Senter for forskningsdrevet innovasjon

(sfi)^2

STATISTICS FOR INNOVATION

Monte Carlo simulation in statistics, with a view to recent ideas

ARNOLDO FRIGESSI
FRIGESSI@MEDISIN.UIO.NO

UNIVERSITAS OSLOENSIS MDCCCXI

FUGE Funksjonell genomforskning

Norges forskningsråd

NR

THE JOURNAL OF CHEMICAL PHYSICS VOLUME 21, NUMBER 6 JUNE, 1953

Equation of State Calculations by Fast Computing Machines

NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,

Los Alamos Scientific Laboratory, Los Alamos, New Mexico

AND

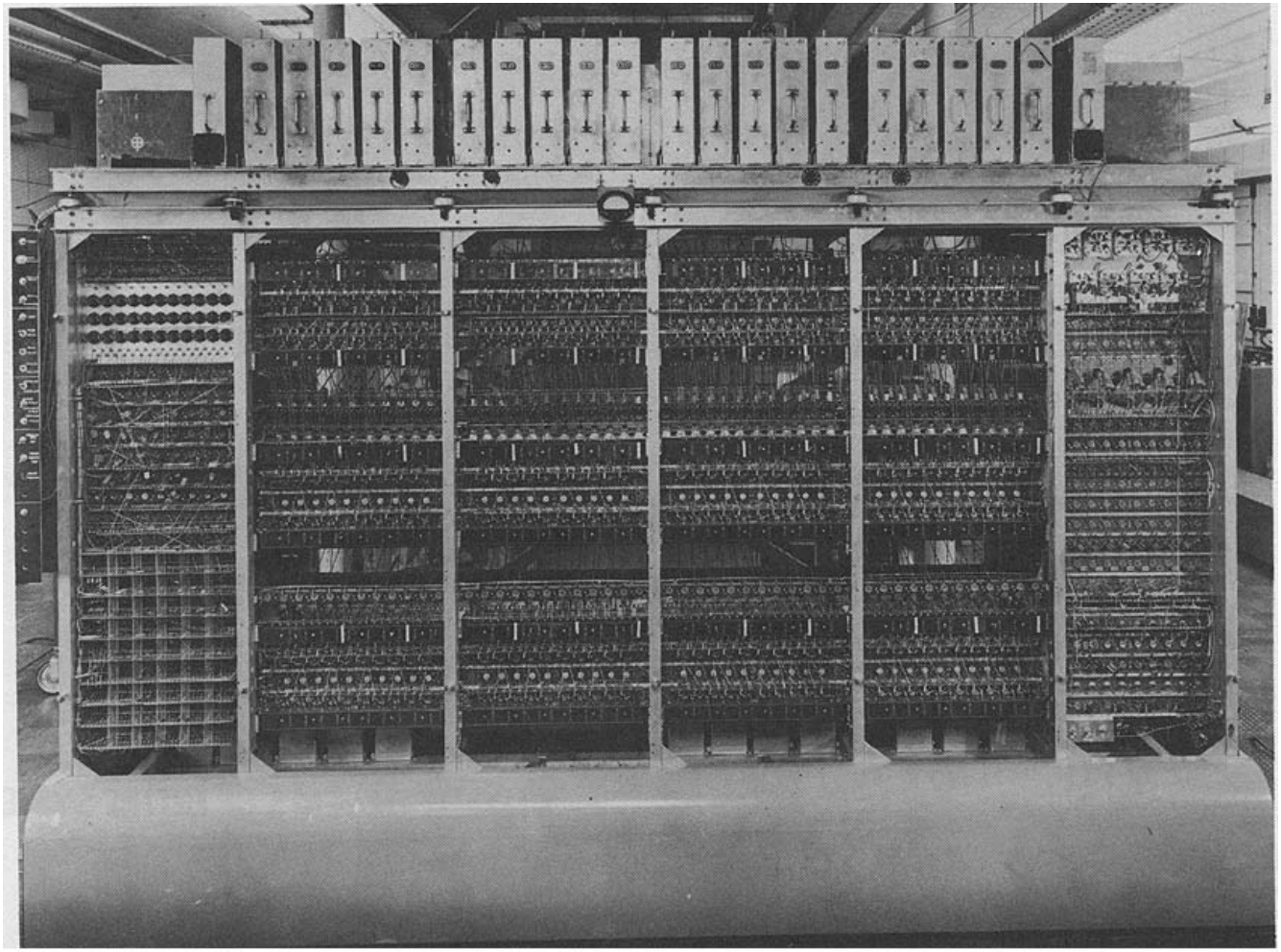
EDWARD TELLER, * *Department of Physics, University of Chicago, Chicago, Illinois*

(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.

The Metropolis algorithm from1953 has been cited as among the top 10 algorithms having the "greatest influence on the development and practice of science and engineering in the 20th century."

MANIAC



The MANIAC II

1952

Multiplication took a milli-second.

(Metropolis choose the name MANIAC in the hope of stopping the rash of ”silly” acronyms for machine names.)

Some MCMC milestones

1953 Metropolis

Heat bath, dynamic Monte Carlo

1970 Hastings

Political Analysis, 10:3 (2002) Society for Political Methodology

Location, Location, Location:
An MCMC Approach to Modeling the Spatial Context of War and Peace

Michael D. Ward and Kristian Skrede Gleditsch

1993 Besag & Green, MCMC

1994 Grenander & Miller

1995 Green Transdimensional MCMC

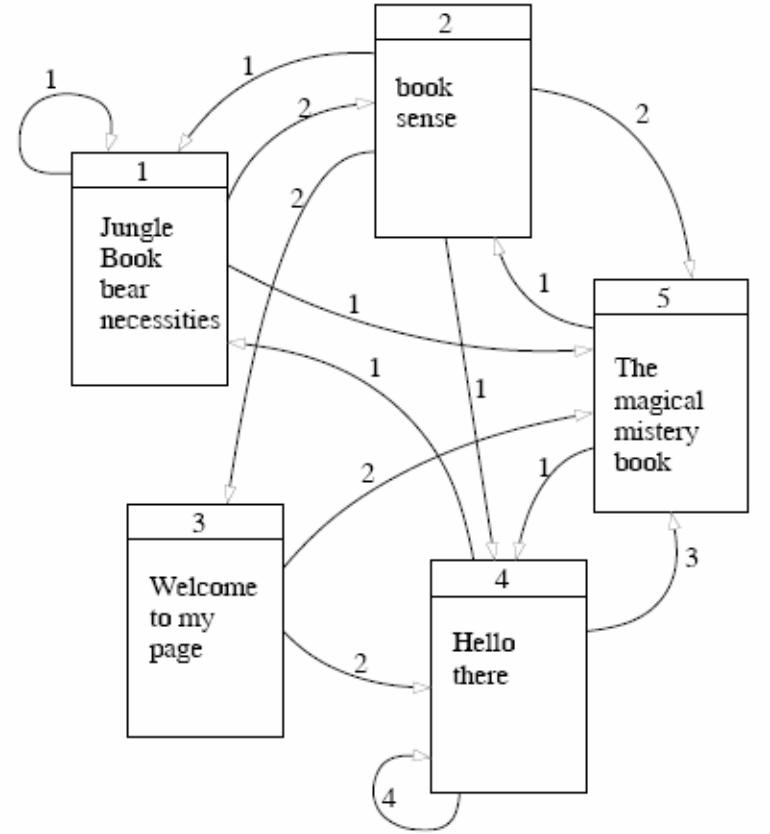
1996 Propp & Wilson Exact Sampling

NOW from Riemann to Lebesgue MCMC

The MCMC idea

- Design a Markov Chain on a state space S, such that when simulating a trajectory of states $x_1, x_2, \ldots x_t$, from it, you will explore the state space S spending more time in the most important regions, where importance is given by a probability *p(x)* on S, which is the ergodic limit of the chain. (i.e. you will stay most where *p(x)* is large)

Google vs. MCMC



PageRank

- Supposing you browse this for infinitely long time, what is the probability to be at page x_i.
- No matter where you started.

$p\,T = p$

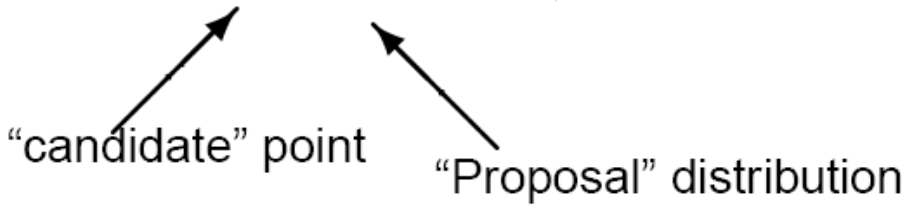
- Google: given **T** Google finds $p(x)$
- MCMC: given p MCMC finds **T**

How do we construct a Markov chain whose stationary distribution is our target distribution, $\pi(x)$?

Metropolis-Hastings algorithm

At each iteration t

Step 1 Sample $y \sim q\left(y|x^{(t)}\right)$.



Step 2 With probability

$$\alpha(x^{(t)}, y) = \min\left\{1, \frac{\pi(y)q\left(x^{(t)}|y\right)}{\pi\left(x^{(t)}\right)q\left(y|x^{(t)}\right)}\right\}$$

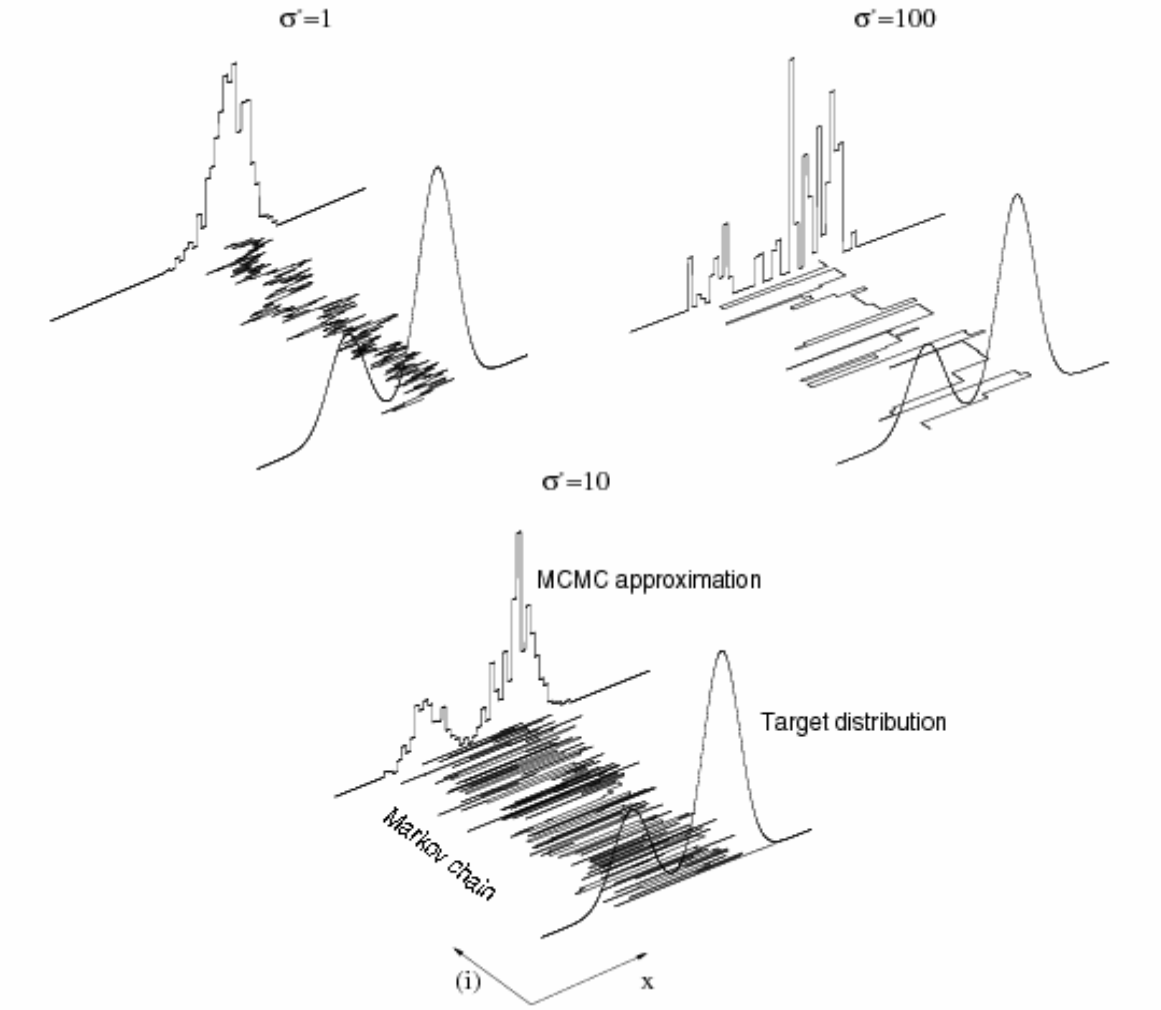
set

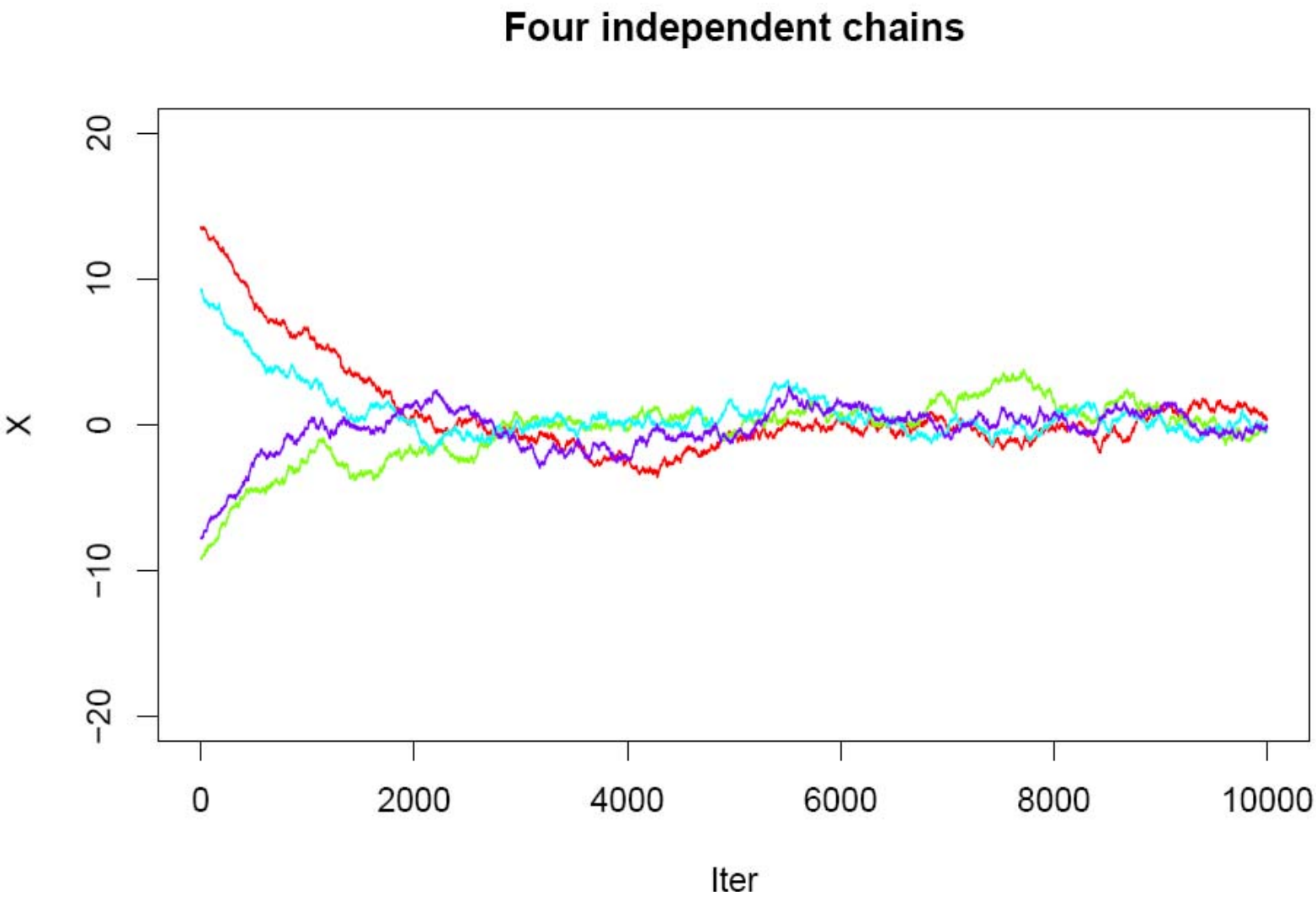
$$x^{(t+1)} = y \quad \text{(acceptance)},$$

else set

$$x^{(t+1)} = x^{(t)} \quad \text{(rejection)}.$$

Examples of M-H simulations with q a Gaussian with variance sigma




Four independent chains
X
20
10
0
−10
−20
0
2000
4000
6000
8000
10000
Iter

Bayesian Inference

Data: Y (realisation y)

Parameters, latent variables:

$\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_p)$

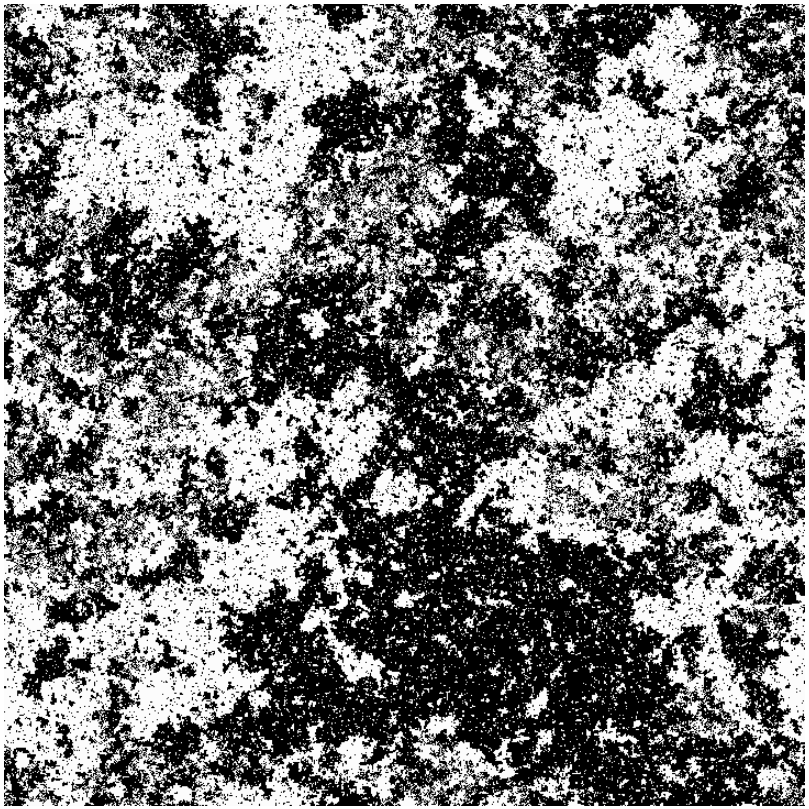
Likelihood: $L(y|\boldsymbol{\theta})$

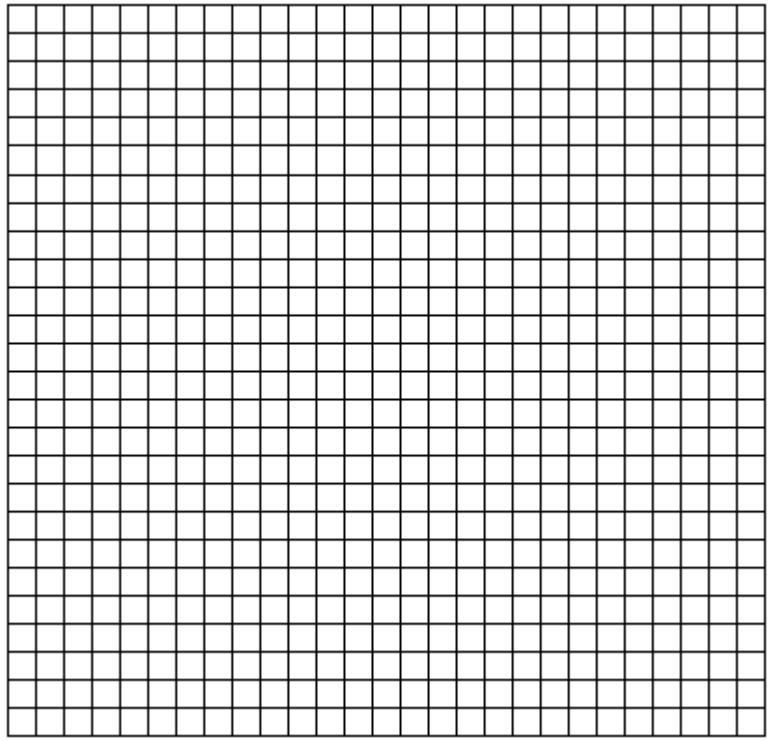
Prior: $\pi_0(\boldsymbol{\theta})$

Inference is based on the *joint posterior*

$$\begin{aligned} \pi(\boldsymbol{\theta}|y) &= \frac{L(y|\boldsymbol{\theta})\pi_0(\boldsymbol{\theta})}{\int L(y|\boldsymbol{\theta})\pi_0(\boldsymbol{\theta})d\boldsymbol{\theta}} \\ &\propto L(y|\boldsymbol{\theta})\pi_0(\boldsymbol{\theta}) \end{aligned}$$

$$i.e. \quad Posterior \propto Likelihood \times Prior$$





Ising model

- on each site/pixel i there is a random variable X_i talking values +1 or -1
- there is a preference for each variable to be like its neighbours

$$P(\mathbf{X}) = \frac{1}{Z_\beta} \exp\left\{\beta \sum_{\langle ij \rangle} \mathbf{1}_{\{X_i = X_j\}}(\mathbf{X})\right\}, \qquad \beta \geqslant 0$$

Signal: X, with c colours, on a 2D grid, with prior Ising model

Data: Y

Next we define the noise. Conditional on $\mathbf{X}$, at each pixel independently the colour $Y_i \in \{0, 1, \ldots, c-1\}$ is observed exactly with probability $1-\epsilon$, whereas with probability $\epsilon/(c-1)$ it is switched to any of the remaining colours:

$$P_i(Y_i|X_i) = \begin{cases} 1-\epsilon & \text{if } Y_i = X_i, \\ \dfrac{\epsilon}{c-1} & \text{otherwise.} \end{cases} \tag{2.3}$$

Posterior:

$$L(\mathbf{X}|\mathbf{Y}) = \beta \sum_{\langle ij \rangle} \mathbf{1}_{\{X_i = X_j\}}(\mathbf{X}) + h \sum_{i \in \Lambda} \mathbf{1}_{\{X_i = Y_i\}}(\mathbf{X})$$

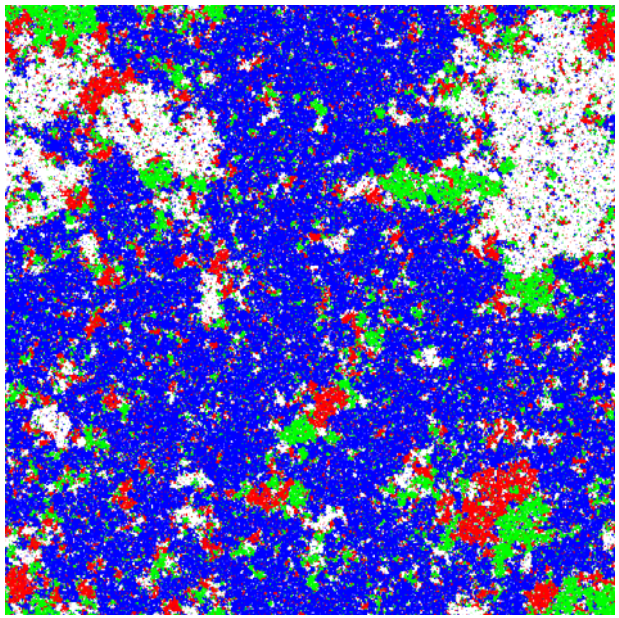
MAP estimate of X, obtained by maximisation of the posterior.
(X is in $c^{10000 \cdot 10000}$ dimensional space)

Parallel versions.

4 codings to a 2 colour image

$0 \rightarrow 0$, $1,2,3 \rightarrow 1$
$1 \rightarrow 0$, $0,2,3 \rightarrow 1$
$2 \rightarrow 0$, $0,1,3 \rightarrow 1$
$3 \rightarrow 0$, $0,1,2 \rightarrow 1$

Data: Y



N=number of sites

- Solve four 2-colours MAP problems.
- Each is a min-cut problem.
- Can be solved in $O(N^2)$

Parallel versions.

$0 \rightarrow 0$, $1,2,3 \rightarrow 1$
$1 \rightarrow 0$, $0,2,3 \rightarrow 1$
$2 \rightarrow 0$, $0,1,3 \rightarrow 1$
$3 \rightarrow 0$, $0,1,2 \rightarrow 1$

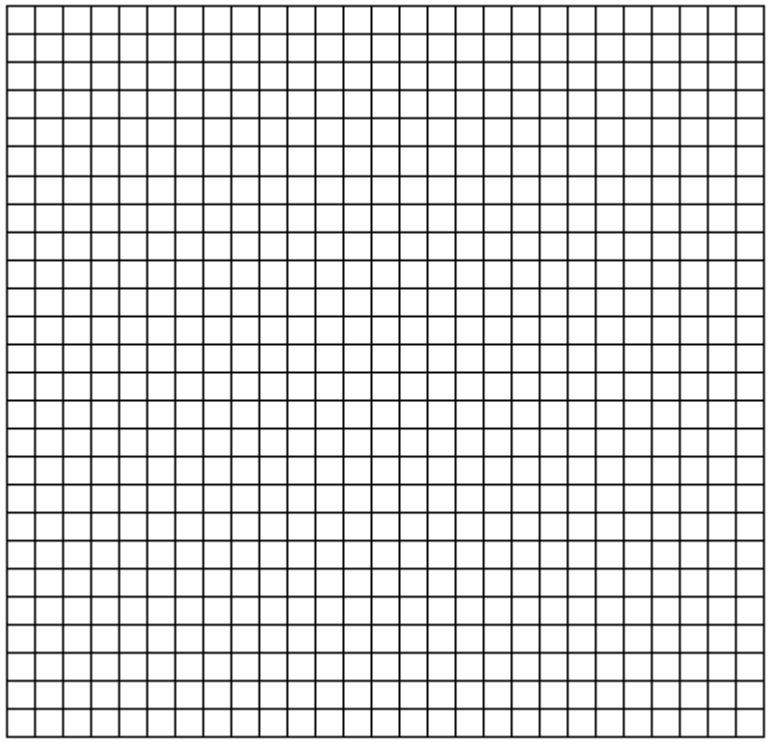
- Solve four 2-colours MAP problems.
- Whenever the solution corresponds to 0 in one of the 2-colour reductions, then it is also globally optimal
- Pixels that are not resolved, call them “grey”, are “difficult” and need a second look

Second look:

- Work in parallel on each “grey” island, with given boundary conditions
- Do full enumeration if island is small, simulated annealing on island otherwise.

β	ϵ	*New algorithm:* % *of ?s*
1.4	0.2	0.1
	0.3	0.0
	0.4	0.0
1.2	0.2	3.1
	0.3	4.4
	0.4	7.6
	0.5	100.0
1.0	0.1	6.9
	0.2	17.5
	0.3	35.6
	0.4	72.9
0.7	0.3	42.0

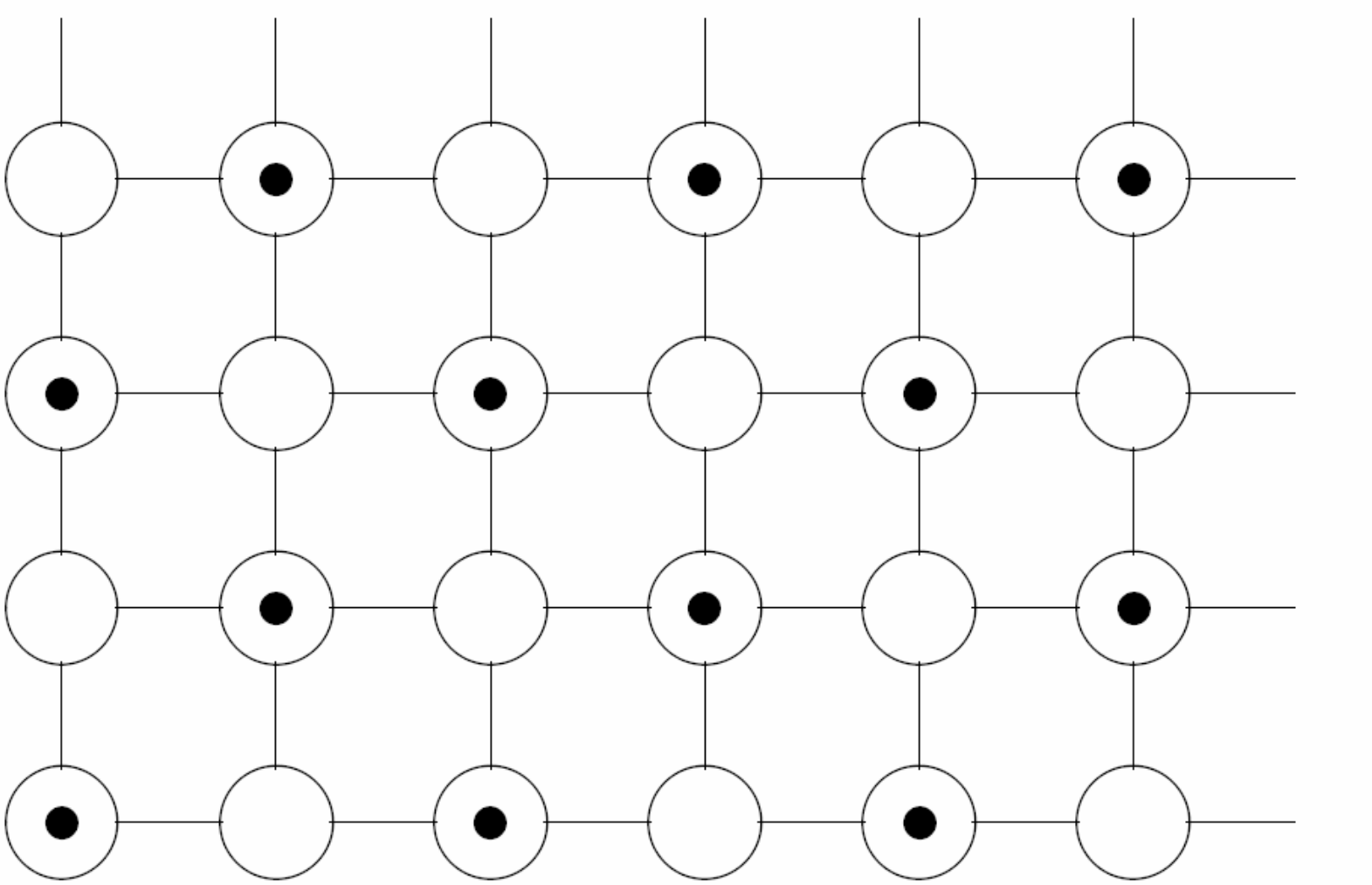
Islands are mostly border areas between resolved parts.
They are “long”, with much boundary (since otherwise they would have been easy to resolve).
Hence easy to solve exactly.



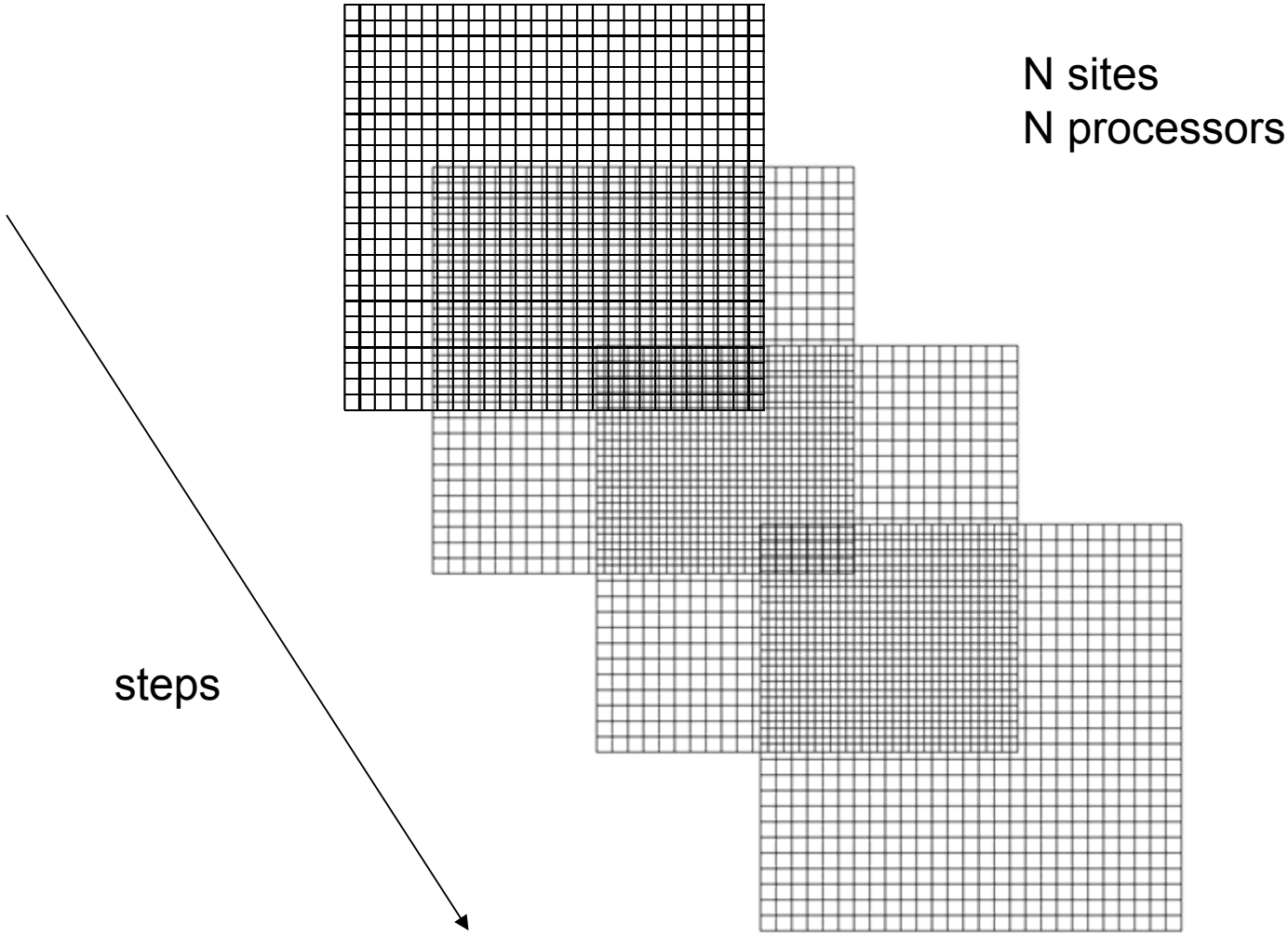
$$P(\mathbf{X}) = \frac{1}{Z_\beta} \exp\left\{\beta \sum_{\langle ij \rangle} \mathbf{1}_{\{X_i = X_j\}}(\mathbf{X})\right\}, \qquad \beta \geqslant 0$$

Ising model

- Variables on “even” sites are conditional independent given the variables on the odd sites. (Colouring of the graph)
- Hence we can put a processor on each site and update each even variable in parallel.

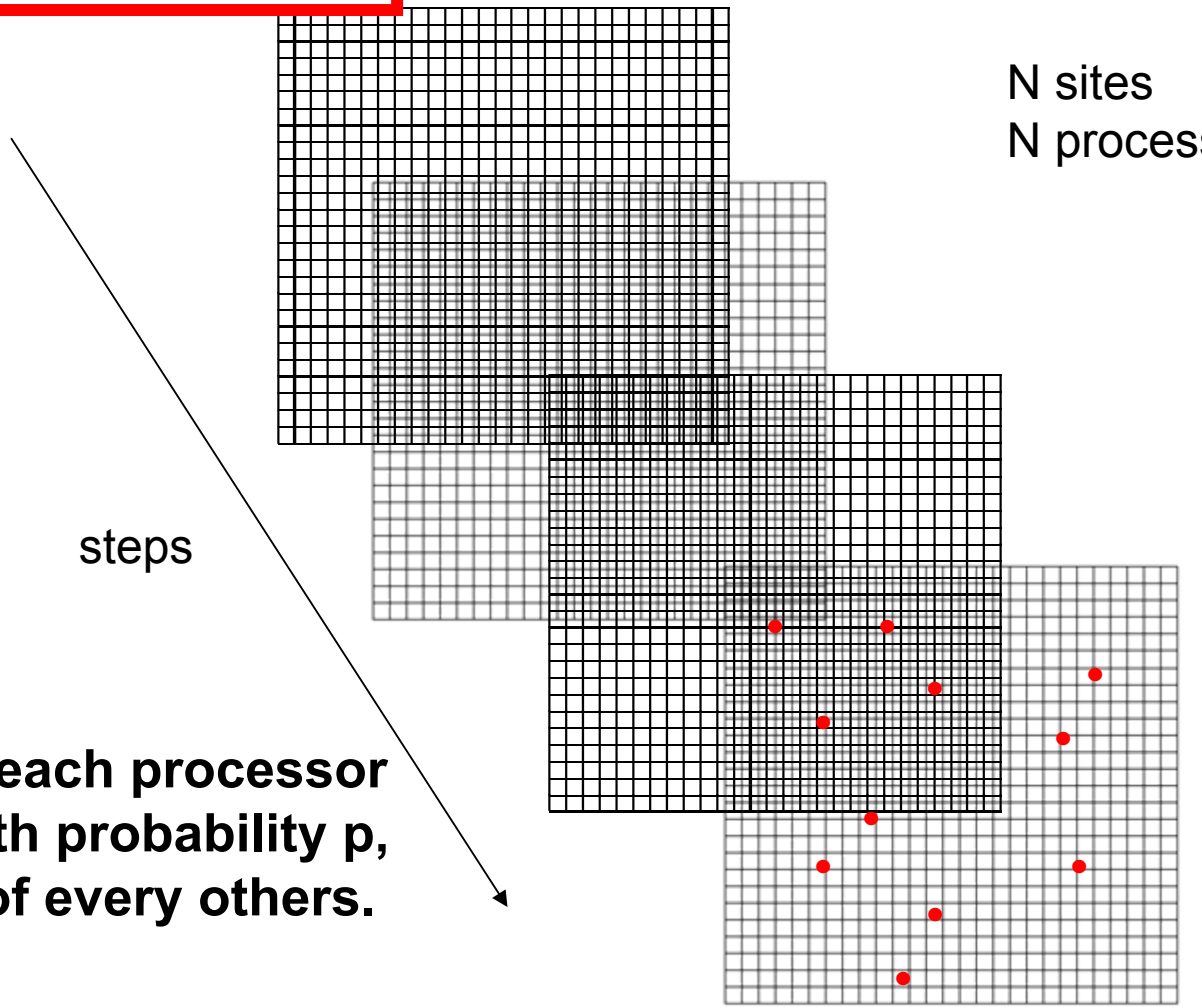


Can we do full parallel updating?



Theorem: Does not converge!

p-pertubed fully parallel MCMC



At each step, each processor is silenced with probability p, independent of every others.

Theorem: Does converge for every p, so small as you like !

Making MCMC parallel for huge data sets

Marit Holden

Norsk Regnesentral

TransCount

- MCMC
- Estimates of absolute transcript concentrations from spotted microarray data
- Hierarchical Bayesian model for microarray experiments
- MCMC to sample from the posterior model

Arnoldo Frigessi, Mark A. van de Wiel, Marit Holden, Debbie H. Svendsrud, Ingrid K. Glad and Heidi Lyng.
Genome-wide estimation of transcript concentrations from spotted cDNA microarray data.
Nucleic Acids Research, 2005;33(17):e143.

TransCount

- Huge data sets
  - Estimate lots of parameters - typically 10^6
- Main problem with the TransCount method
  - Not memory usage, but CPU-time
  - CPU-times of several days or weeks
- A typical MCMC run:
  - 10^5-10^6 iterations - each with 10^6 updates
    - Mixing and convergence
    - Sufficiently many samples from the posterior
- Difficult to test program and model, debug

The hierarchical model (gaussian approximation)

$$J_s^{d,a} \sim \text{Normal}\left(\mu_s^{d,a} , \mu_s^{d,a} \cdot \left(1 - \min\left(1, \exp\left(\gamma_0 + \gamma_{1,g(s)} + \overline{\beta X_s}\right)\right)\right) \cdot \alpha_0 \right)$$

$$D_s^{d,a} \sim \text{Normal}\left(C_{2,s}^{d,a} \cdot J_s^{d,a} \cdot \alpha_1^{d} , C_{3,s}^{d,a} \right)$$

$$\overline{\beta X_s} = \beta_3 \cdot C_{4,s} + \beta_4 \cdot C_{5,s} + \beta_{5,RID(s)} + \beta_{6,PID(s)} + \beta_{7,p(s)}$$

$$\mu_s^{d,a} = C_{1,s}^{d,a} \cdot K_{g(s)}^{t(d,a)} \cdot \exp\left(\beta_1^{a} + \beta_2^{d,a}\right) \cdot \exp\left(\overline{\beta X_s}\right)$$

$\gamma_{1,g(s)} \sim \text{Normal}(0, \sigma_1^{\,2})$,
$\beta_{5,\text{RID(s)}} \sim \text{Normal}(0, \sigma_5^{\,2})$, $\beta_{6,\text{PID(s)}} \sim \text{Normal}(0, \sigma_6^{\,2})$, $\beta_{7,p(s)} \sim \text{Normal}(0, \sigma_7^{\,2})$

d=1,2
typically a= 1,2,…, 100

- A single-update random-walk Metropolis-Hastings sampler
- Uniform proposals
  - Each parameter has its own interval
    - Interval includes the current parameter value
    - New parameter value drawn from this interval
  - Interval lengths updated adaptively before burn-in
    - Acceptance rates around 0.3
- Parallel random number generator
  - SPRNG - The Scalable Parallel Random Number Generators Library for ASCI Monte Carlo Computations

Parallelization

- Parallelized the program using Message-Passing Interface (MPI)
- The first half of the processes computes expressions, probabilities etc. for d=1
  - The other half computes those for d=2
- a-indexes divided between processes for d=1
  - Similarly for d=2
- Expressions for one (d,a)-pair computed by only one process
- If each process has the same number of (d,a)-pairs:
  - Each process performs approximately the same number of computations between each process synchronization or inter-process communication

Titan Cluster

- A 380 CPU Intel Xeon cluster
- The 64 bits CPUs are 3.4 GHz
- Each dual CPU node has
  - 2 or 4 Gb memory
  - Two 80 Gb local disks
  - Option for InfiniBand interconnect
  - A central GPFS based storage capacity of 4.5 Tb are connected to all nodes

Example

- $d \in \{1,2\}$, $a \in \{1,2,\ldots,40\}$, $s \in \{1,2,\ldots,14000\}$
- There are around $1.2 \cdot 10^6$ unknown parameters
- Local disks were used
- InfiniBand and 1 gigabit interconnect
- Obtain around eight times less time usage

Number of processors used	Observed time usage for 80 iterations	Estimated time usage for 250 000 iterations	Per cent of observed time usage for the one-processor run
1	110 seconds	95.5 hours	100%
2	59 seconds	51.2 hours	53.6%
4	35 seconds	30.4 hours	33.6%
8	23 seconds	20.0 hours	20.9%
20	14 seconds	12.2 hours	12.7%
40	14 seconds	12.2 hours	12.7%

Why not more speed-up?

- Between each process synchronization or inter-process communication:
  - Approximately the same number of computations
- Discover new bottlenecks in the code when parallelizing
- For 20 and 40 processors:
  - Improved results with InfiniBand
  - Communication between processes a bottleneck?

Conclusion

- It's hard work to parallelize within an iteration;
- Communication overhead will eventually kill of speed gains;

(BUT: This makes MCMC possible at all for large problems!)

More unsupervised parallel MCMC

Parallel Tempering,

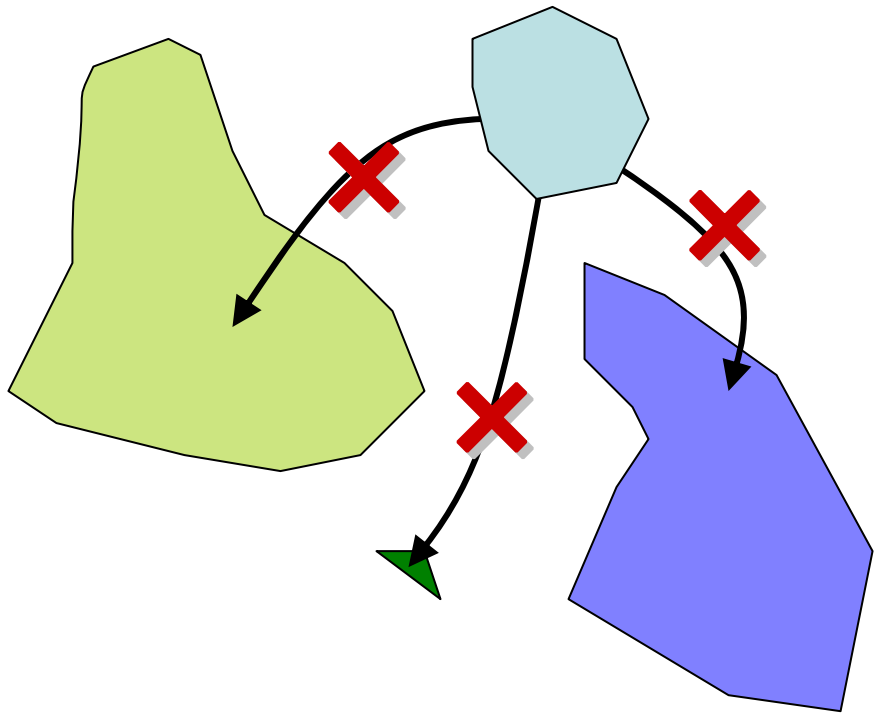
Simulated Tempering,

Multicanonical Monte Carlo,

Equi-Energy sampler,

Nested Sampling

Slow mixing due to multimodality

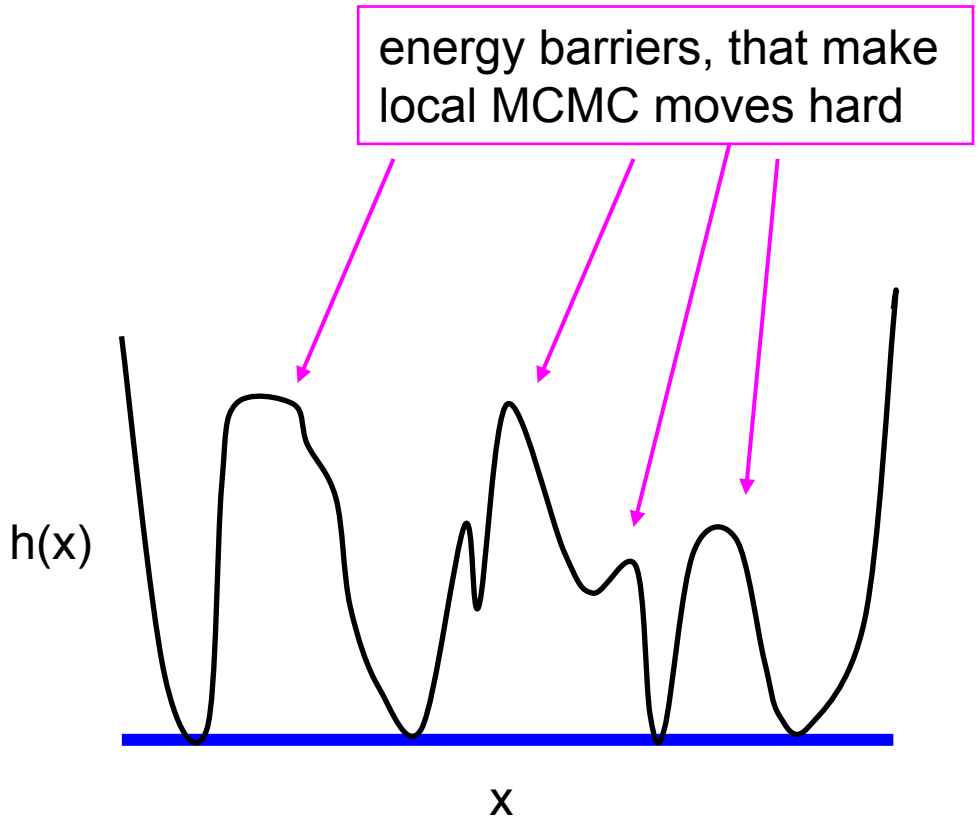


$$P(\mathbf{X}) = \frac{1}{Z_\beta} \exp\left\{\beta \sum_{\langle ij \rangle} \mathbf{1}_{\{X_i = X_j\}}(\mathbf{X})\right\}, \qquad \beta \geqslant 0,$$

Gibbs models – Exponential family

p(x,T) = (1 / Z(T)) exp (- h(x) / T),

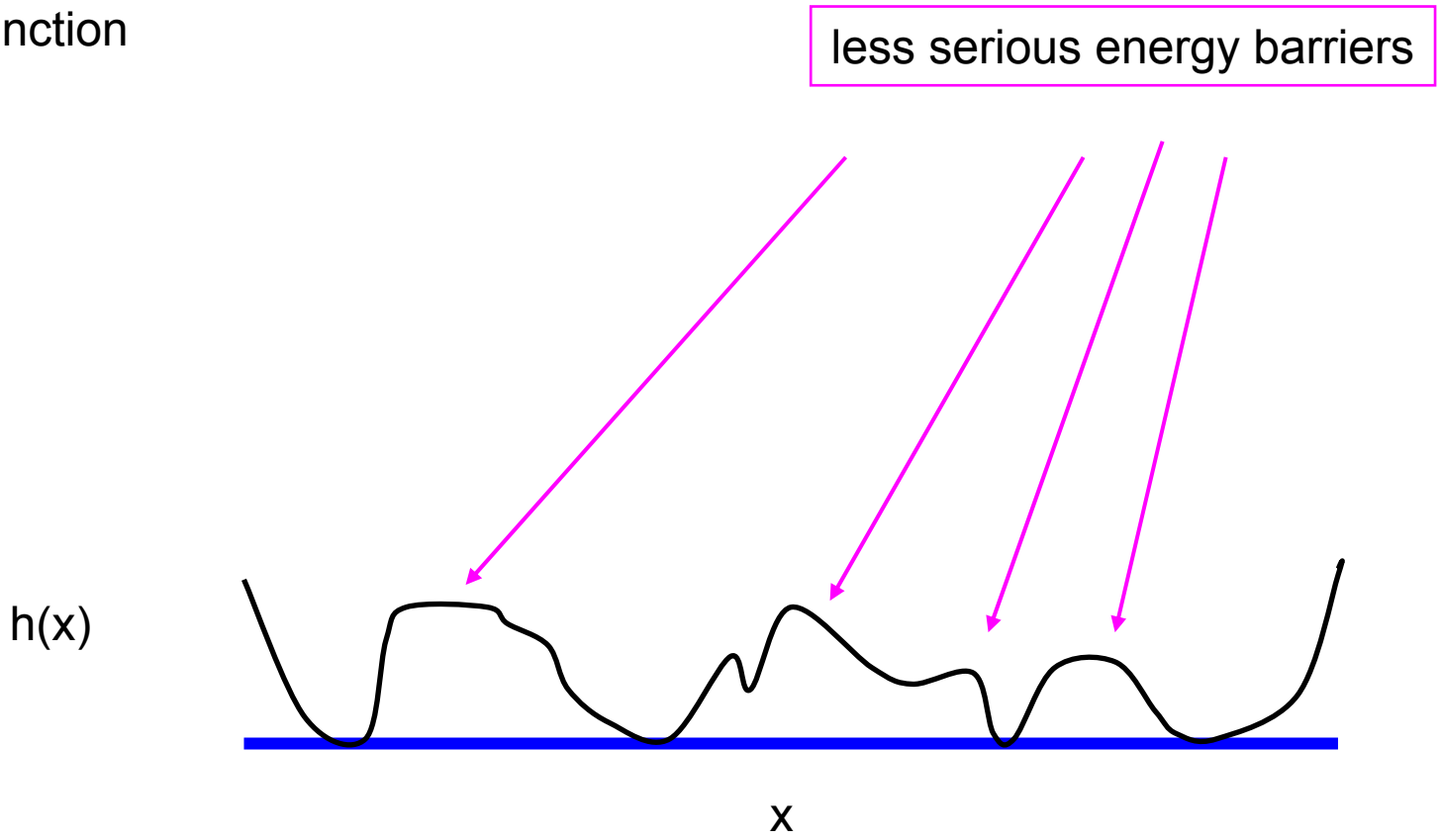
h(x) energy function, h > 0
T temperature
Z(T) partition function



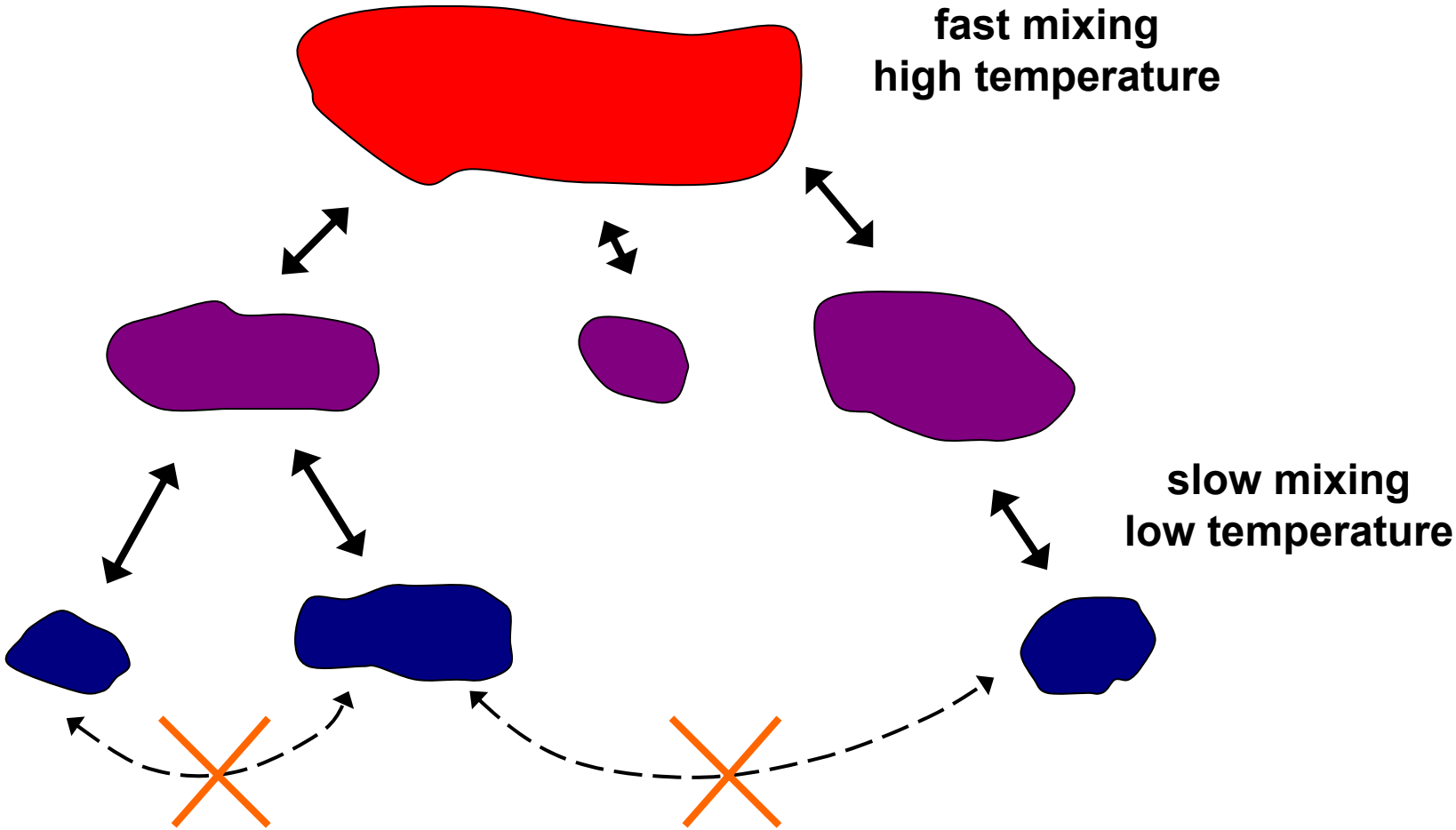
Higher T temperature smoothes energy landscape!

$p(x,T) = (1 / Z(T)) \exp(- h(x) / T)$,

$h(x)$ energy function, $h > 0$
T temperature
$Z(T)$ partition function



Extended space



Parallel Tempering

Geyer (1991),
Kimura and Taki (1991)

- Simulate many “replica” in parallel at different temperatures
- Switch states between samples, to help the chains that are stuck
- MCMC in a Product Space

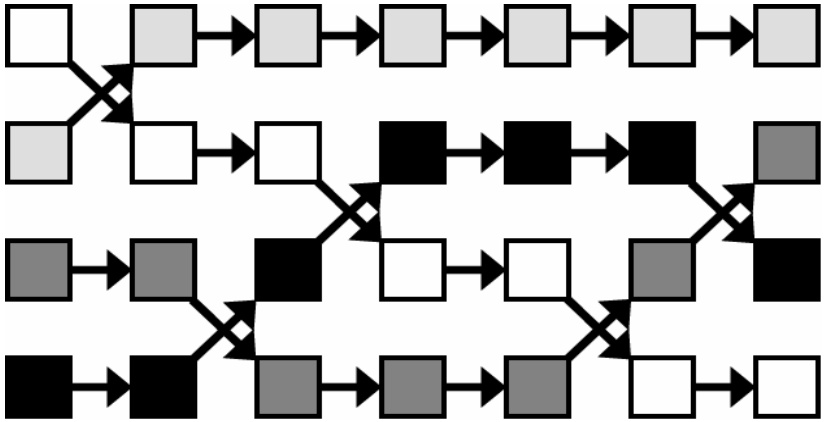
Exchange of Replicas

K=4

easy, high T

harder ↓

original chain with T=1



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

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

Uses faster mixing of the high temperature chains to achieve faster mixing of the low temperature chains, including the original one.

Easy to parallelize.
But care to synchronize is needed.
(though different time scales are ok.)

Simulated Tempering

Marinari & Parisi, 1992

Uses just one chain, that switches between easy and harder temperatures. Use samples generated when using the original temperature for inference at the end.

Not easy to parallelize!

Density of States $D(h_0)$

The number of states x such that $h(x) = h_0$.

The conditional distribution of x, given $h(x) = h_0$ is uniform on the equi-energy set $\{ x : h(x) = h_0 \}$.

Density of States $D(h_0)$

Suppose that the infinitesimal volume of the energy slice $\{ x : h < h(x) = h + dh\}$ is $D(h)\, du$. Then $D(h)$ is the density of states.

$$Z(\beta) = \sum_x \exp(-\beta E(x))$$

from Riemann to Lebesgue MCMC

$$Z(\beta) = \sum_E \exp(-\beta E) D(E)$$

Equi-Energy Sampler

Kou, Zhou, Wong, Annals of Statistics, 2006

Sample from

$p(x,T=1) = (1 / Z) \exp (- h(x))$,

to approximate

$E(g(X)) = \sum g(x)\, p(x)$

Equi-Energy Sampler

Auxiliary models (again)

$p(x,T) = (1 / Z(T)\ )\ exp\ (\ -\ h(x) / T)$

Introduce a sequence of energy levels and associated temperatures

$$H_0 < H_1 < H_2 < \ldots < H_K$$

HARD

$$T_0=1 < T_1 < T_2 < \ldots < T_K$$

EASY!

where $H_0 < min\ \{h(x)\}$

Auxiliary models:

$p_i(x,T) = (1 / Z(T_i)\ )\ exp\ (\ -\ max\ \{\ h(x),\ H_i\ \} / T_i\ )$

Note that $p_0(x,T=1)$ is the original model of interest!

Equi-Energy Sampler

$H_0 < H_1 < H_2 < \ldots < H_K$
$T_0 = 1 < T_1 < T_2 < \ldots < T_K$

$p_i(x,T) = (1 / Z(T_i)) \exp (- \max \{ h(x), H_i \} / T_i)$

Note that $p_0(x,T=1)$ is the original model of interest!

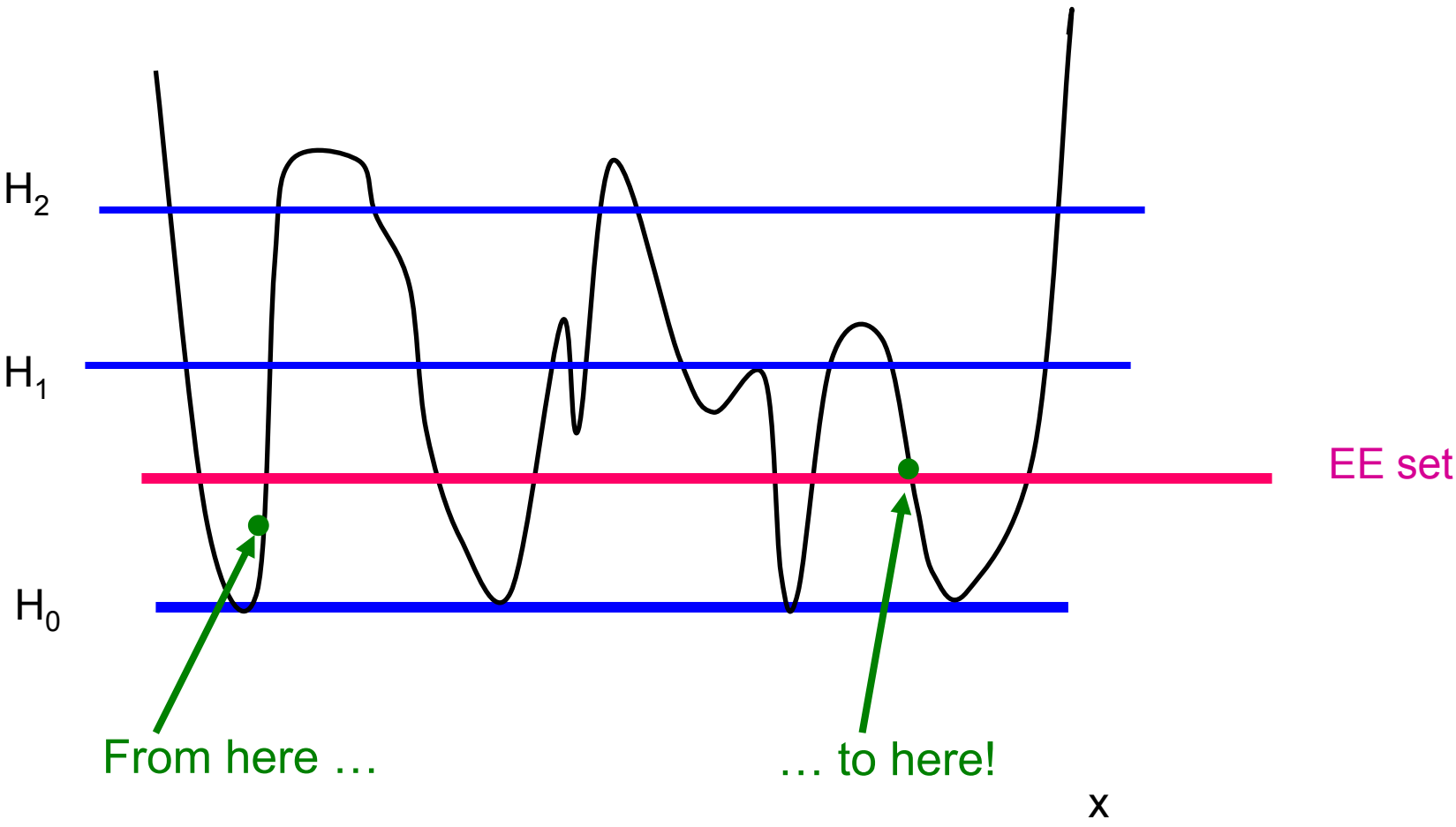
EE uses K MCMC chains, each targeting an auxiliary model $p_i(x,T)$.

The higher the value of i, the easier the mixing.

EE makes a special step, the EE-jump, to help chains with low i to move. It is a jump to a state somewhere else with similar energy value.

EE-sets $S(h_0) = \{ x : h(x) = h_0 \}$.

EE-sets are such for every auxiliary model. Hence they can be shared between MCMC chains.


H_2
H_1
H_0
EE set
From here …
… to here!
x

Parallel Equi-Energy Sampler

Each EE MCMC chains can go in parallel, with sharing of EE sets.

Not necessary that the chains are synchronous.

But EE-jumps should be avoided, if EE-sets are not well approximated yet.

Improved parallel strategies

Task decomposition and assignment
Orchestration
Load Balancing
Communication
Asymmetric parallel algorithm
… and much more to come