4. Cluster update algorithms

- Cluster update algorithms are the most succesful *global update* methods in use. These methods update the variables globally, in one step, whereas the standard local methods operate on one variable at a time.

- Another common method: **multigrid**. Used a lot in solving differential equations; however, multigrid has very limited success in Monte Carlo simulations.

- A global update can reduce the *autocorrelation time* of the update and thus greatly reduce the statistical errors.

- To repeat, we recall that the expectation values ($\beta \sim 1/T$):

$$\langle O \rangle_T = \frac{1}{Z} \int \prod_x d\phi_x \, O(\phi) \, e^{-\beta E(\phi)}$$

- With Monte Carlo simulation: generate a series of configurations $\Phi_1, \Phi_2 \ldots \Phi_N$ with some MC method. Measure $O_i = O(\Phi_i)$. Now

$$\frac{1}{N}\sum_i O_i \to \langle O \rangle, \quad \text{when } N \to \infty$$

- At finite N, the estimated error:

$$\delta O = \sqrt{\frac{\sum_i (O_i - \langle O \rangle)^2}{N(N-1)}}$$

  if and only if the measurements O_i are statistically independent.

- However, in reality the update modifies the previous configuration (Markov chain). Thus, the successive configurations are correlated: $C(t) \propto \exp -t/\tau$. $\tau =$ autocorrelation time.

- # of independent configs is $\sim N/(2\tau) \to \delta O \approx \sqrt{2\tau}\delta O_{\rm naive}$.

- τ depends on the MC update algorithm: we want an update with as small τ as possible $\to$ Cluster algorithms.

4.1. Fundamentals: Fortuin-Kasteleyn cluster decomposition

Ising model (arbitrary dim.):

$$E = -\sum_{<i,j>} s_i s_j \qquad\qquad Z = \sum_{\{s\}} e^{-\beta E} \, .$$

($s_i = \pm 1$, $< i, j >$ nearest neighbour sites).

Note: previously we defined E with a factor $\frac{1}{2}$ in front. Thus, $\beta_{\text{here}} = \beta_{\text{prev.}}/2$, and in 2 dimensions $\beta_c = 1/2 \log(1+\sqrt{2}) \approx 0.44$. This way is more conventional; the transformation between normalizations is trivial.

Consider interaction between fixed n.n.-sites $< l, m >$, and remove it from E:

$$E_{l,m} = -\sum_{<i,j> \neq <l,m>} s_i s_j \, .$$

Define now partition functions where s_i, s_j are equal or different:

$$Z^{\mathrm{same}}_{l,m} \equiv \sum_{\{s\}} \delta_{s_l,s_m} e^{-\beta E_{l,m}} \,, \qquad Z^{\mathrm{diff.}}_{l,m} \equiv \sum_{\{s\}} (1-\delta_{s_l,s_m}) e^{-\beta E_{l,m}} \,.$$

Now we can clearly write the original partition function as

$$Z = e^{\beta} Z^{\mathrm{same}}_{l,m} + e^{-\beta} Z^{\mathrm{diff.}}_{l,m}$$

Furthermore, defining

$$Z^{\mathrm{ind.}}_{l,m} \equiv \sum e^{-\beta E_{l,m}} = Z^{\mathrm{same}}_{l,m} + Z^{\mathrm{diff.}}_{l,m} \,.$$

the partition function finally becomes

$$Z = (e^{\beta} - e^{-\beta}) Z^{\mathrm{same}}_{l,m} + e^{-\beta} Z^{\mathrm{ind.}}_{l,m}$$

Remember that the partition function is $Z = \sum_{\{s\}} p(\{s\})$. Thus, since Z^{same} contains only configs where $s_l = s_m$, and $Z^{\mathrm{ind.}}$ contains no restriction or input for s_i, s_j-link, the weighting factors of Z's can be considered

as relative probabilities of a ***bond*** between sites l, m and no bond (= independent states).
Normalizing the probabilities: $p^{\text{bond}} = 1 - e^{-2\beta}$, $p^{\text{ind.}} = e^{-2\beta}$.

Repeating for all $< i, j >$,

a) sites linked to each other by bonds form clusters

b) different clusters are independent: $s_c = \pm 1$

and the partition function can be written as

$$Z = \sum_{\text{bonds}} \sum_{s_c} p^b (1-p)^n = \sum_{\text{bonds}} p^b (1-p)^n 2^{N_c}$$

Here

$$\begin{aligned} p &= 1 - e^{-2\beta} \\ b &= \textbf{\# of interactions that form a bond}, \\ n &= \textbf{\# of interactions that do not form a bond}, \\ N_c &= \text{\# of clusters.} \end{aligned}$$

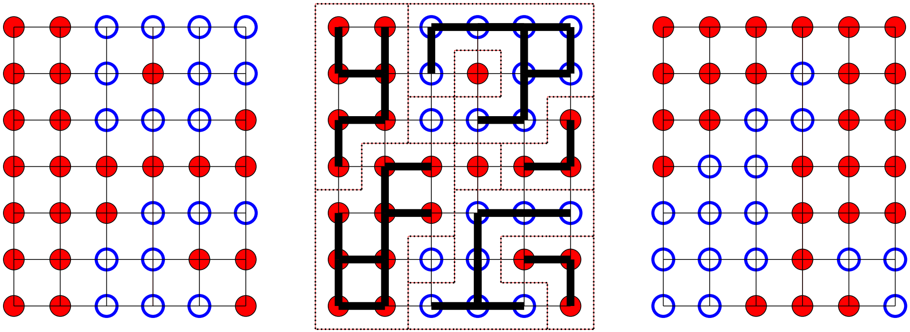
This decomposition is at the core of the cluster algorithms.

4.2. Swendsen-Wang cluster update

[Swendsen and Wang, PRL 58 (1987) 86]

Beginning with an arbitary configuration s_i, one SW cluster update cycle is:

1. Inspect all nn-states s_i, s_j. If $s_i = s_j$, create a bond between sites i, j with probability $p = 1 - \exp(-2\beta)$ (otherwise no bond).
2. Construct clusters $=$ sets of points connected by bonds.
3. Set each cluster to a *random* value ± 1.



The configuration changes a lot in one update!
Is this a valid update? It satisfies

a) ergodicity (obvious)

b) detailed balance: $\dfrac{P(A \mapsto B)}{P(B \mapsto A)} = \exp -\beta(E_B - E_A)$?

Proof: consider $A \mapsto \mathcal{C} \mapsto B$, where $\mathcal{C}$ is some bond configuration compatible with both A and B. Since the clusters in $\mathcal{C}$ are independent, $P(\mathcal{C} \mapsto A) = P(\mathcal{C} \mapsto B) = 1/2^{N_c}$.

Now,

$$\frac{P(A \mapsto \mathcal{C})}{P(B \mapsto \mathcal{C})} = \frac{p^b\ (1-p)^{d_A}}{p^b\ (1-p)^{d_B}} = \exp[-\beta(E_B - E_A)]$$

where $d_{A,B}$ are the numbers of similar nn-states which are not connected by a bond. The last step comes from $E_A = \mathrm{dim} \times V - 2(b + d_A)$. Thus $A \mapsto \mathcal{C} \mapsto B$ and $B \mapsto \mathcal{C} \mapsto A$ satisfy detailed balance for arbitrary $\mathcal{C}$, and the total transition probabilities $A \mapsto B$, $B \mapsto A$ must do it also.

4.3. Wolff single cluster update

[U. Wolff, PRL 62 (1989) 361]

Principle: do the cluster decomposition as in S-W, but invert ('flip') only one randomly chosen cluster! In practice:

1. Choose random site i.
2. Study neighbouring sites j. If $s_j = s_i$, join site j to cluster with probability $p = 1 - \exp(-2\beta)$.
3. Repeat step 2 for site j, if it was joined to the cluster. Keep on doing this as long as the cluster grows.
4. When the cluster is finished, invert the spins which belong to it.

• Usually slightly more effective than S-W (the average size of the clusters is larger. Why?).
• The minimum cluster size = 1, maximum = volume.
• Nicely recursive.
• Satisfies detailed balance.

Cluster update in c-pseudocode:

On the next page is a c-code snippet which performs Wolff cluster update.

- `s[loc]`: spin array, and `dran()` is a random number generator which returns a number fro 0 to 1.
- `n_neighbours =` $2 \times d$ is the number of neighbours a site has.
- `neighbour[i][loc]` is the neighbour array: it gives the index of the site `loc` to direction `i`.
- `s[loc]` is flipped during the cluster growth: this prevents revisiting sites already included in the cluster.
- Recursion (as shown) is neat but not efficient! Better to unroll the recursion with loops and temporary arrays.

(Besides, there are unnecessary variables pushed on the stack: `state`, which is constant, and `new_loc`, which is just unnecessary).

```
void update_cluster()
{
  int start,state;

  start = dran() * volume;    /* starting location */
  state = s[start];           /* starting spin value */
  /* start growing and inverting the cluster */
  grow_cluster(start, state);
}

void grow_cluster(int loc,int state)
{
  int i,new_loc;

  s[loc] = -s[loc];        /* invert the spin at this location */

  /* begin loop over neighbour locations */
  for (i=0; i<n_neighbours; i++) {
    new_loc = neighbour[i][loc];   /* neighbour index */

    if (s[new_loc] == state && exp(-2.0*beta) < dran())
      grow_cluster(new_loc,state);
  }
}
```

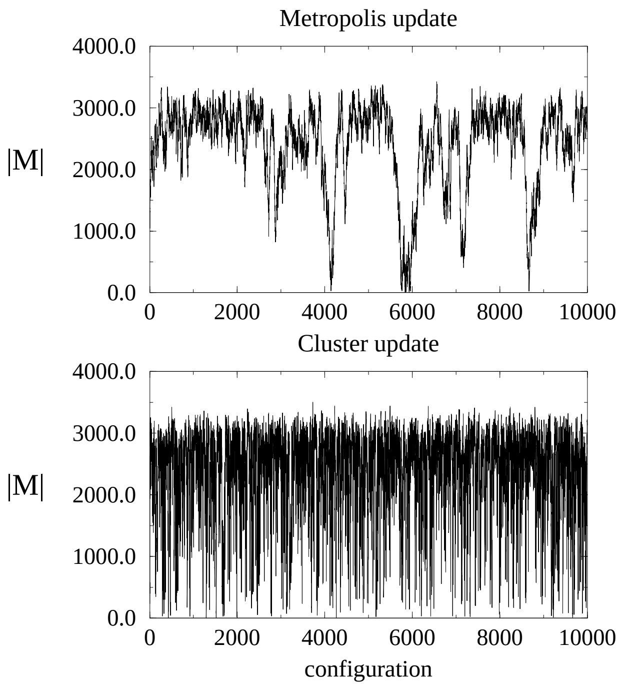
As an example, let us consider 2-dimensional Ising model. It has 2nd order phase transition at $\beta = 1/2 \log(1+\sqrt{2}) \approx 0.44$ (Curie point). Correlation length (and cluster size!) diverges when one approaches the critical point.

Compare the performance of
a) Metropolis (typewriter ordering) and
b) Wolff cluster algorithm.

System size 64^2, $\beta = \beta_c \approx 0.44$. Compare measurements of absolute value of magnetization

$$|M| = \left|\sum_i s_i\right|$$

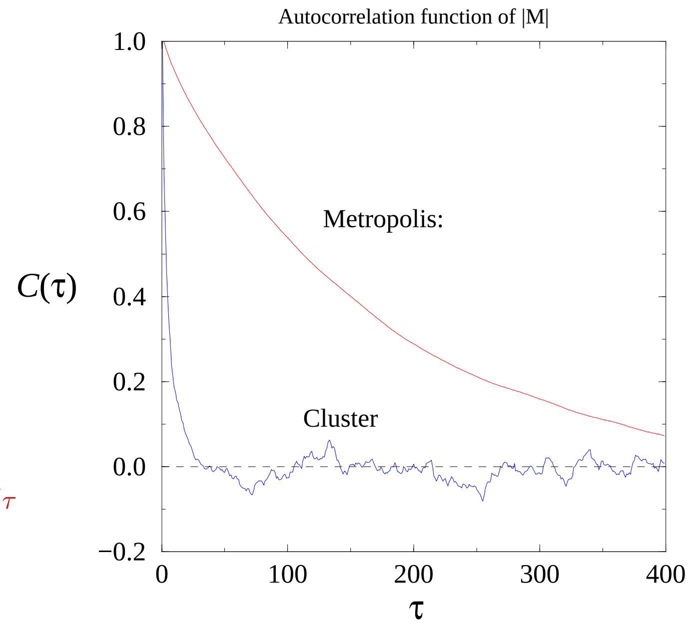
Cluster update covers the phase space much faster than Metropolis! (which was the best of the "single-site" update methods).



The autocorrelation function describes how fast measurements become decorrelated.

Let now $O_i = |M|_i - < |M| >$, where $|M|_i$ is the measurement $\#i$ of $|M|$. Autocorrelation function $C(t)$ of $|M|$:

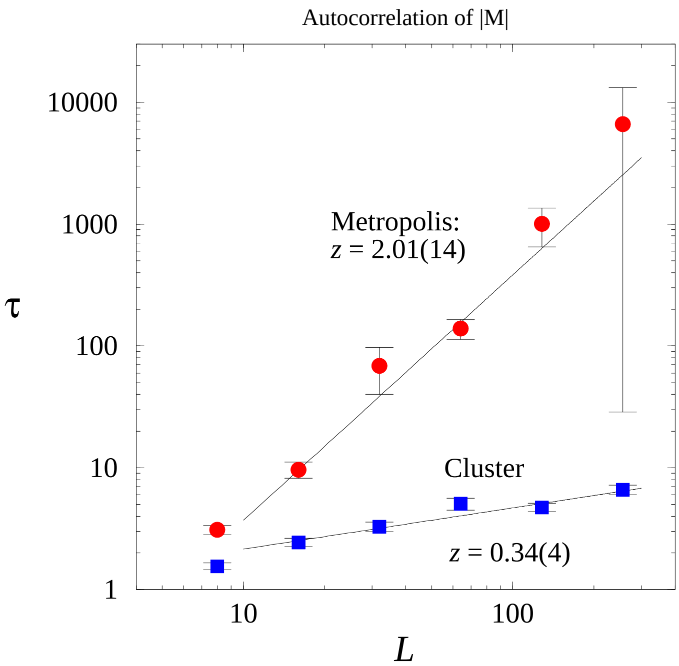
$$C(t) \equiv \frac{\frac{1}{N-t} \sum_i O_i O_{i+t}}{\langle O^2 \rangle} \propto e^{-t/\tau}$$



• Exponential autocorrelation time: the exponential decay length of $C(t)$ at large t.

• Integrated autocorrelation time: $\tau_{\text{int.}} \equiv \frac{1}{2} + \sum_{t=1}^{\infty} C(t)$.

Shown here is the integrated autocorrelation time $\tau_{\text{int.}}$ measured from different lattice sizes $L^2 = 8^2$ – 256^2 at β_c.



• Expected behaviour: $\tau_a \propto L^z$, where z is the dynamical critical exponent. For local (dissipative) algorithms, $z \gtrsim 2$.

• For volumes $\sim 256^2$, cluster algorithm is ~ 1000 times better than Metropolis!

• Measured in real (cpu/wallclock) time, cluster is even better (at the critical temperature $\beta_c = 0.44$):
time/update $\propto V$ (local Metropolis)
time/update $\propto V^x$, where $x < 1$

Using the example of section 3.14, 2d Ising, volume 64^2, and measuring the integrated autocorrelation time of E:

	$\tau_{\text{int.}}$	time(ms)/iteration	time(ms)/iter. $\times 2\tau_{\text{int}}$
Cluster	5.7	0.28	3.2
Metro, typew.	54	1.0	108
HB, typew.	157	1.2	364
Metro, random	271	1.4	748
HB, random	316	1.6	1004

4.4. What about other models?

Cluster algorithms can work, if one can embed an Ising system into the original system. Example: O(N) non-linear sigma model:

The Hamiltonian of the model is

$$E = - \sum_{<ij>} \sigma_i \cdot \sigma_j \;, \qquad \text{where} \quad |\sigma_i|^2 = \sum_{\alpha=1}^{N} (\sigma_i^\alpha)^2 = 1$$

Thus, spins σ_i are N-dimensional vectors of unit length.
Embedding: [U.Wolff, PRL 62 (1989) 361]

Map arbitrary sigma model configuration to Ising model with Hamiltonian

$$E_I = - \sum_{<ij>} J_{ij} s_i s_j$$

as follows:

- Choose a random O(N) vector r.

- Set all Ising model links $J_{ij} = (r \cdot \sigma_i)(r \cdot \sigma_j)$.
- Ising update $s_i \rightarrow -s_i$ corresponds to reflection of σ_i along the vector r: $\sigma_i \rightarrow \sigma_i - 2(r \cdot \sigma_i)\, r$
- Initially, all $s_i = +1$.

After the mapping, we can use either S-W or Wolff update on the Ising spins. In practice, one Wolff update cycle proceeds as follows:

1. Choose random O(N) vector r.
2. Choose random site i as the starting point for the cluster.
3. Study neighbouring sites j. Join site j to cluster with probability $p = 1 - \exp(-2\beta J_{ij})$, where $J_{ij} = (r \cdot \sigma_i)(r \cdot \sigma_j)$.
4. Repeat step 3 for all sites joined to the cluster. Keep on doing this as long as the cluster grows.
5. When the cluster is finished, reflect $\sigma_i \rightarrow \sigma_i - 2(r \cdot \sigma_i)$ for all sites which belong to the cluster.

Typically the reflection is performed during the cluster growth.

Single cluster O(N) sigma model algorithm works even better than for Ising: In 3-dimensional O(4) sigma model, the dynamical critical exponent z becomes negative.

In what models clusters work, in what models they fail?

Clusters (usually) fail, if

- there are frustrated couplings (spin glasses, gauge theories . . .)
- one cannot construct a symmetric reflection operation
- spins are 'frozen' in place by external fields etc.

Cluster updates are (normally) usable only in proximity of a 2nd order phase transitions: large correlation lengths $\rightarrow$ large clusters.

Nevertheless, sometimes they are useful when correlation lengths are finite but still large ($\gg 1$) in lattice units.

More conditions:

In order to work, the reflection operation $R\sigma = \sigma'$ must satisfy the following: the set of the fixed points of the reflection ($R\sigma = \sigma$) must separate the phase space of σ in two disconnected sets.

For example, in O(3), the fixed points are the points with $r \cdot \sigma = 0$, i.e. the points on the 'equator' perpendicular to the reflection vector r. This divides O(3) into disconnected 'northern' and 'southern' hemisphere.

Mathematically: fixed point set should have codimension 1 [Lüscher, Weisz, Wolff, NPB 359 (1991) 221].

4.5. Cluster search

- The presented algorithm for growing the cluster is a ***depth-first*** algorithm: it starts from the root and follows the cluster tree branch as far as possible, before taking another branch.
- Another option is ***width-first***:
  1. Start from the root.
  2. Mark all neighbouring points which are accepted to the cluster.
  3. Looping through all these points, mark all points where the cluster grows further.
  4. Continue from 3. for these new points, until no more points are accepted.
- Both the depth-first and width-first require comparable amount of information, and are probably about as good. Width-first is somewhat easier to program, though.

- However, for Swendsen-Wang update all of the clusters have to be found. In this case there exists an optimized cluster search algorithm by Hoshen and Kopelmann.

4.6. *Reduced variance*

- Clusters can also substantially reduce statistical noise in some measurements – this is in addition to the acceleration in update speed (*reduced variance*).

- For example, consider spin-spin correlation function

$$G(i,j) = s_i s_j \qquad\qquad \langle G(i,j)\rangle \sim e^{-|i-j|/\xi}$$

- Using normal MC averaging over spin configurations (whichever algorithm we use), the variance of G is

$$\langle G^2\rangle - \langle G\rangle^2 \approx 1 - e^{2|i-j|/\xi} \approx 1$$

  Thus, the absolute error is $\sim$ constant independent of $|i-j|$, and signal/error$\sim e^{-|i-j|/\xi}$.

- However, consider the expectation value using the cluster partition function

$$\langle G(i,j)\rangle = \frac{1}{Z}\sum_{\text{bonds}} p^b(1-p)^n \sum_{s_{\text{clust.}}} s_i s_j$$

The last sum is clearly

$$\sum_{s_{\text{clust.}}} s_i s_j = \begin{cases} 1 & \text{if } s_i \text{ and } s_j \text{ belong to the same cluster} \\ 0 & \text{otherwise.} \end{cases}$$

- Thus, in a (Swendsen-Wang) cluster MC update/measurement, we measure

$$\langle s_i s_j\rangle_{SW} = \left\langle \sum_{\text{clust.}} \Theta_c(i)\Theta_c(j) \right\rangle_{SW}$$

where $\Theta_c(i) = 1$ if point i belongs to cluster c, 0 otherwise. Since the fraction of 1 to 0's must be $e^{-|i-j|/\xi}$, the variance is

$$(\langle G^2\rangle - \langle G\rangle^2)_{SW} \approx e^{-|i-j|/\xi} - e^{-2|i-j|/\xi} \approx e^{-|i-j|/\xi}$$

The absolute error $\sim e^{-|i-j|/2\xi}$, and signal/error$\sim e^{-|i-j|/2\xi}$.

- An exponential factor is gained over the 'naive method! This is due to the fact that the cluster measurement effectively sums over all possible spin configurations compatible with the cluster decomposition.

- Another example: magnetic susceptibility

$$\chi_M = \langle M^2 \rangle = \frac{1}{V} \sum_{i,j} s_i s_j = \frac{1}{V} \sum_{\text{clust.}} N_c^2$$

where N_c is the cluster size, and we assume $\langle M \rangle = 0$ ($T > T_c$).

- This readily generalizes to other spin observables. However, 3- or higher point functions do not usually gain from cluster measurements, nor observables which depend on energy. The operators rapidly become quite complicated, for example,

$$\langle s_i s_j s_k s_l \rangle = \theta(i,j,k,l) + \ \theta(i,j)\theta(k,l) + \theta(i,k)\theta(j,l) + \theta(i,l)\theta(j,k).$$

- The above was for Swendsen-Wang type update. What about the Wolff single cluster update? In this case, the clusters are chosen with biased probability $p_c = N_c/V$. This we can compensate by multiplying the observable with $1/p_c$, and thus

$$\langle s_i s_j \rangle_{1C} = \left\langle \frac{V}{N_c} \Theta_c(i) \Theta_c(j) \right\rangle_{1C}$$

- Likewise, the single-cluster susceptibility measurement becomes

$$\chi_M = \langle M^2 \rangle = \langle N_c \rangle_{1C}$$

- Through the Ising embedding, the reduced variance measurements can be done also for other models.