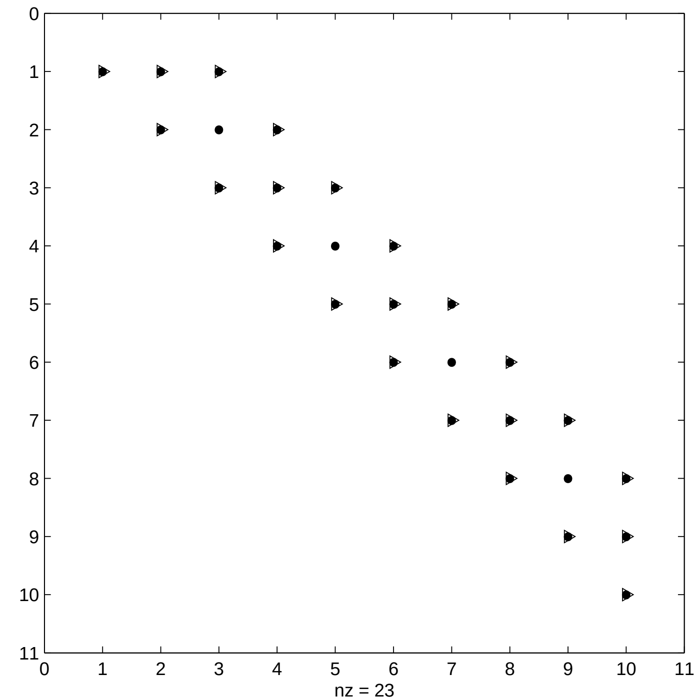
- The ordering of the variables are dummy.
- A reordering can reduce the fi ll-in.

# Reordering

Reordered graph:

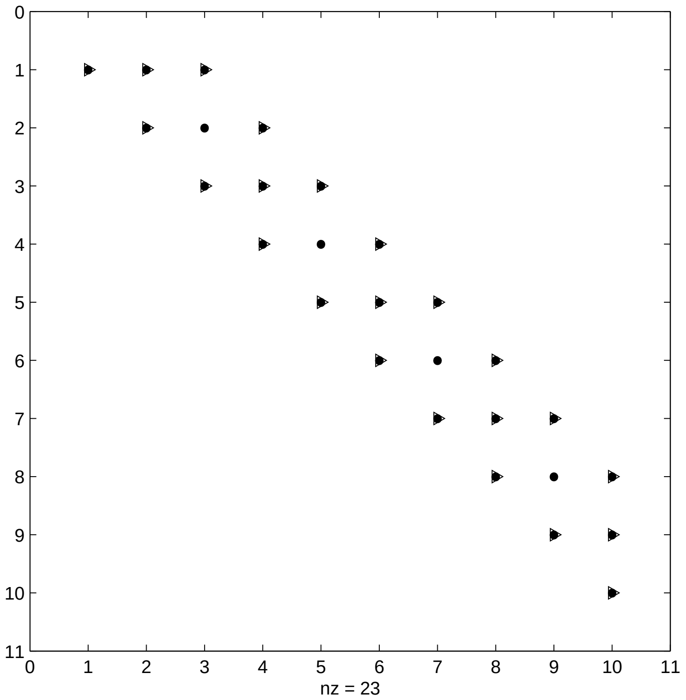



New Choleskey factor $L$, $\triangle$ non-zeros in $Q$.



# Reordering

New Choleskey factor $L$, $\triangle$ non-zeros in $Q$.

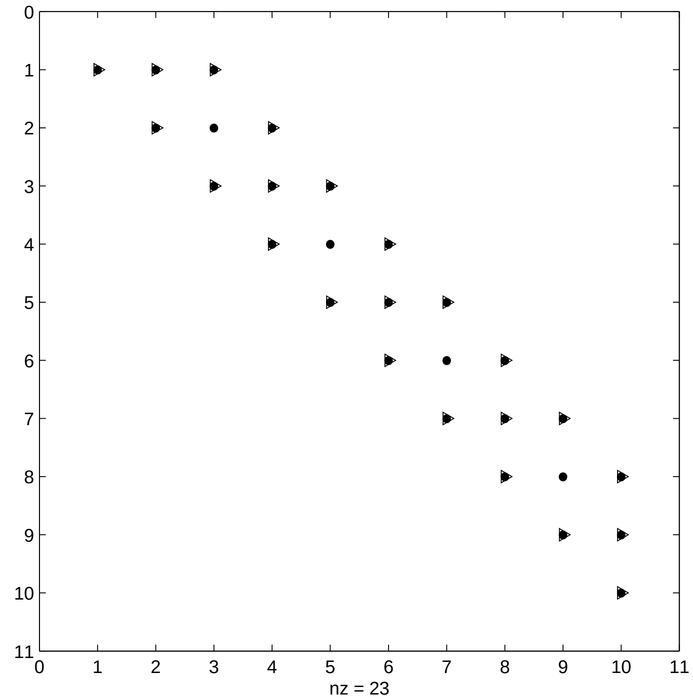



Original ordering: fi ll-in = 16

Reordered graph: fi ll-in = 4

# Reordering

New Choleskey factor $L$, $\triangle$ non-zeros in $Q$.




Original ordering: fill-in = 16
Reorder graph: fill-in = 4

- Computational complexity spatial GMRF: $\mathcal{O}(n^{1.5})$

# Fast sampling GMRF

Algorithm:

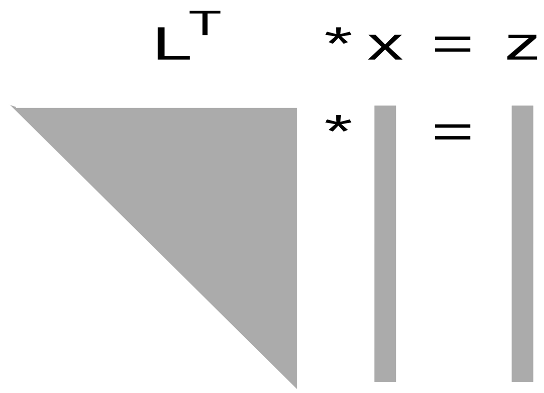
- Reorder (reduce fill-in)
- Calculate $L$
- Solve $L^T x = z$

# Fast sampling GMRF

Algorithm:

- Reorder (reduce fill-in)
- Calculate $L$
- Solve $L^T x = z$

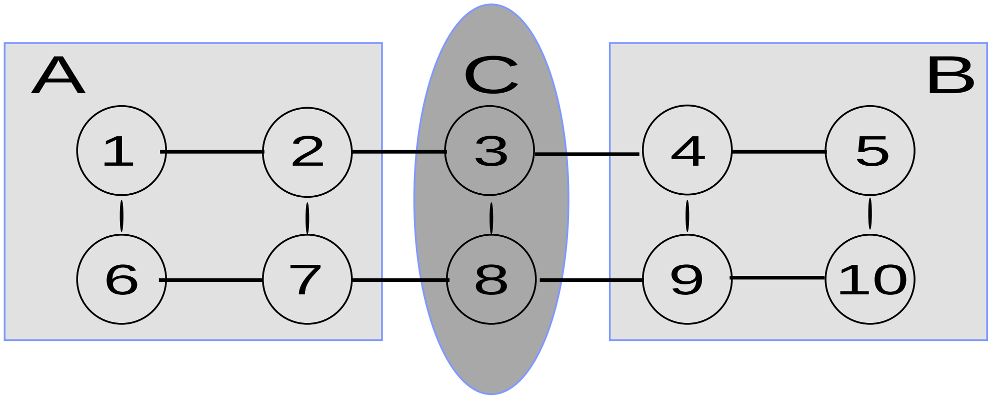
Triangular system:




$$\pi(x) = \pi(x_n)\pi(x_{n-1}|x_n)\dots\pi(x_1|x_2, x_3, \dots, x_n)$$

# Parallel sampling of GMRF

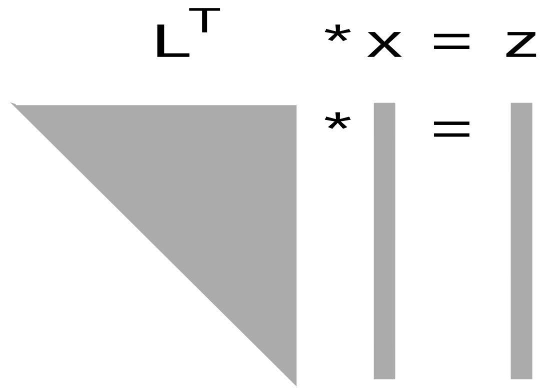
Intuitive idea:



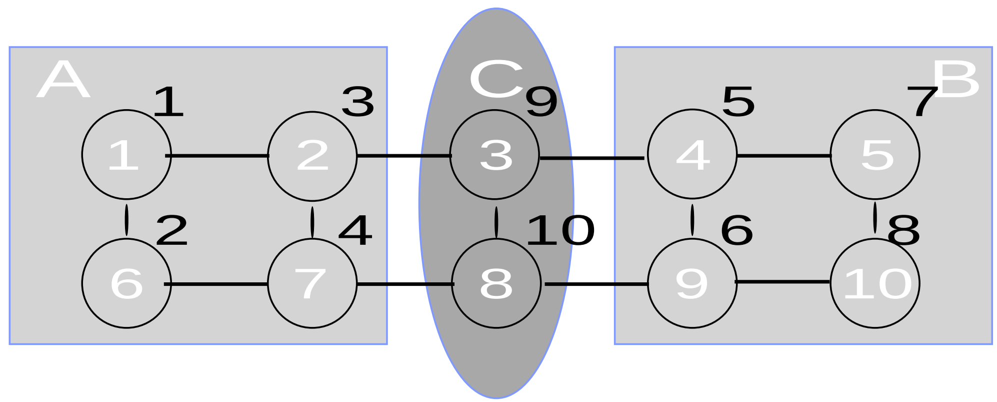

- If we have a sample $x_C \sim \pi_C(x)$.
- $x_A \sim \pi_{A|C}(x)$ and $x_B \sim \pi_{B|C}(x)$ are independent
- $x_A|x_C$ and $x_B|x_C$ can be sampled in parallel.
- $x^* = (x_A, x_C, x_B)$ a sample from our GMRF.

Markov property used to get conditional independent sets

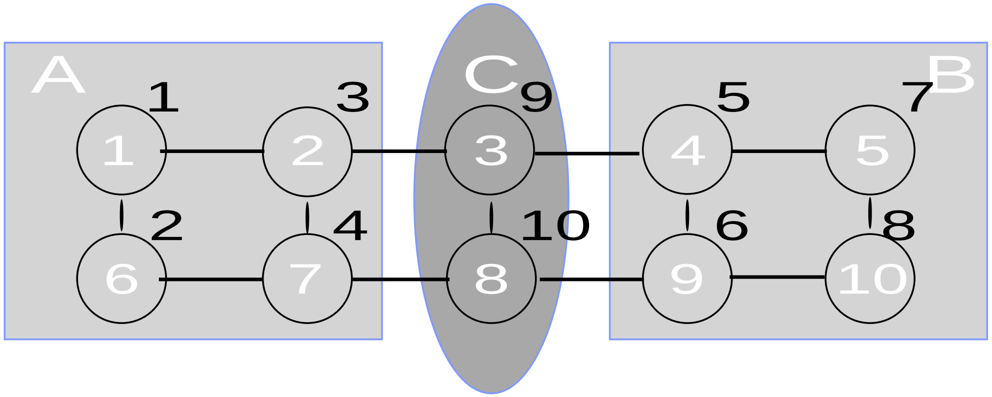
# Parallel solving of triangular system



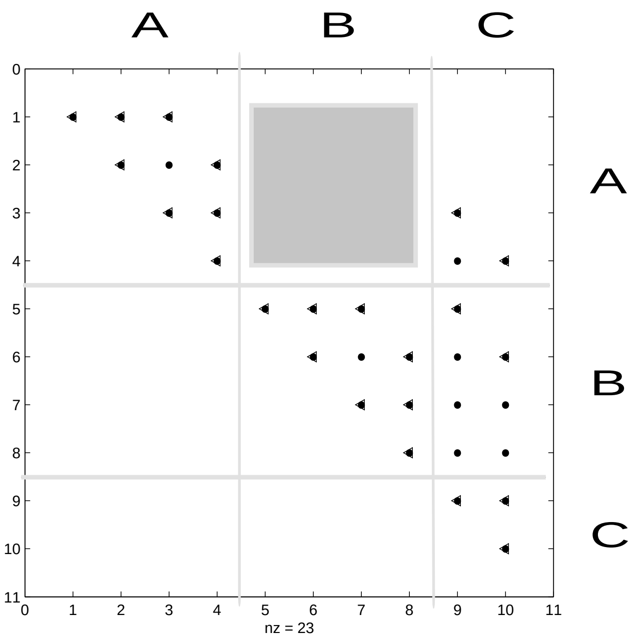

Reordering $x_r = (x_A, x_B, x_C)$:



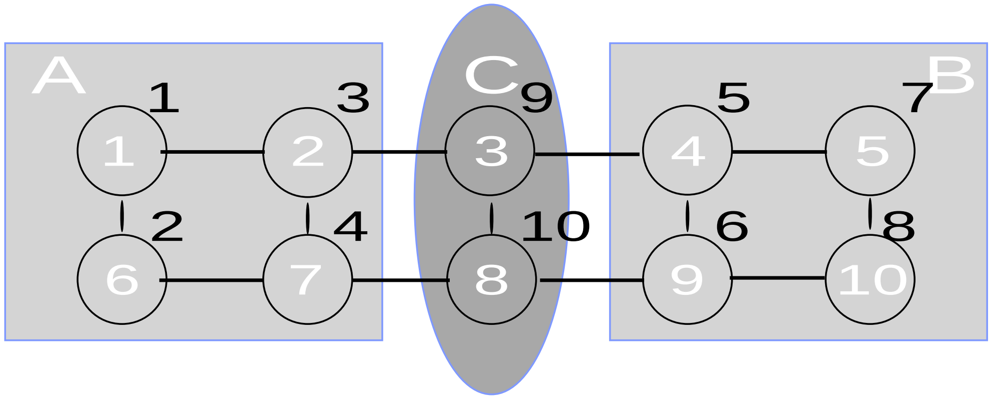
# Parallel solving of triangular system

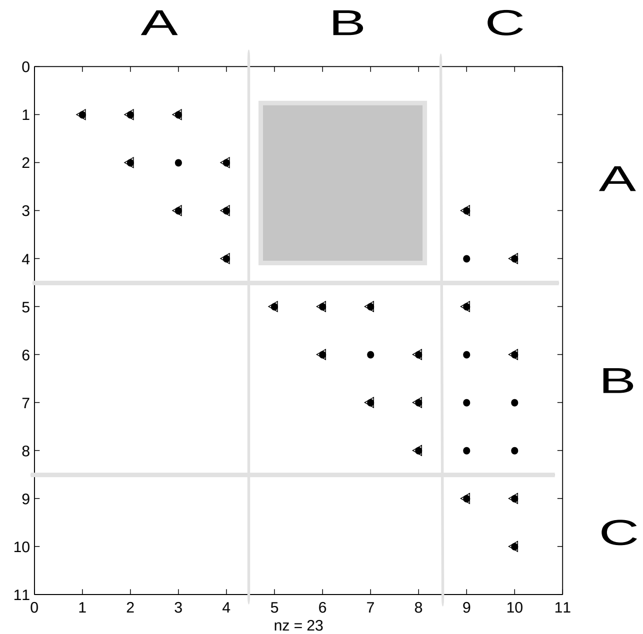



Choleskey factor $L^T$



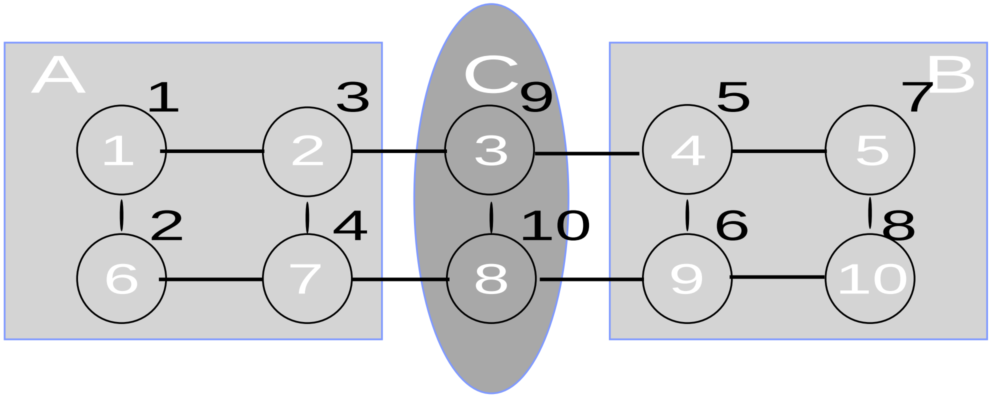
# Parallel solving of triangular system

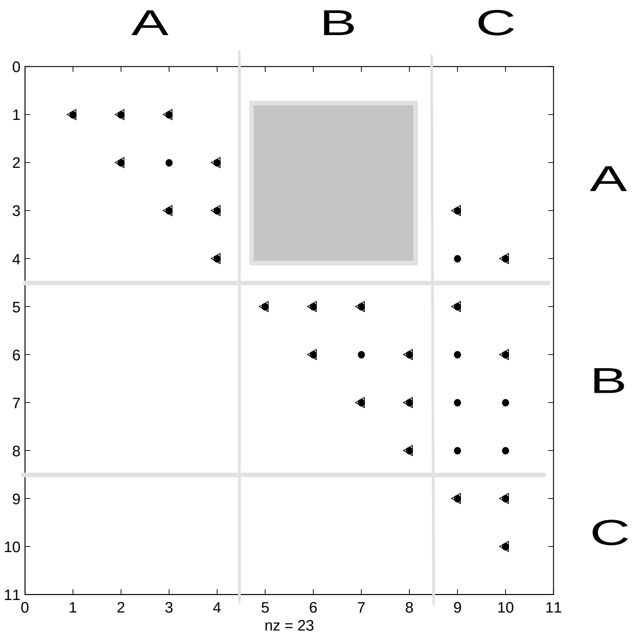



$AB$ zero $\Rightarrow$ $A$ and $B$ can be calculated in parallel.

# Parallel solving of triangular system

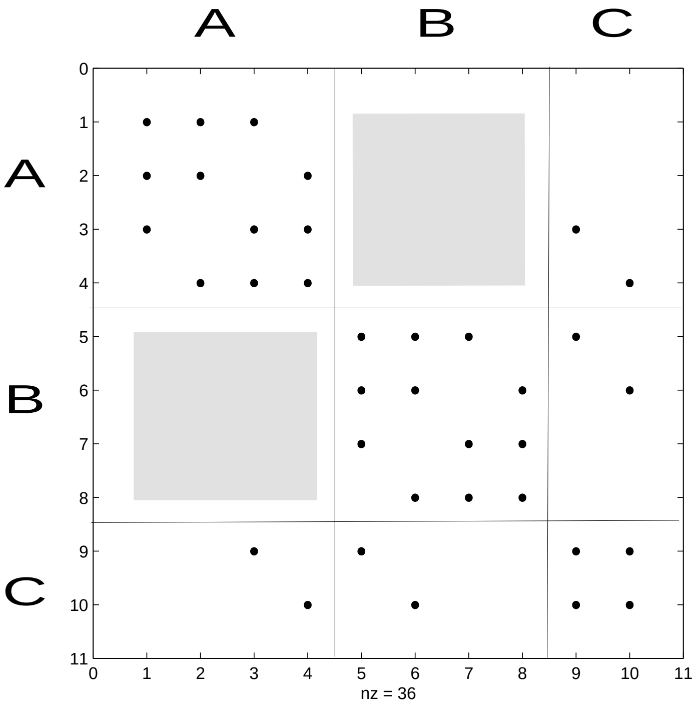



Parallel ordering; fill-in = 8 (best sequential; fill-in = 4)

# Parallel Choleskey factorisation

Precision matrix for $x_r$:




- Choleskey decomposition is done by "column wise right looking elimination" $\Rightarrow$ Choleskey part OK.

# Interpretation and computers

Library

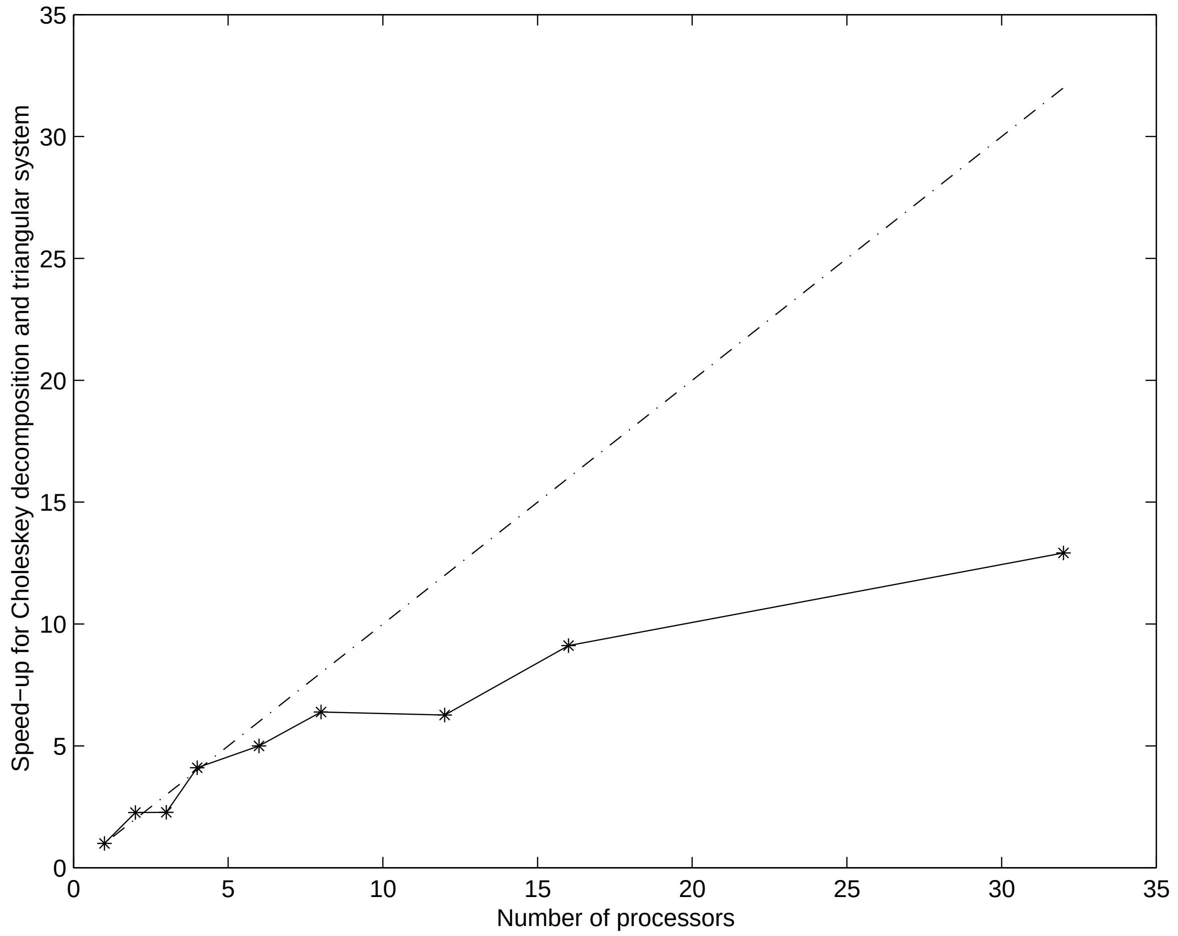
- Use WSMP ((the Watson Sparse Matrix Package, only for IBM RS6000 workstation and IBM SP systems)
- Dimension should be $\geq 5000$.

Computers

- 48 node SP/2 system (Queens University Belfast)
- 160MHz Power2CS processors.
- Nodes have between 256Mb and 1Gb of main memory.

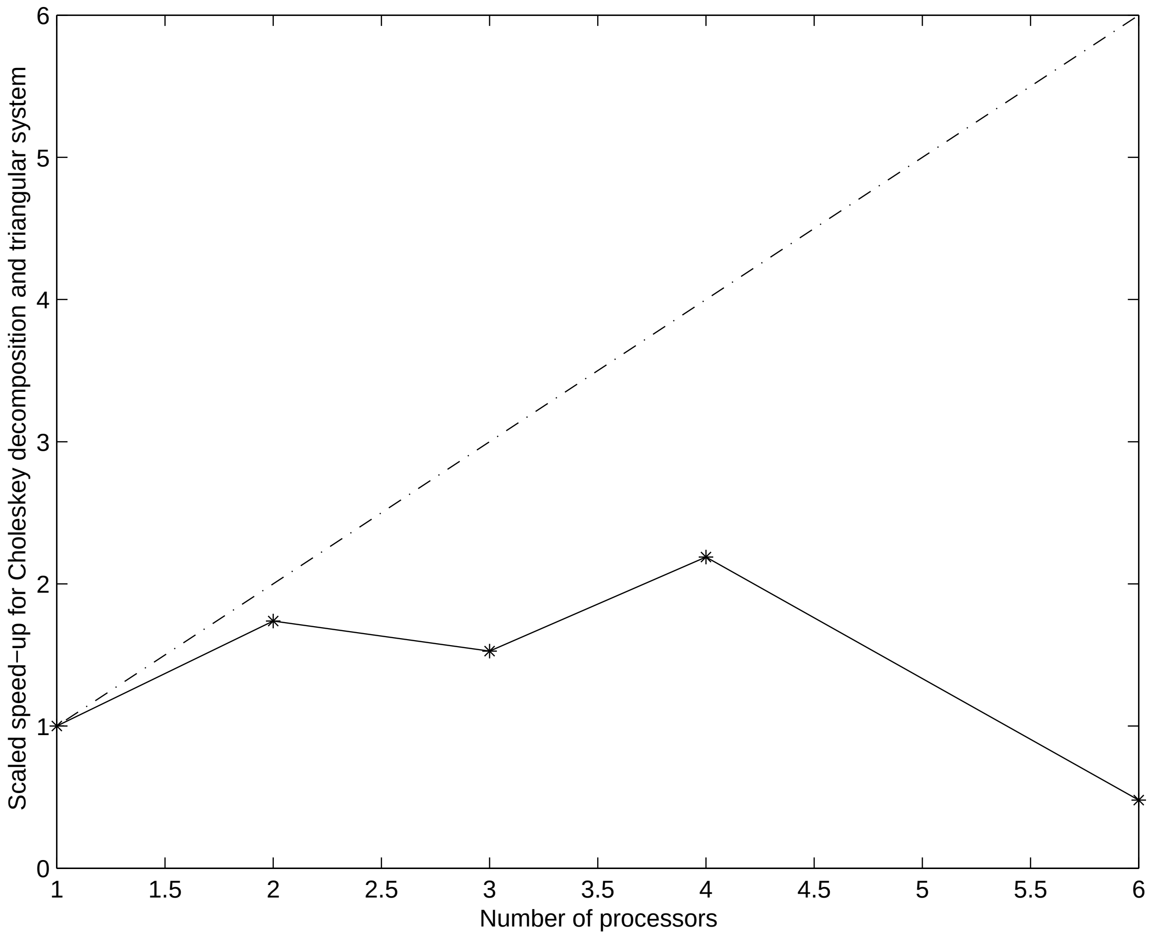
# Performance example 1

- Sampling GMRFs on a $400 \times 400$ lattice ($160000$ variables).



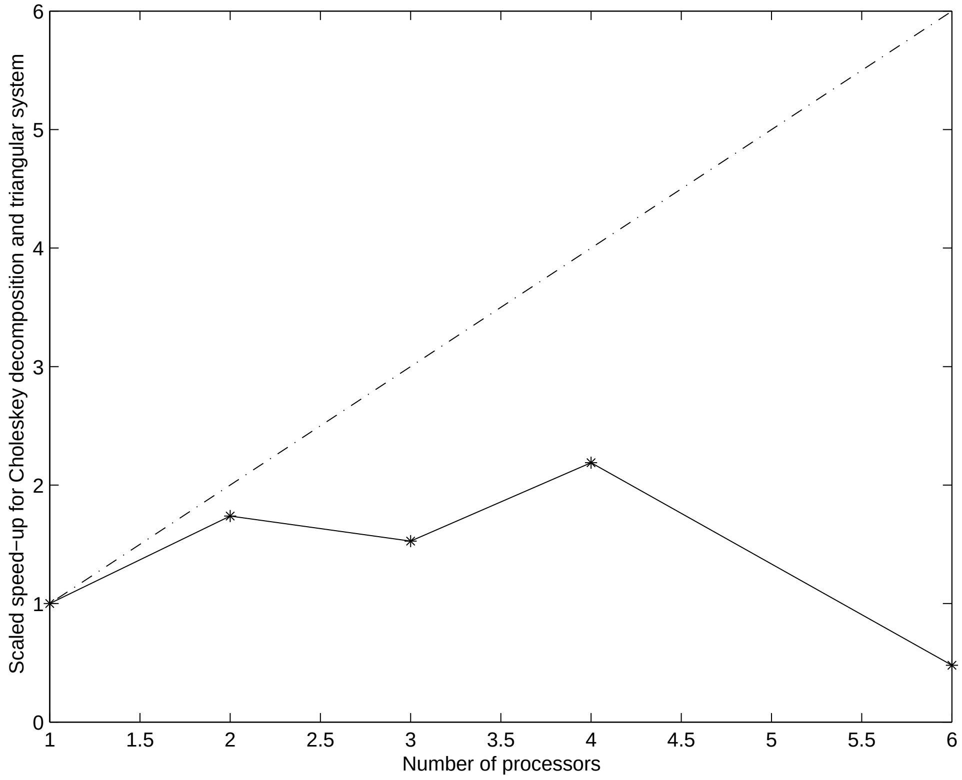
# Performance example 2

- Sampling GMRFs on a $400\sqrt{p} \times 400\sqrt{p}$ lattice ($160000 \cdot p$ variables).



# Performance example 2

- Sampling GMRFs on a $400\sqrt{p} \times 400\sqrt{p}$ lattice ($160000 \cdot p$ variables).

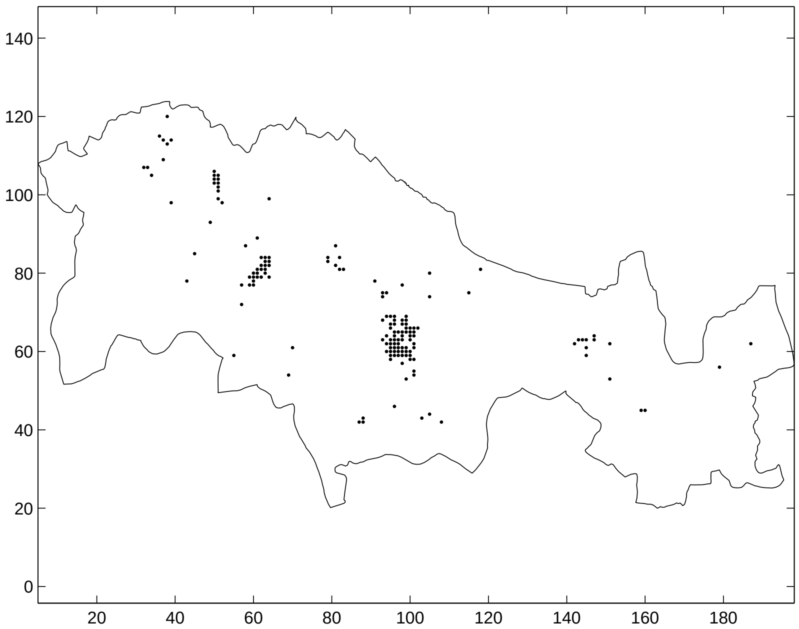



- $565 \times 565$ ($p = 2$) problem too large for one processor.

# Example 3

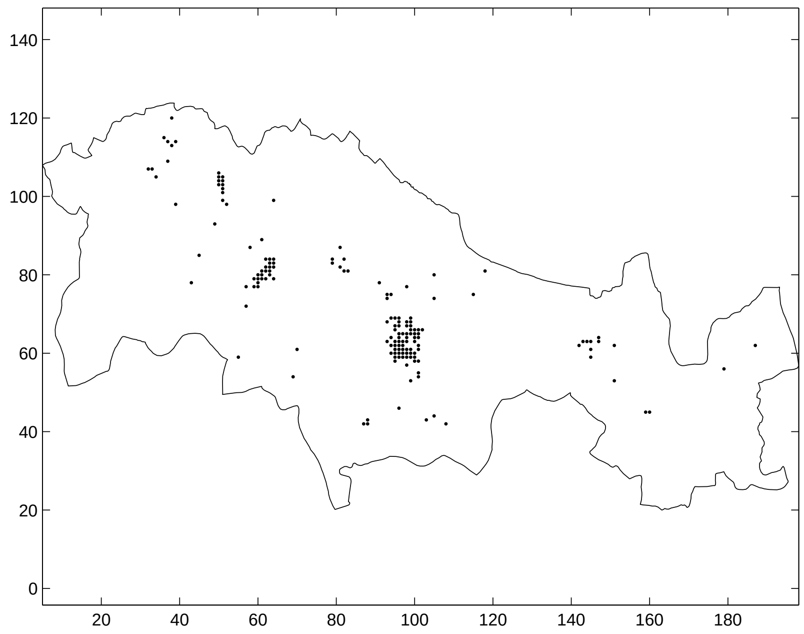
Campylobacter infections in north Lancaster.

- The data
  - 399 outbreak of enteric infections, $m_{ij}$
  - 234 of these are campylobacter, $y_{ij}$
  - their location $(i, j)$ (248 different location).



# Example 3

Campylobacter infections in north Lancaster.




- Of interest: The proportion of enteric outbreaks that are Campylobacter and its spatial variation.

# GMRF model

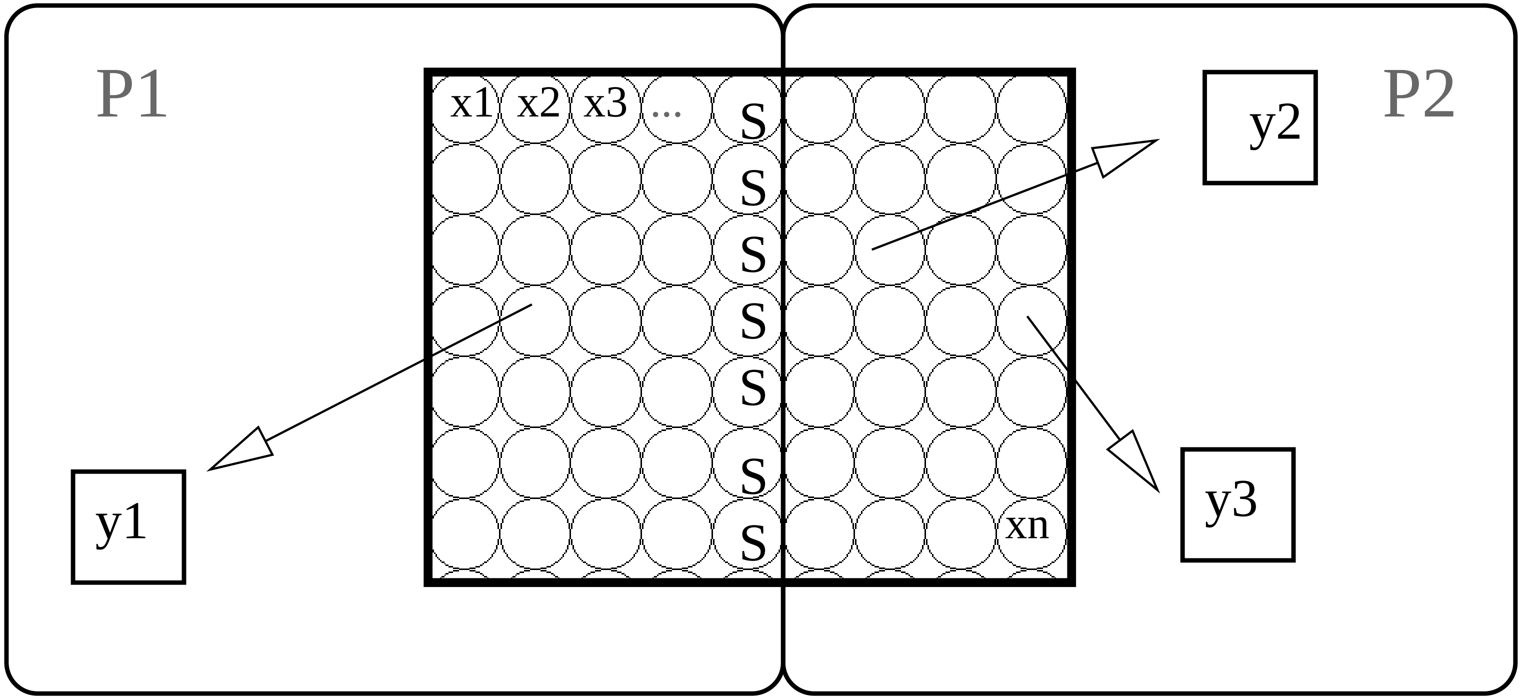
- GMRF model:
  - Probability of 'success' $p_i$ given by

$$\log(\frac{p_i}{1-p_i}) = \beta + x_i$$

  - $\beta$ location independent constant
  - $x = (x_1, x_2, \ldots, x_n)$ a GMRF.
- Traditionally has GRF models been used, GMRF models almost equal and gives large computational benefits.

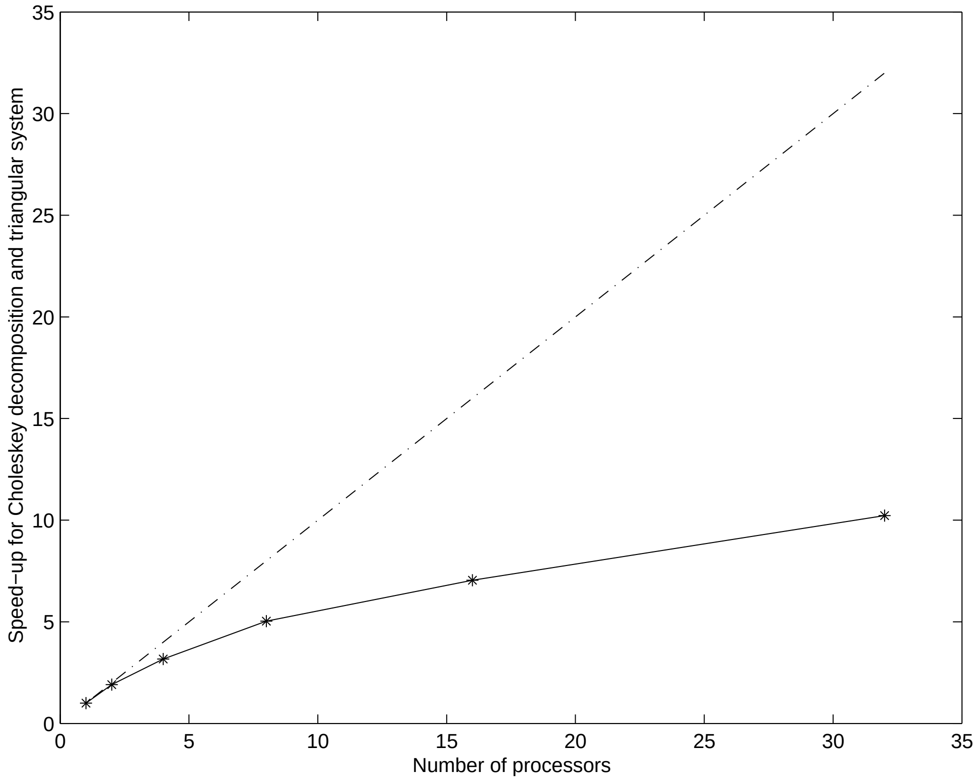
# Parallelisation

Domain decomposition:

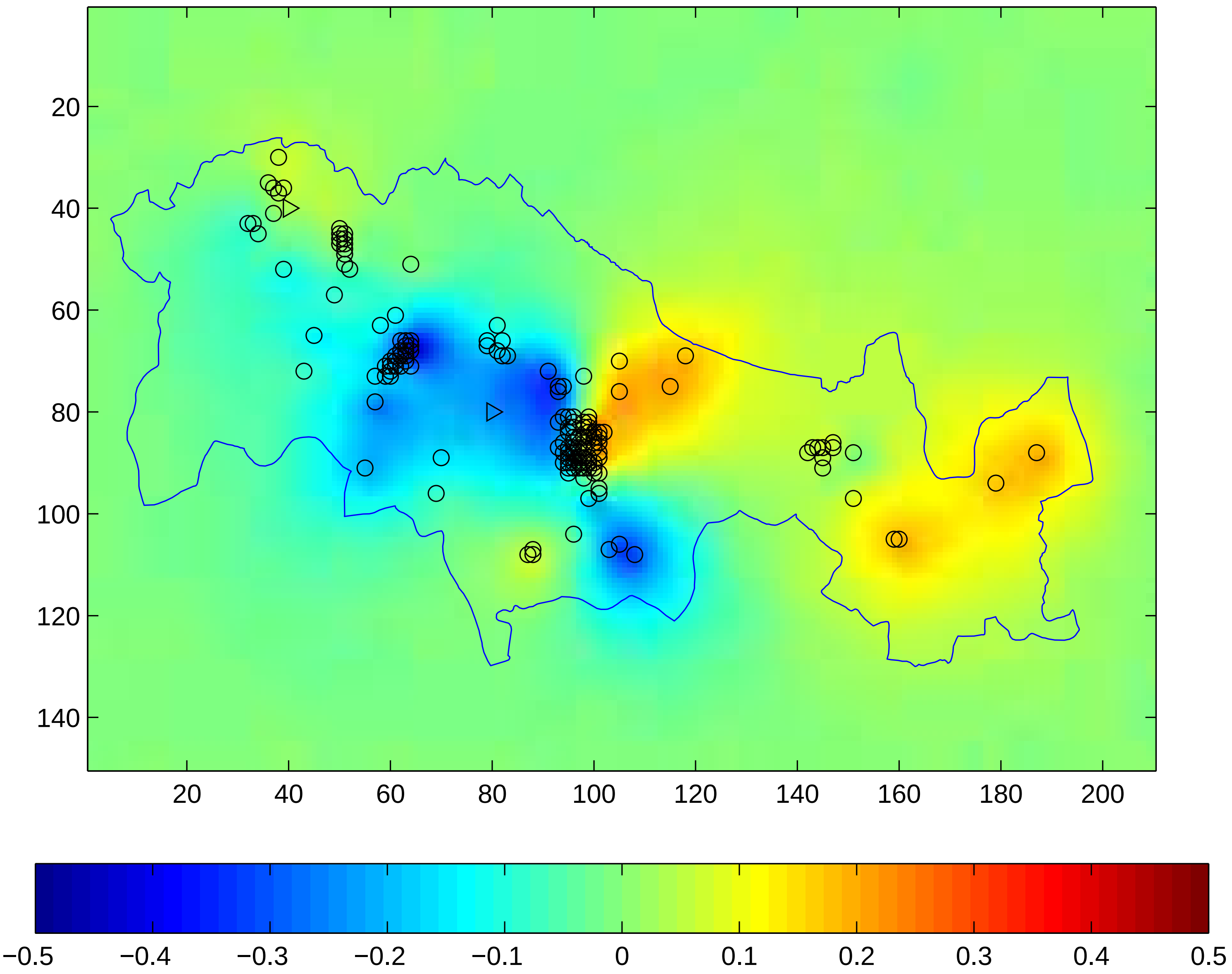


# Performance

Speed-up:



# Estimated mean



# Summary

- Parallelise to
  - solve problems faster and/or
  - be able to solve larger problems.
- Often have to take a new approach to the problem.
- Two main strategies;
  - functional decomposition and
  - domain decomposition
- Communication between processors is often the major extra cost of parallelisation;
  - load balance and
  - communication overhead.