Advanced Monte Carlo algorithms

Bad Honnef Physics School
Computational Physics of Complex and Disordered Systemss

Werner Krauth

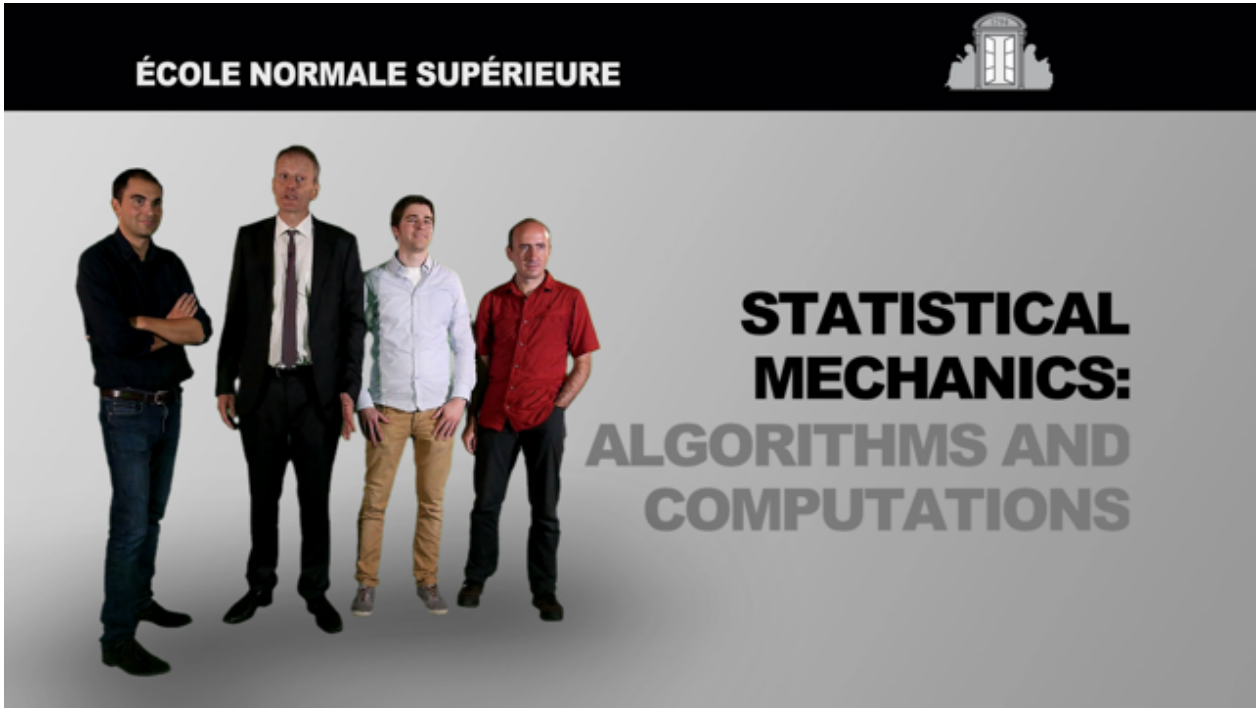
Département de Physique
Ecole Normale Supérieure, Paris, France

23 September 2015

- Monte Carlo algorithms - basic notions.
- Hard disks: From detailed balance to global balance and to cluster algorithms.
- Integration and Sampling: From Gaussians to Maxwell and Boltzmann.
- Cluster algorithms for spin models: Ising, XY, Spin glasses.

Département de Physique
École normale supérieure

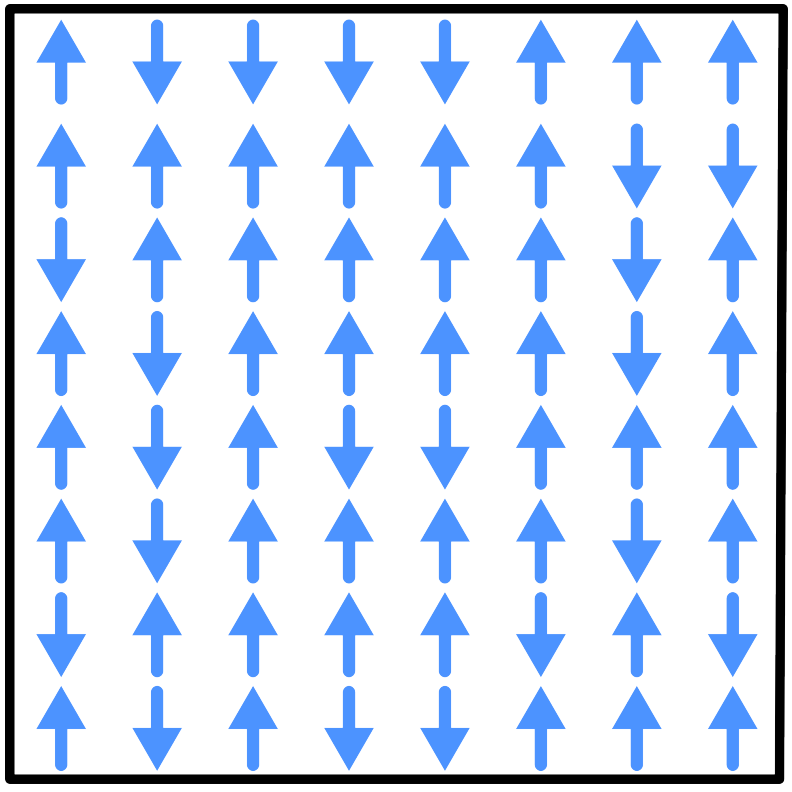
MOOC Statistical Mechanics: Algorithms and Computations



- see also: Werner Krauth *Statistical Mechanics: Algorithms and Computations* (Oxford, 2006)

Département de Physique
École normale supérieure

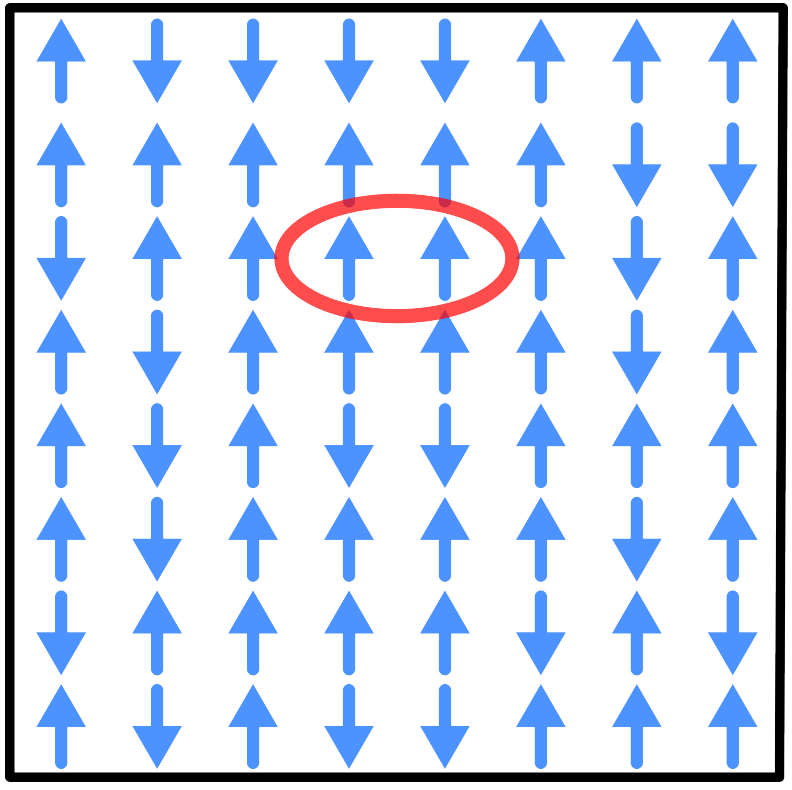
Ising model



- spins $\sigma = \pm 1$

Département de Physique
École normale supérieure

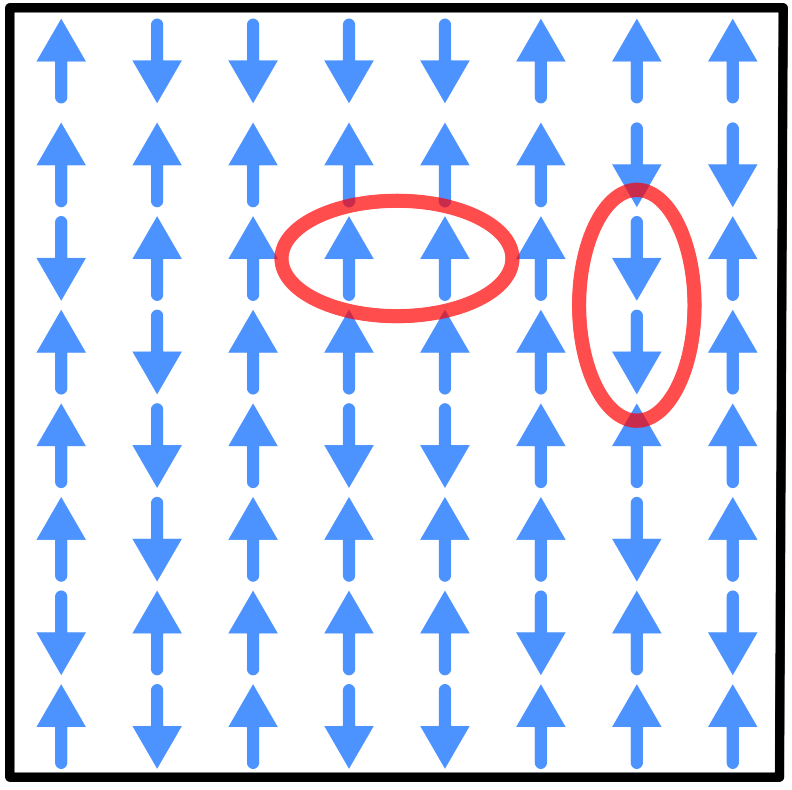
Ising model



- pair energy $E_{ij} = -1$

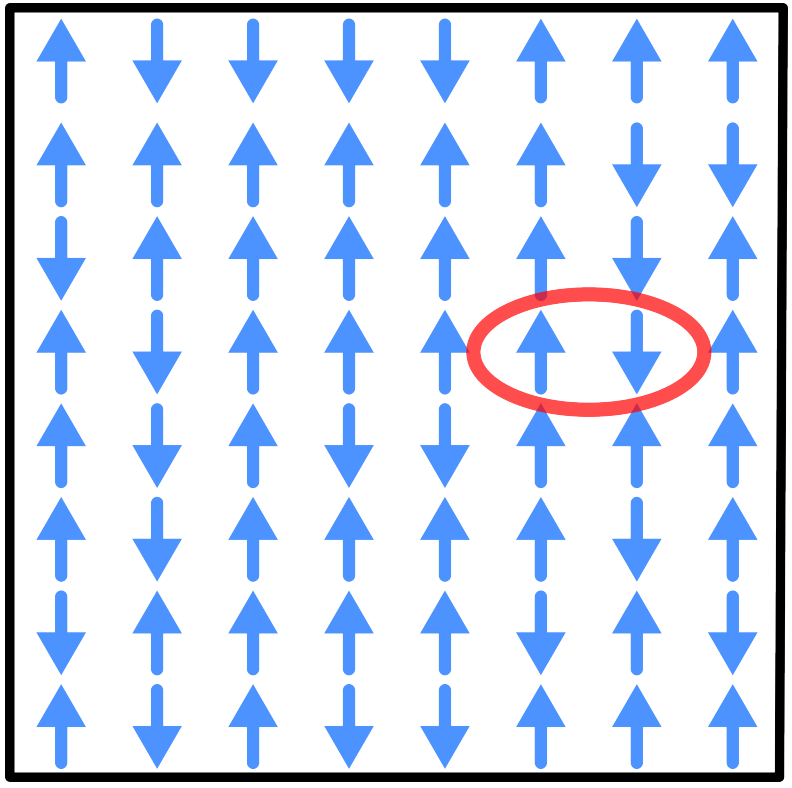
Département de Physique
École normale supérieure

Ising model



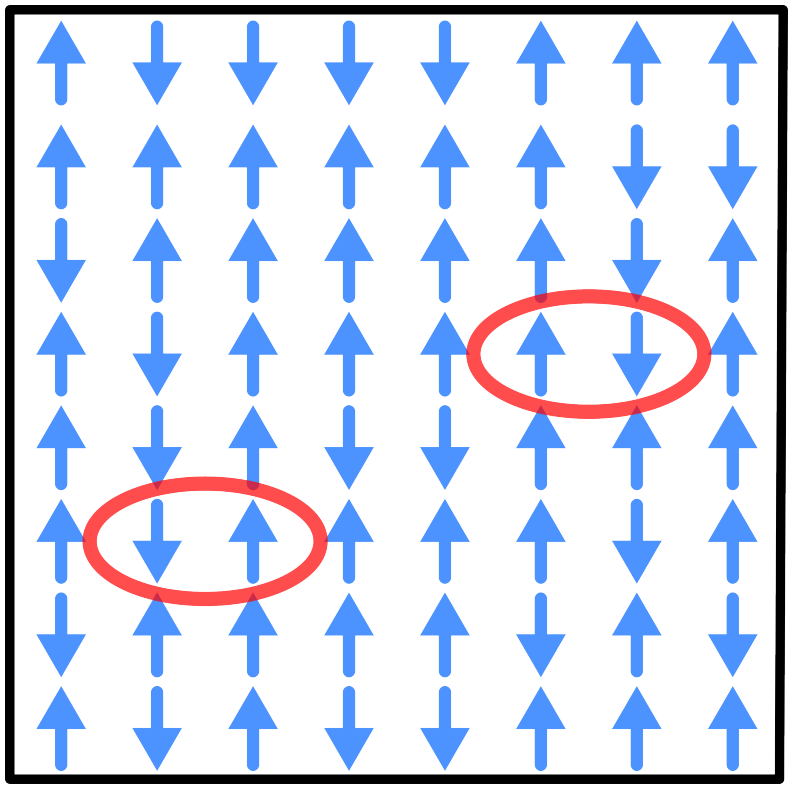
- pair energies $E_{ij} = -1$

Ising model



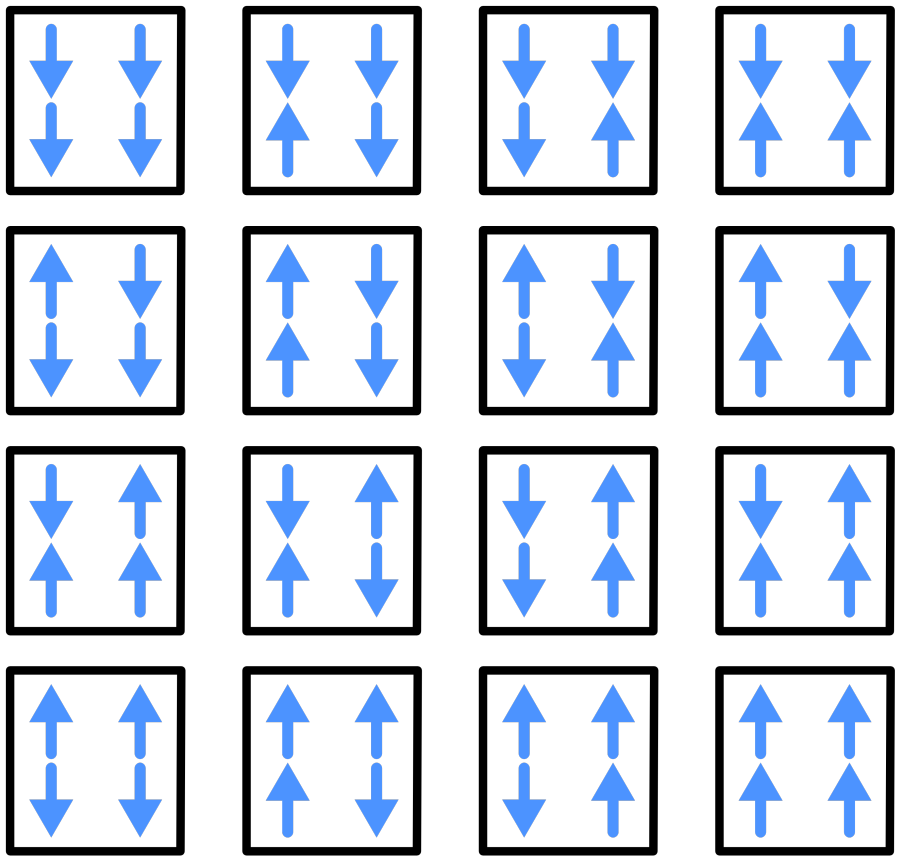
- pair energy $E_{ij} = +1$

Ising model

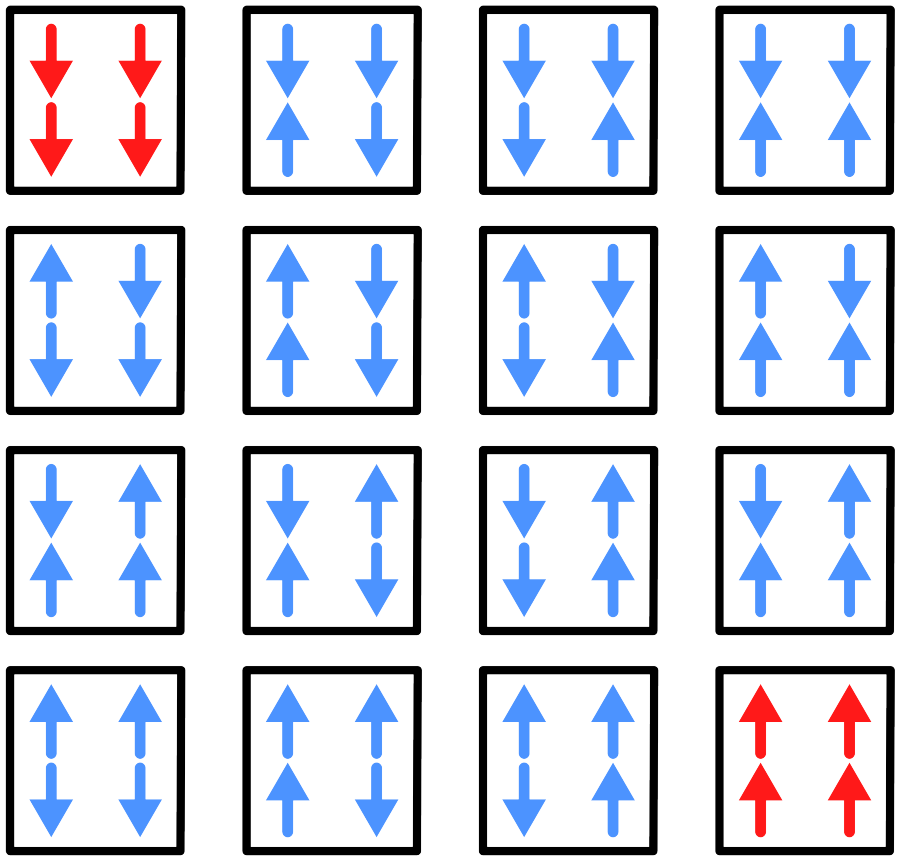


- pair energies $E_{ij} = +1$

2x2 Ising model



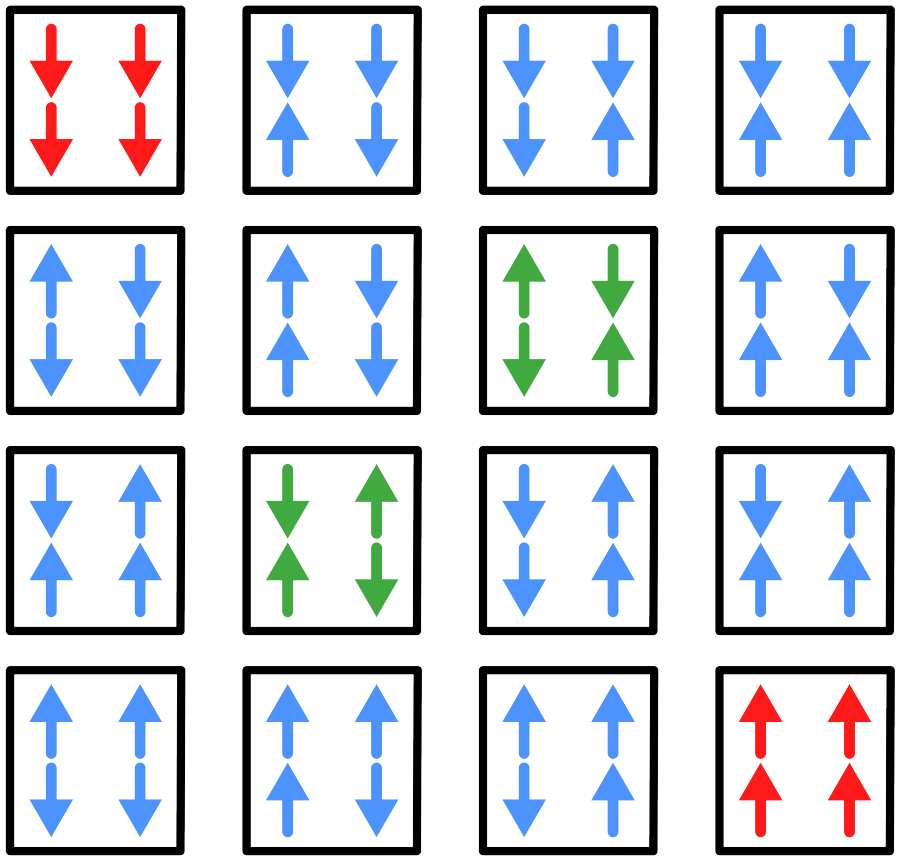
2x2 Ising model



E=-4

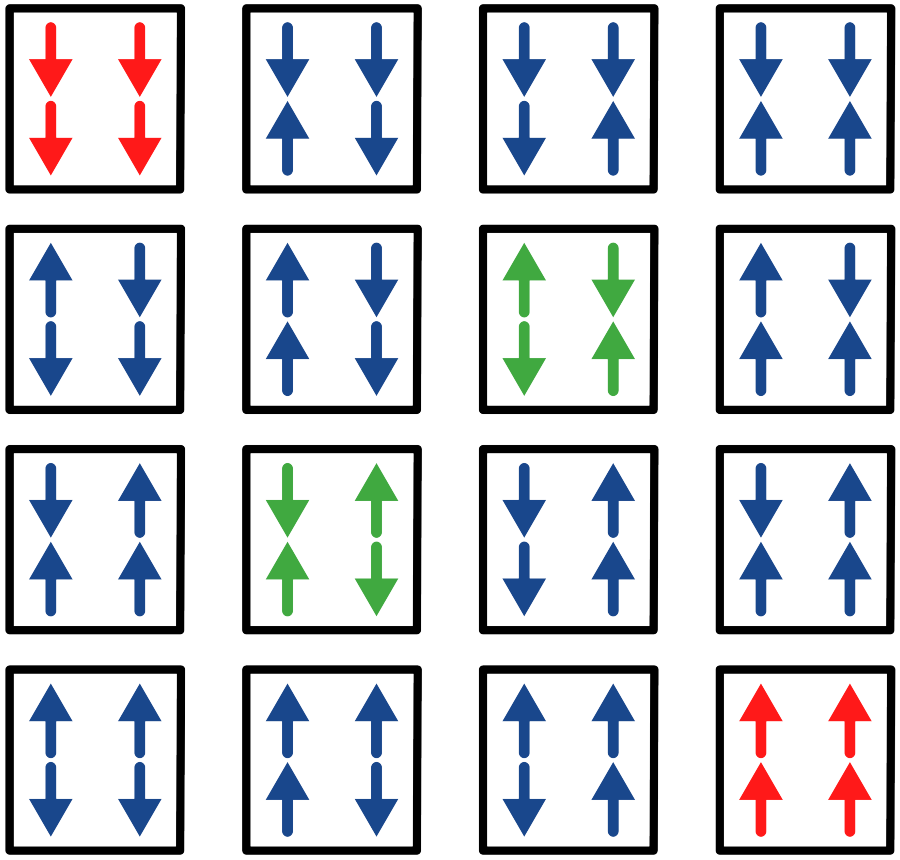
Département de Physique
École normale supérieure

2x2 Ising model



Département de Physique
École normale supérieure

2x2 Ising model



E=-4 E=4 E=0

Département de Physique
École normale supérieure

Gray-code enumeration

```
[-1, -1, -1, -1]  E = -4
```

Département de Physique
École normale supérieure

Gray-code enumeration

```
[-1, -1, -1, -1]  E = -4
[ 1, -1, -1, -1]  E =  0
```

Département de Physique
École normale supérieure

Gray-code enumeration

```
[-1, -1, -1, -1]  E = -4
[ 1, -1, -1, -1]  E =  0
[ 1,  1, -1, -1]  E =  0
[-1,  1, -1, -1]  E =  0
```

Département de Physique
École normale supérieure

Gray-code enumeration

```
[-1, -1, -1, -1]  E = -4
[ 1, -1, -1, -1]  E =  0
[ 1,  1, -1, -1]  E =  0
[-1,  1, -1, -1]  E =  0
[-1,  1,  1, -1]  E =  4
[ 1,  1,  1, -1]  E =  0
[ 1, -1,  1, -1]  E =  0
[-1, -1,  1, -1]  E =  0
```

Département de Physique
École normale supérieure

Gray-code enumeration

```
[-1, -1, -1, -1]  E = -4
[ 1, -1, -1, -1]  E =  0
[ 1,  1, -1, -1]  E =  0
[-1,  1, -1, -1]  E =  0
[-1,  1,  1, -1]  E =  4
[ 1,  1,  1, -1]  E =  0
[ 1, -1,  1, -1]  E =  0
[-1, -1,  1, -1]  E =  0
[-1, -1,  1,  1]  E =  0
[ 1, -1,  1,  1]  E =  0
[ 1,  1,  1,  1]  E = -4
[-1,  1,  1,  1]  E =  0
```

Département de Physique
École normale supérieure

Gray-code enumeration

```
[-1, -1, -1, -1]  E = -4
[ 1, -1, -1, -1]  E =  0
[ 1,  1, -1, -1]  E =  0
[-1,  1, -1, -1]  E =  0
[-1,  1,  1, -1]  E =  4
[ 1,  1,  1, -1]  E =  0
[ 1, -1,  1, -1]  E =  0
[-1, -1,  1, -1]  E =  0
[-1, -1,  1,  1]  E =  0
[ 1, -1,  1,  1]  E =  0
[ 1,  1,  1,  1]  E = -4
[-1,  1,  1,  1]  E =  0
[-1,  1, -1,  1]  E =  0
[ 1,  1, -1,  1]  E =  0
[ 1, -1, -1,  1]  E =  4
[-1, -1, -1,  1]  E =  0
```

Département de Physique
École normale supérieure

enumerate-ising.py

```
def gray_flip(t, N):
    k = t[0]
    if k > N: return t, k
    t[k - 1] = t[k]
    t[k] = k + 1
    if k != 1: t[0] = 1
    return t, k

L = 4
N = L * L
nbr = {i : ((i // L) * L + (i + 1) % L, (i + L) % N,
            (i // L) * L + (i - 1) % L, (i - L) % N)
                                        for i in range(N)}
S = [-1] * N
E = -2 * N
print S, E
tau = range(1, N + 2)
for i in range(1, 2 ** N):
    tau, k = gray_flip(tau, N)
    h = sum(S[n] for n in nbr[k - 1])
    E += 2 * h * S[k - 1]
    S[k - 1] *= -1
    print S, E
```

Département de Physique
École normale supérieure

Partition function

$$Z(\beta) = - \sum_{\substack{\sigma_0 = \pm 1 \\ \dots \\ \sigma_{N-1} = \pm 1}} e^{-\beta E(\sigma_0, \dots, \sigma_{N-1})}$$

Département de Physique
École normale supérieure

Partition function

$$Z(\beta) = - \sum_{\substack{\sigma_0 = \pm 1 \\ \dots \\ \sigma_{N-1} = \pm 1}} e^{-\beta E(\sigma_0, \dots, \sigma_{N-1})}$$

$$= \sum_E \mathcal{N}(E) e^{-\beta E}$$

Département de Physique
École normale supérieure

Computation of observables

$$\langle E \rangle = \frac{\sum_{\boldsymbol{\sigma}} E_{\boldsymbol{\sigma}} \, e^{-\beta E_{\boldsymbol{\sigma}}}}{\sum_{\boldsymbol{\sigma}} e^{-\beta E_{\boldsymbol{\sigma}}}}$$

Département de Physique
École normale supérieure

Computation of observables

$$\langle E \rangle = \frac{\sum_{\boldsymbol{\sigma}} E_{\boldsymbol{\sigma}} \, e^{-\beta E_{\boldsymbol{\sigma}}}}{\sum_{\boldsymbol{\sigma}} e^{-\beta E_{\boldsymbol{\sigma}}}}$$

$$= \sum_{E} E \, \mathcal{N}(E) \, e^{-\beta E}$$

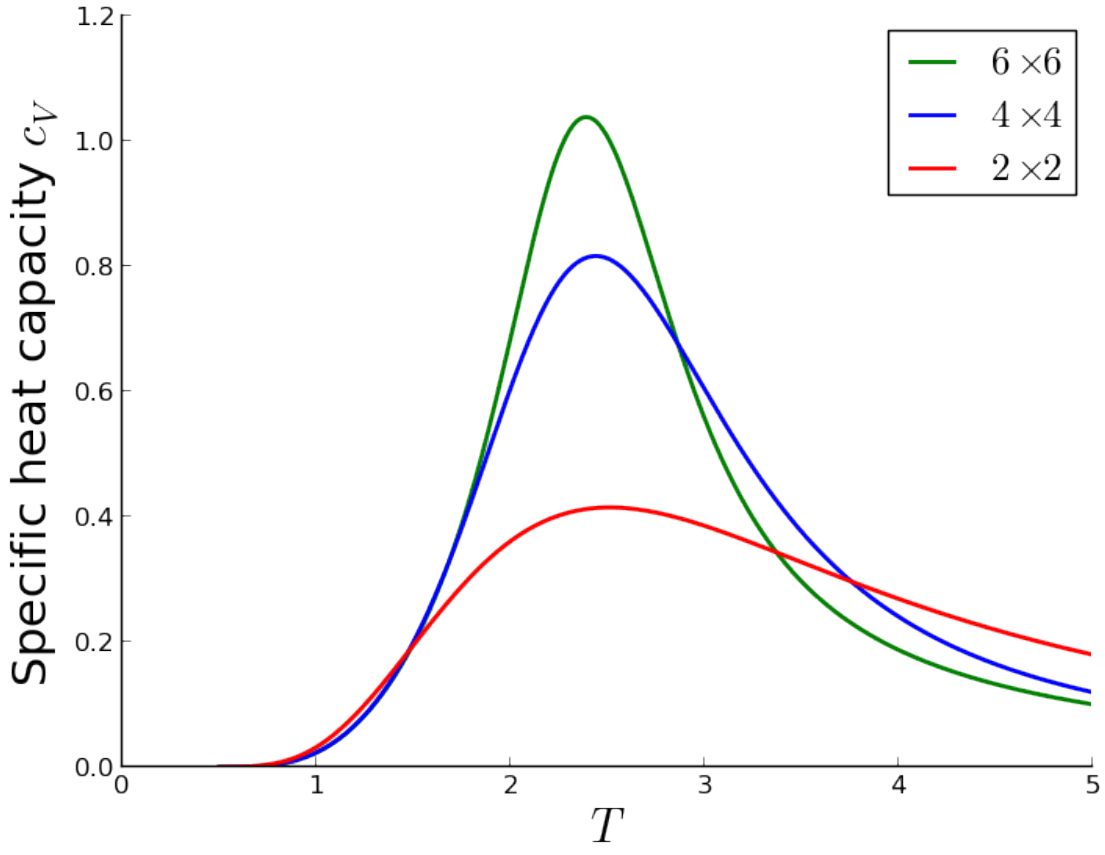
Département de Physique
École normale supérieure

thermo-ising.py

```
L = 4
N = L * L
nbr = {i : ((i // L) * L + (i + 1) % L, (i + L) % N,
            (i // L) * L + (i - 1) % L, (i - L) % N) \
                                        for i in range(N)}
dos = {}
S = [-1] * N
E = energy(S, N, nbr)
dos[E] = 1
tau = range(1, N + 2)
for i in range(1, 2 ** N):
    tau, k = gray_flip(tau, N)
    E += 2 * sum(S[k - 1] * S[n] for n in nbr[k - 1])
    S[k - 1] *= -1
    if E in dos: dos[E] += 1
    else:        dos[E] = 1
for T in [0.1 * i for i in range(1, 51)]:
    Z = 0.0
    E_av = 0.0
    M_av = 0.0
    E2_av = 0.0
    for E in dos.keys():
        weight = math.exp(- E / T) * dos[E]
        Z += weight
        E_av += weight * E
        E2_av += weight * E ** 2
    E2_av /= Z
    E_av /= Z
    cv = (E2_av - E_av ** 2) / N / T
```

Département de Physique
École normale supérieure

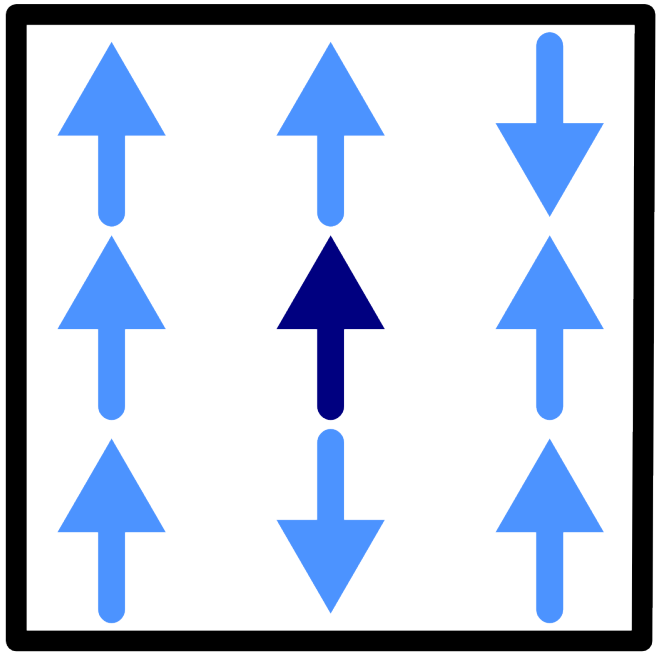
Specific heat of Ising model on finite lattices



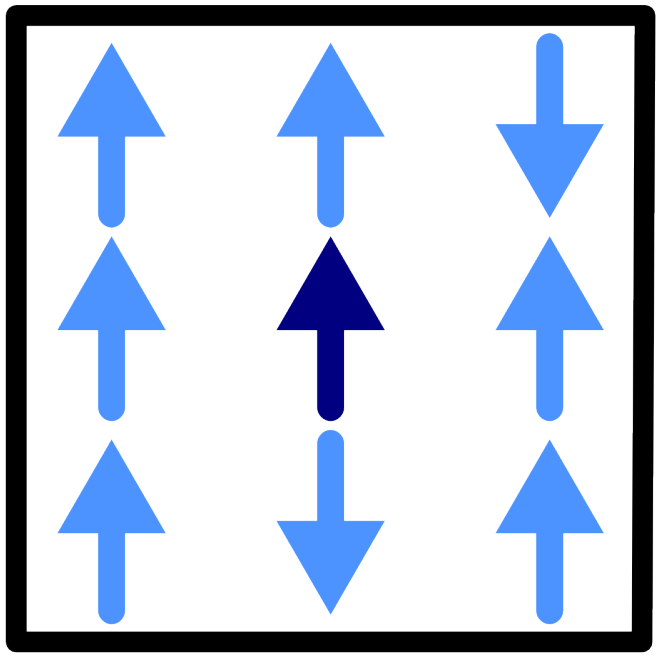
$$T_c = \frac{2}{\log(1+\sqrt{2})} = 2.269...$$

Département de Physique
École normale supérieure

Markov-chain sampling

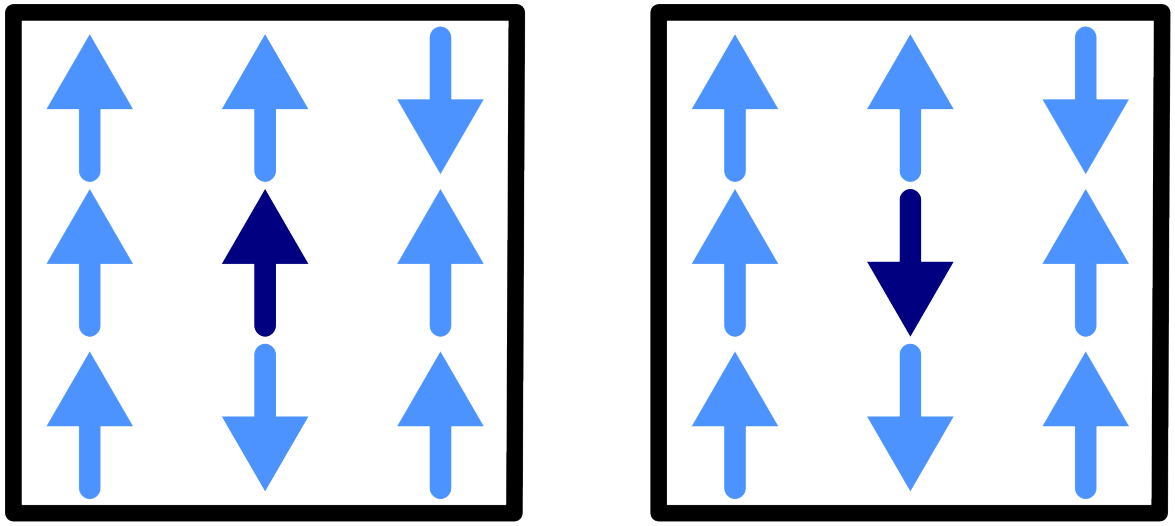


Markov-chain sampling



a

Markov-chain sampling



$$p^{\mathrm{acc}}(a \to b) = \min\left(1, e^{-\beta(E_b - E_a)}\right)$$

Département de Physique
École normale supérieure

markov-ising.py

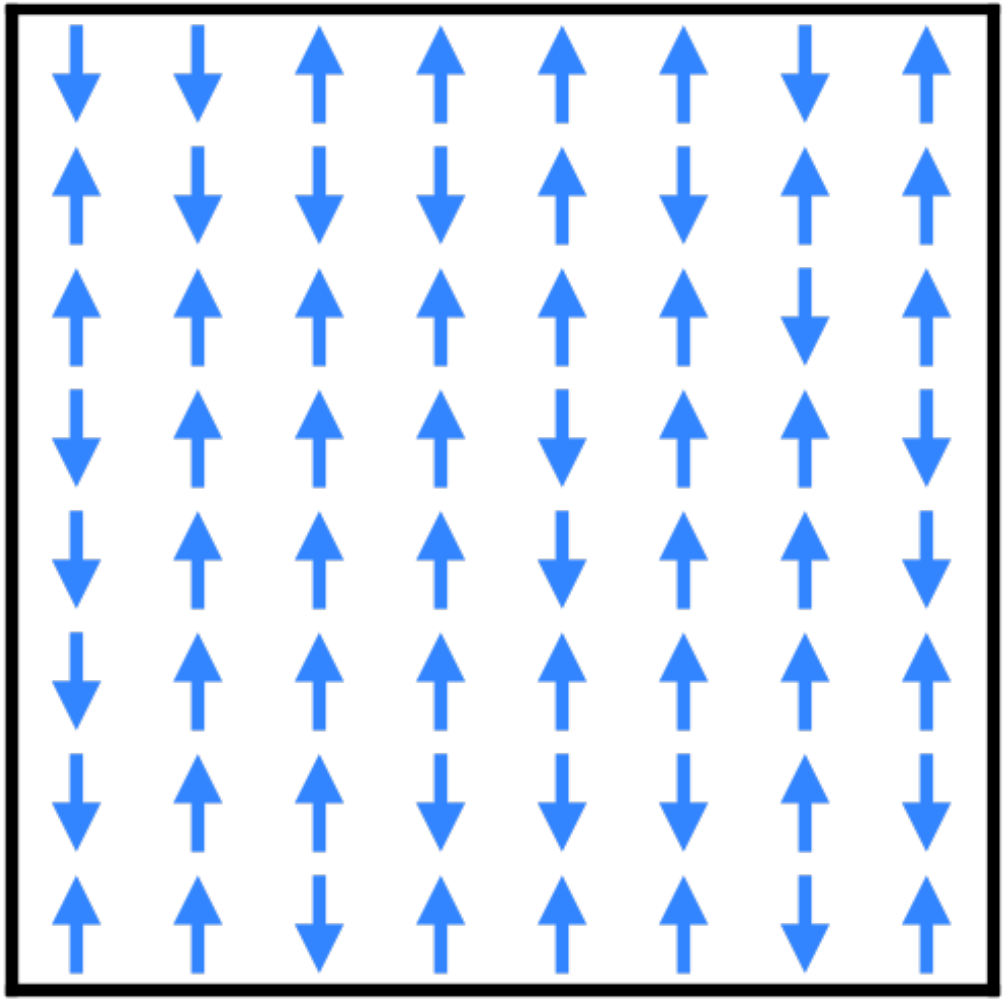
```
import random, math

L = 16
N = L * L
nbr = {i : ((i // L) * L + (i + 1) % L, (i + L) % N,
            (i // L) * L + (i - 1) % L, (i - L) % N) \
                                        for i in range(N)}
nsteps = 1000000
T = 2.0
beta = 1.0 / T
S = [random.choice([1, -1]) for k in range(N)]
for step in range(nsteps):
    k = random.randint(0, N - 1)
    delta_E = 2.0 * S[k] * sum(S[nn] for nn in nbr[k])
    if random.uniform(0.0, 1.0) < math.exp(-beta * delta_E):
        S[k] *= -1
print S, sum(S)
```

Département de Physique
École normale supérieure

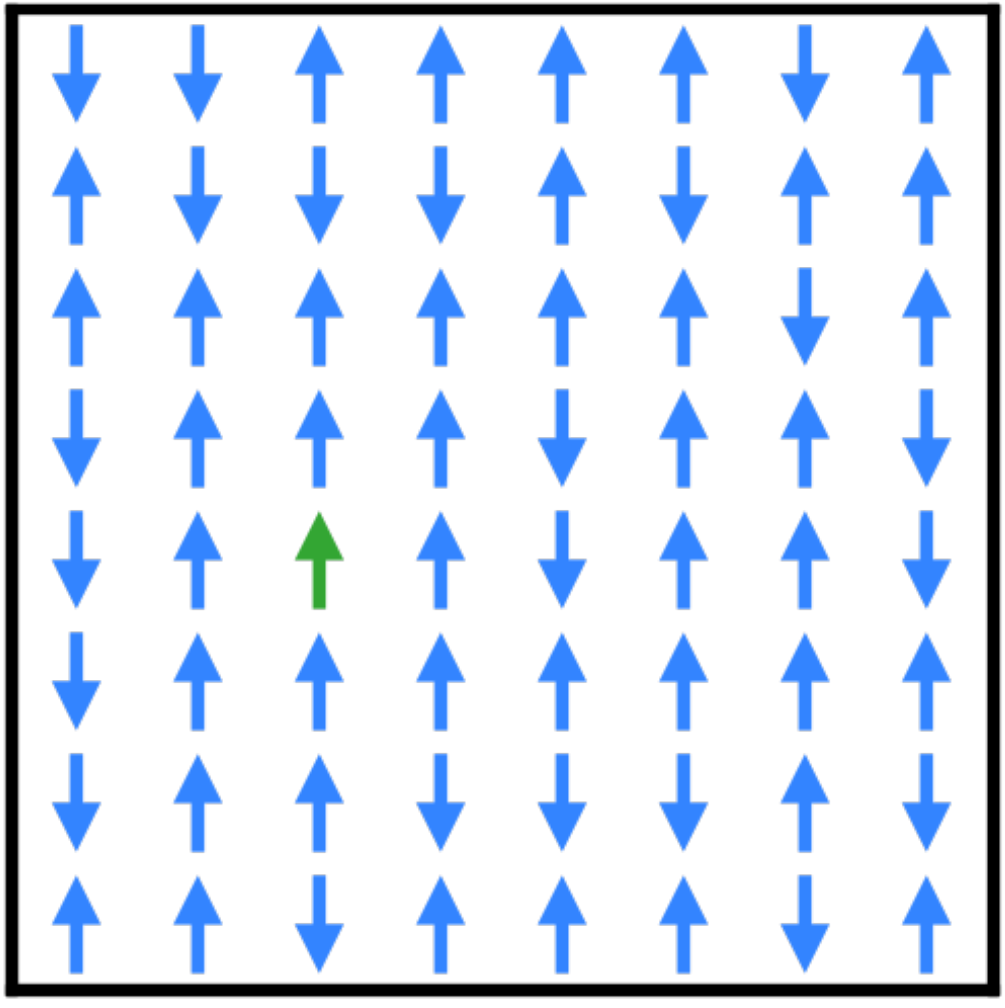
Configurations around Tc

Département de Physique
École normale supérieure

Cluster algorithm, first idea

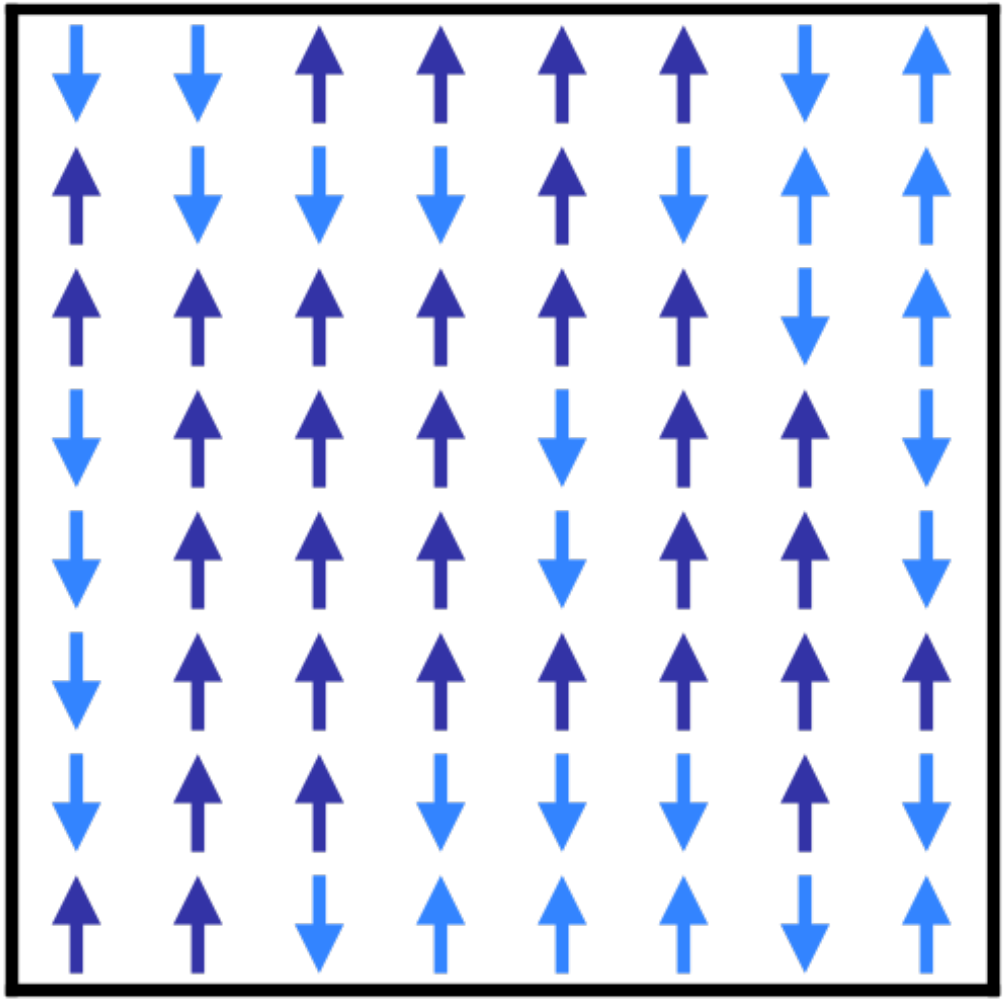


Cluster algorithm, first idea



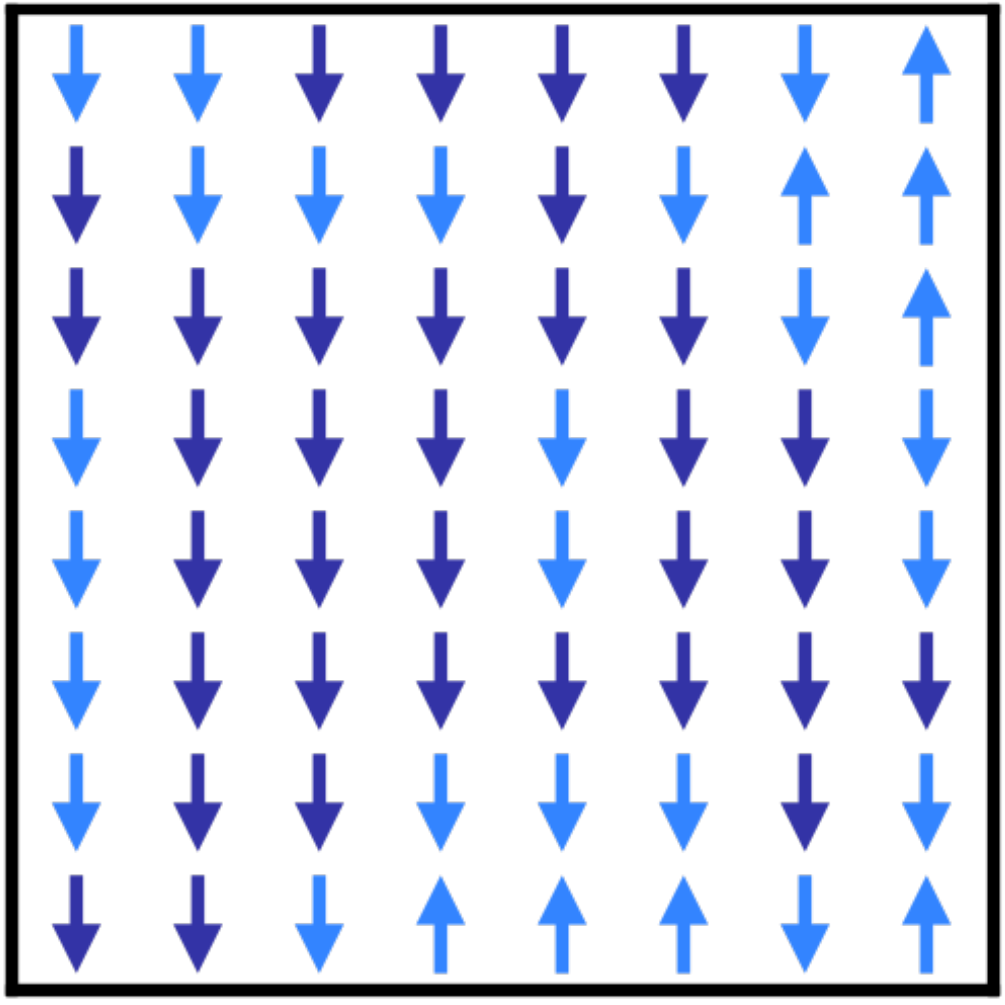
Département de Physique
École normale supérieure

Cluster algorithm, first idea



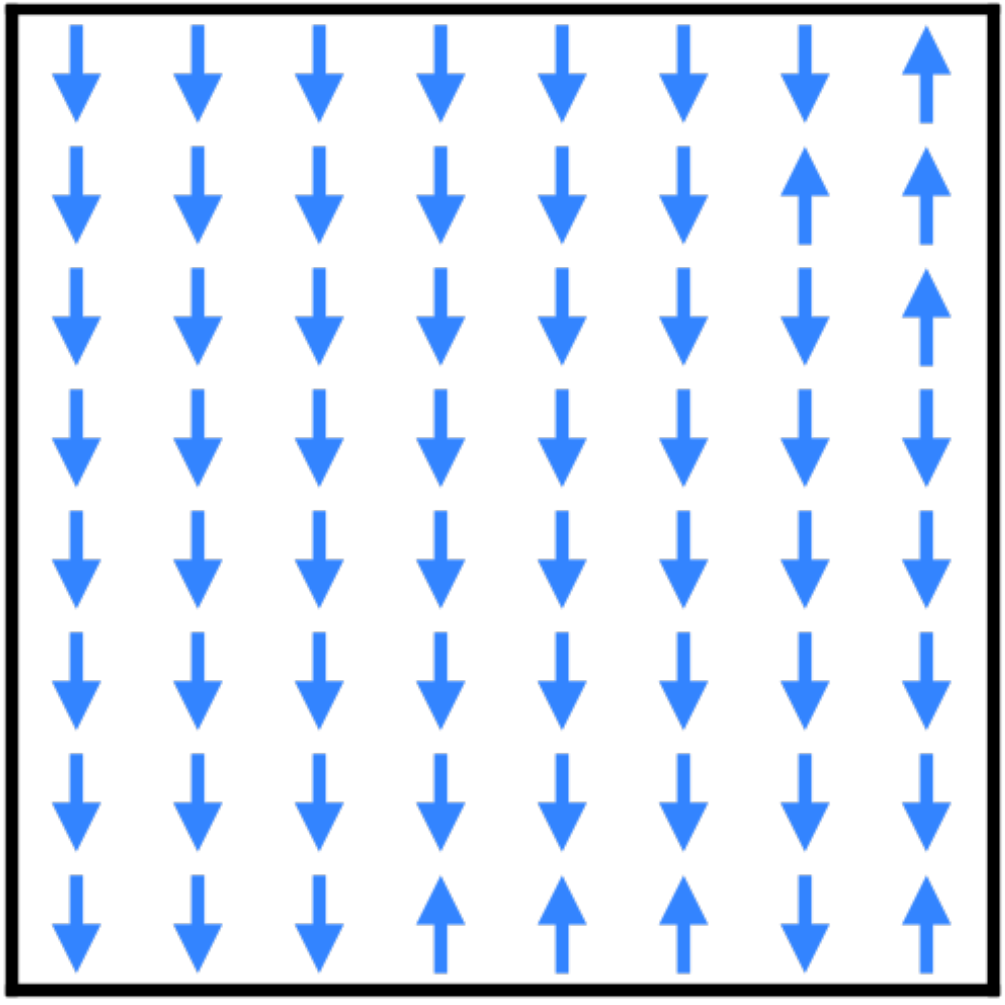
Département de Physique
École normale supérieure

Cluster algorithm, first idea



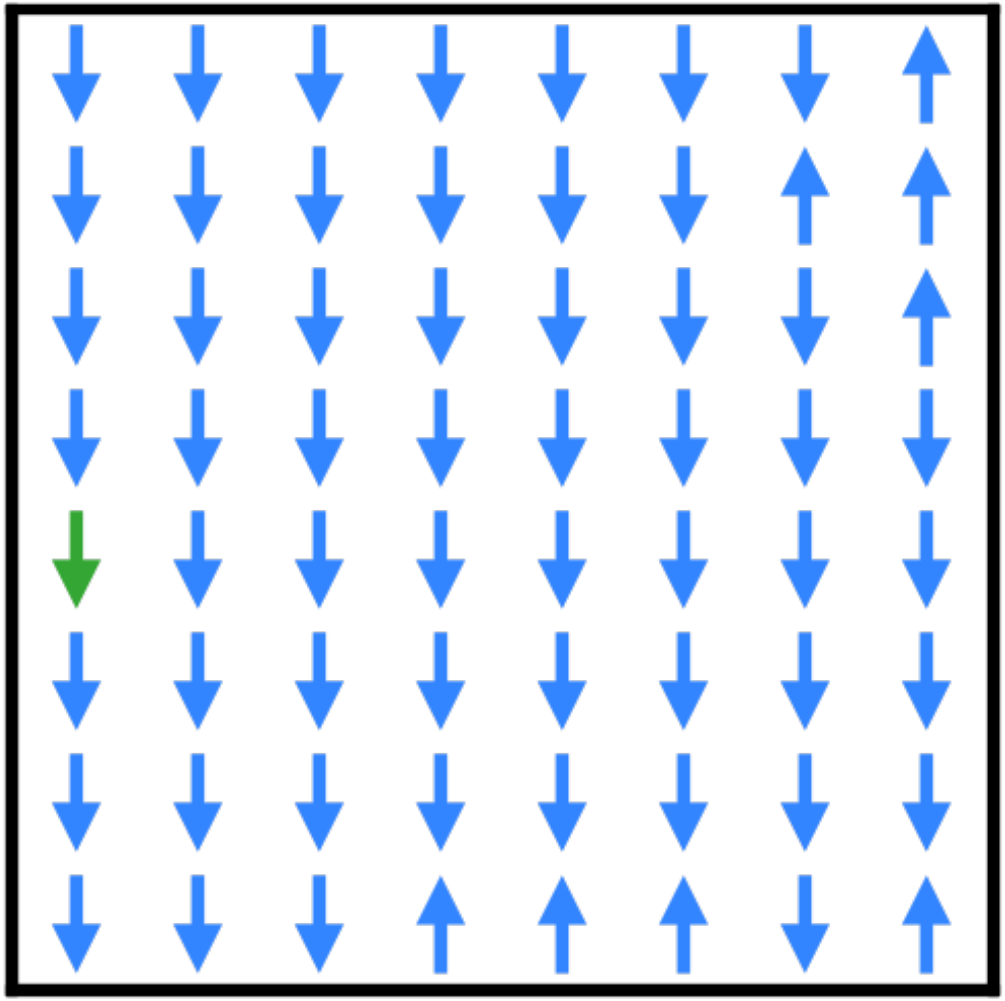
Département de Physique
École normale supérieure

Cluster algorithm, first idea



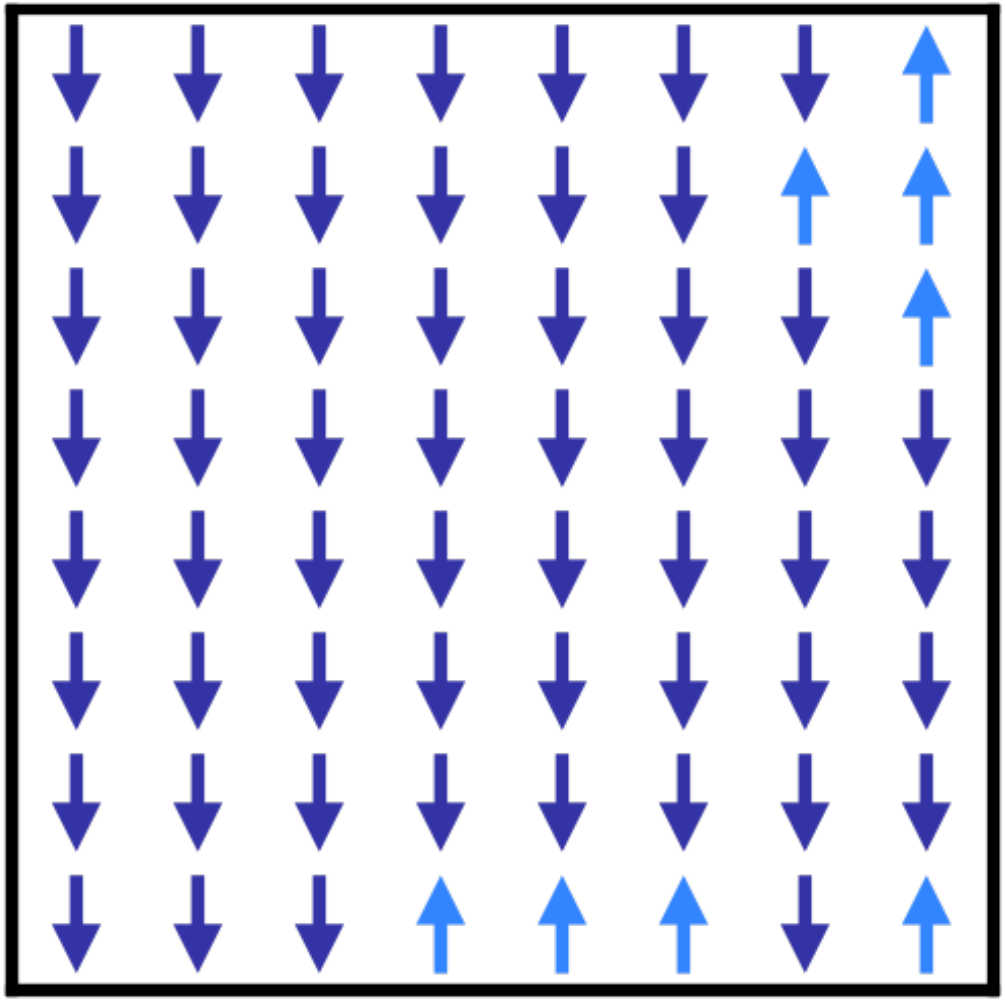
Département de Physique
École normale supérieure

Cluster algorithm, first idea



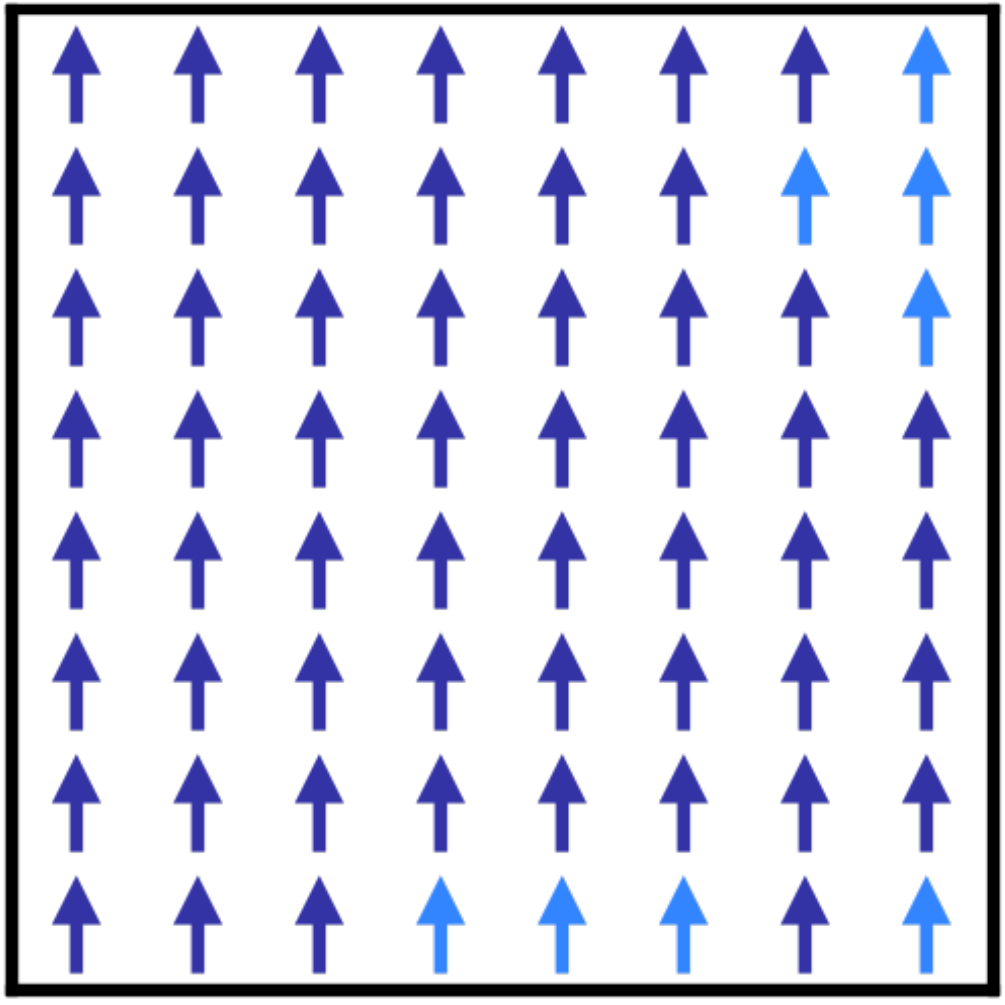
Département de Physique
École normale supérieure

Cluster algorithm, first idea



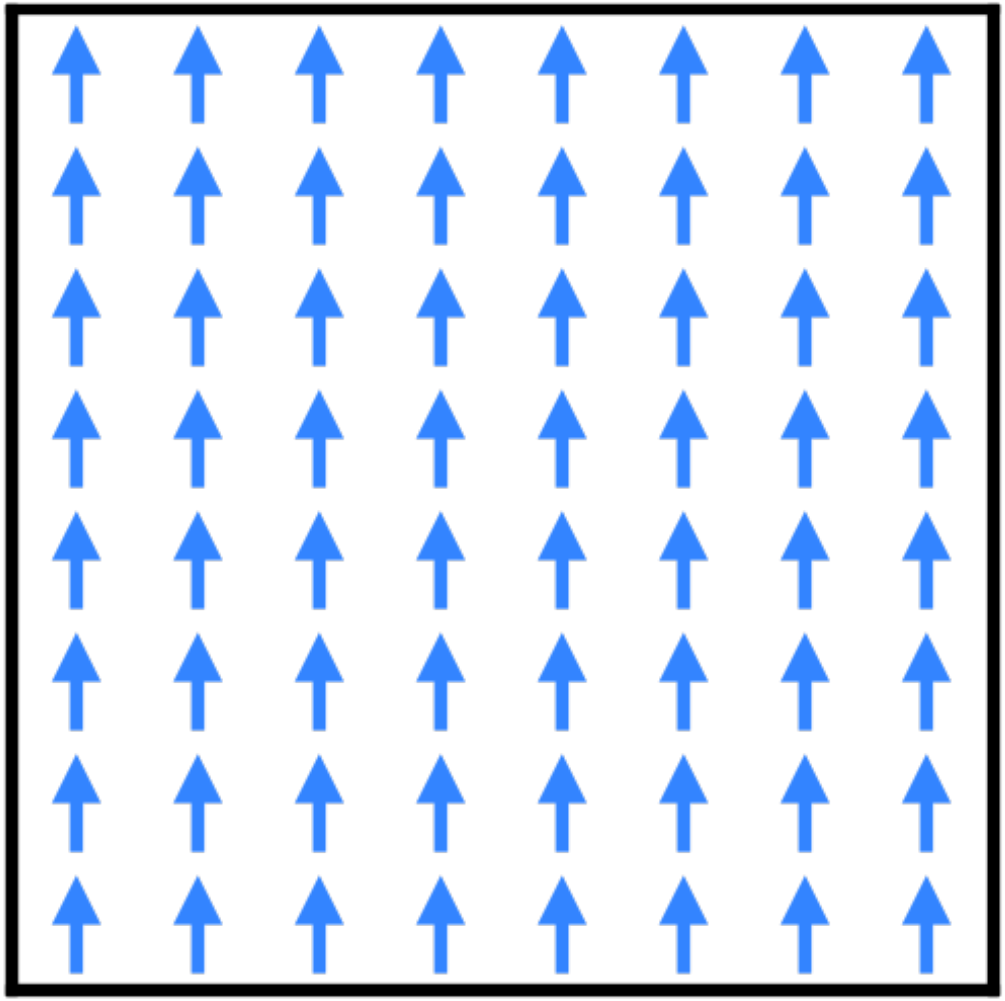
Département de Physique
École normale supérieure

Cluster algorithm, first idea



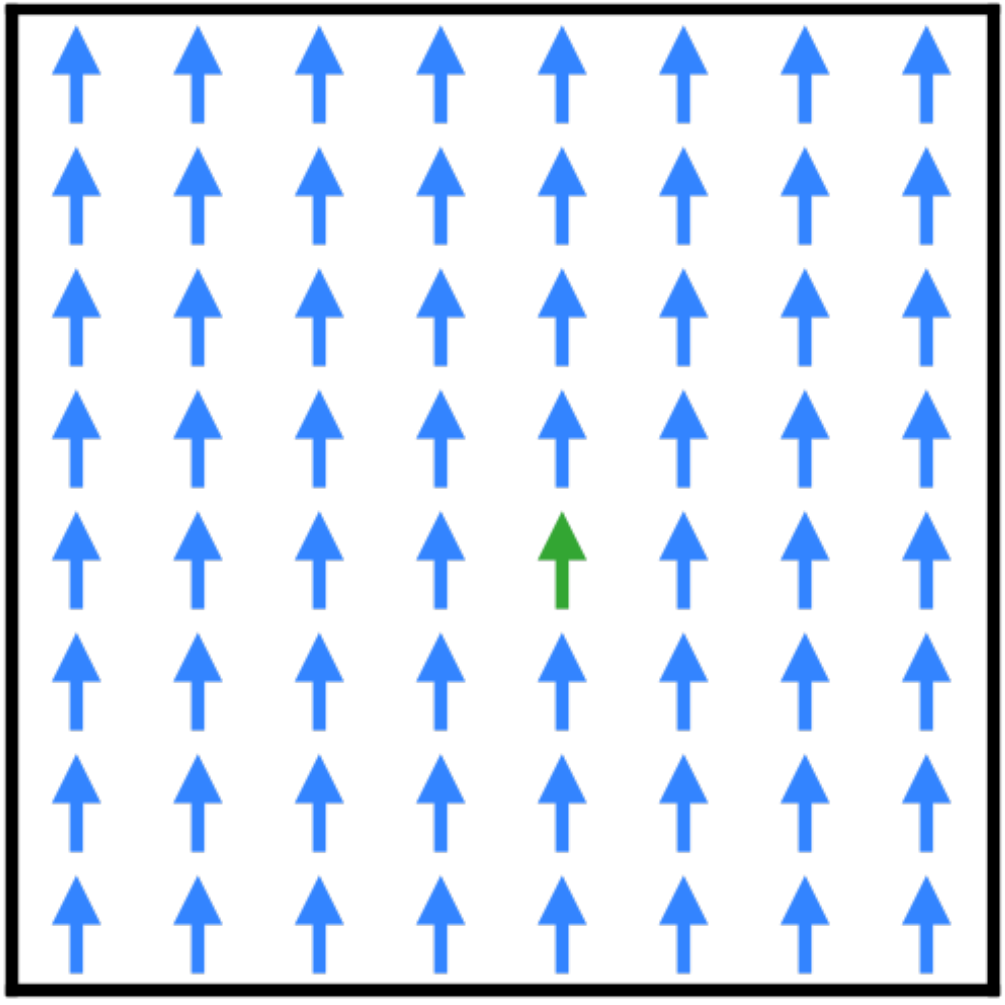
Département de Physique
École normale supérieure

Cluster algorithm, first idea



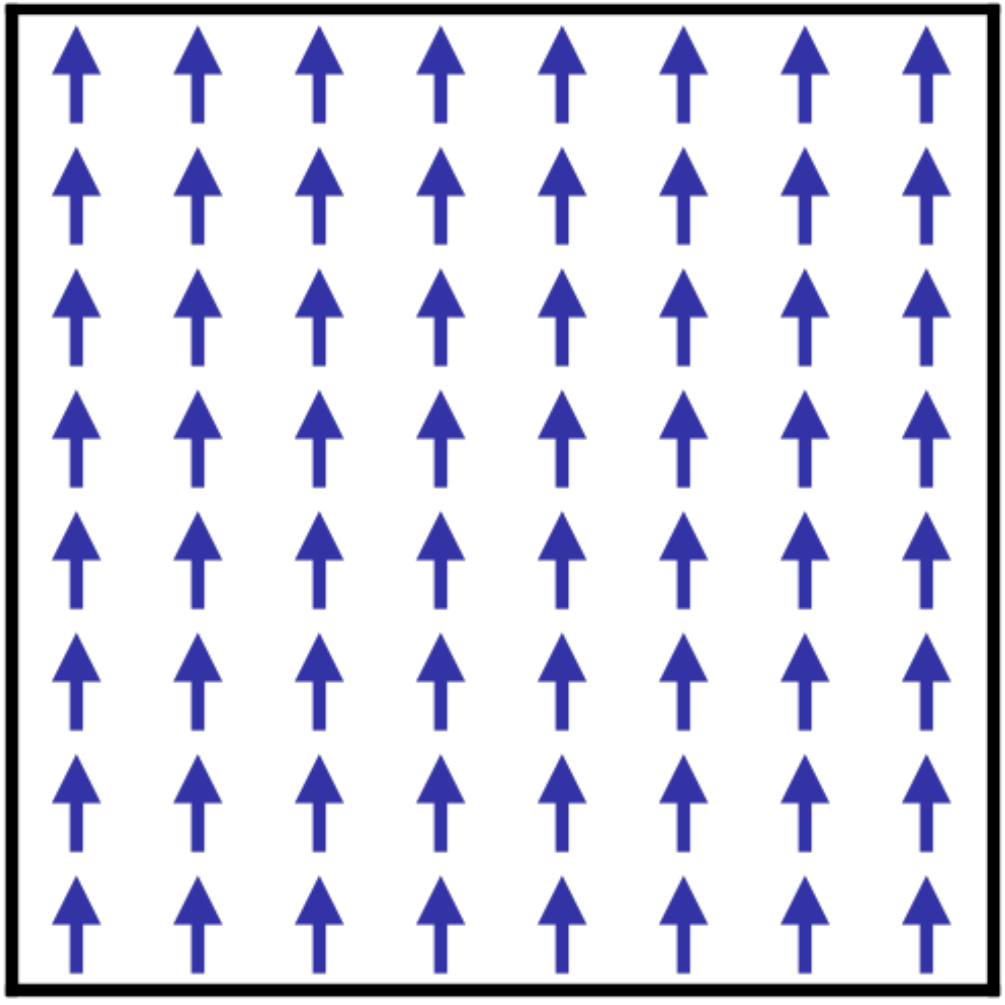
Département de Physique
École normale supérieure

Cluster algorithm, first idea



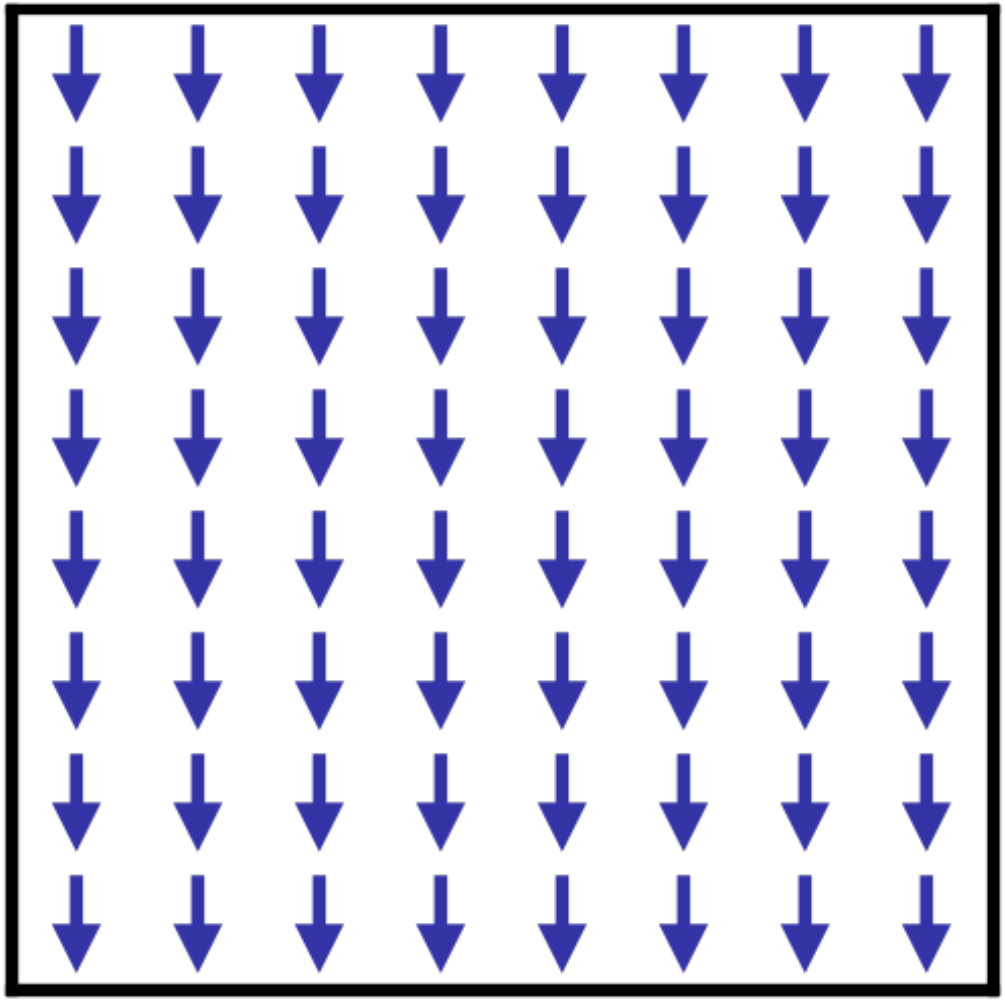
Département de Physique
École normale supérieure

Cluster algorithm, first idea



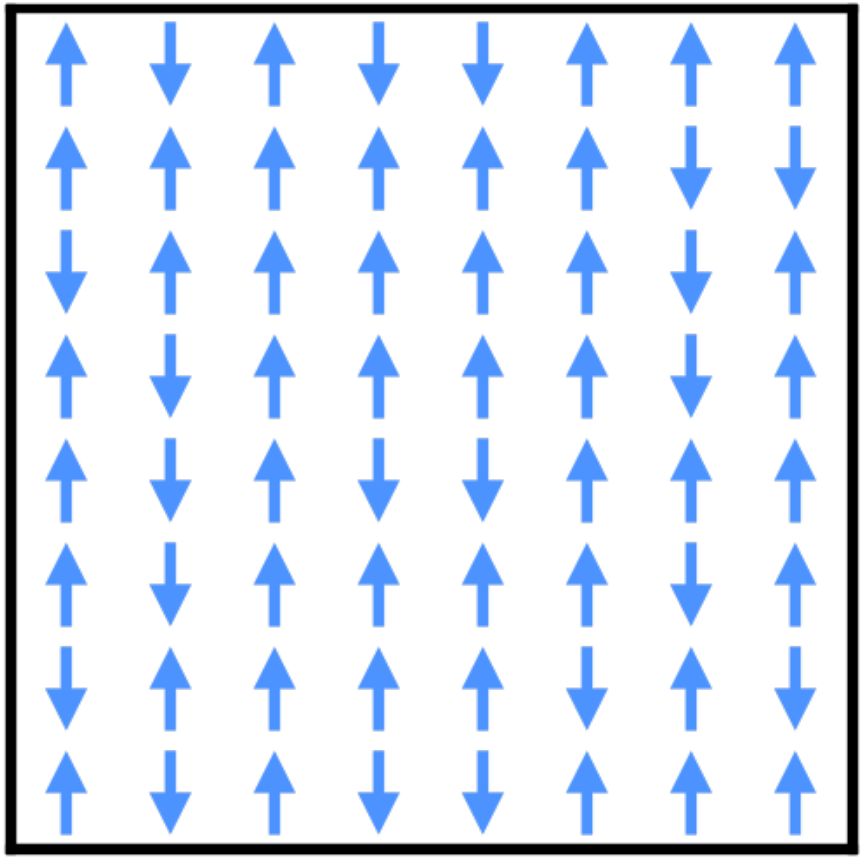
Département de Physique
École normale supérieure

Cluster algorithm, first idea



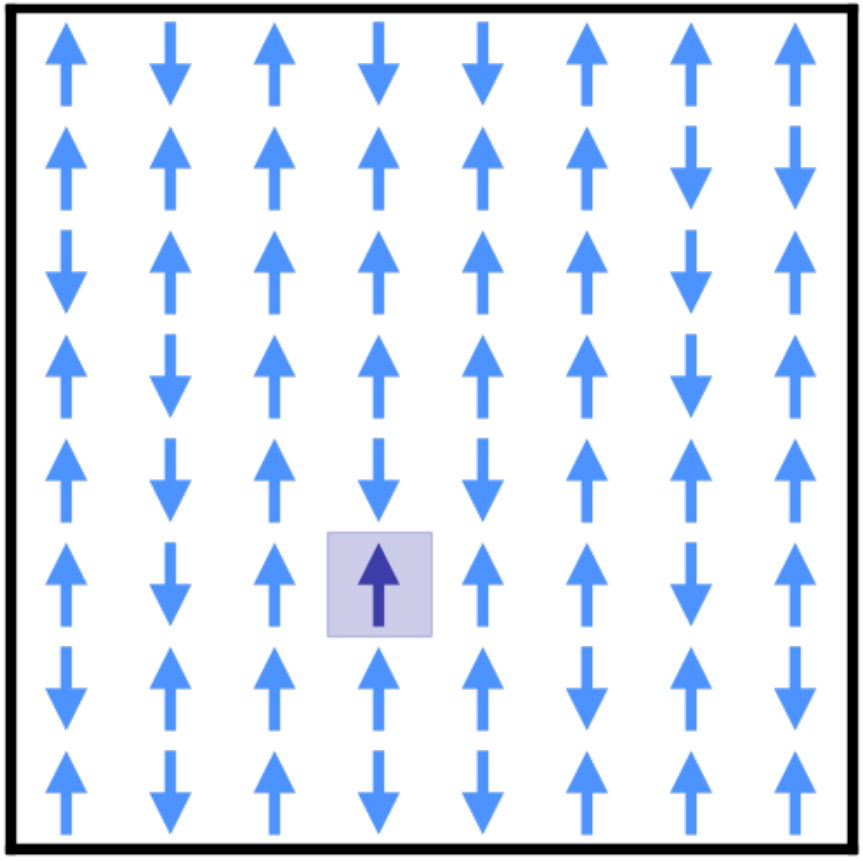
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



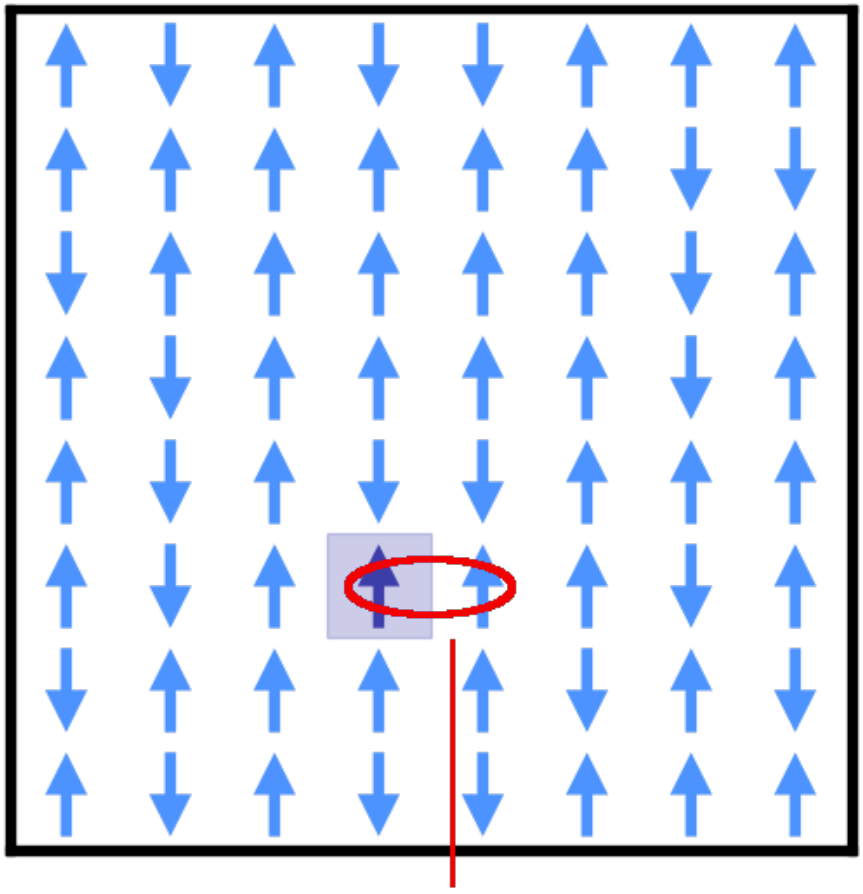
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



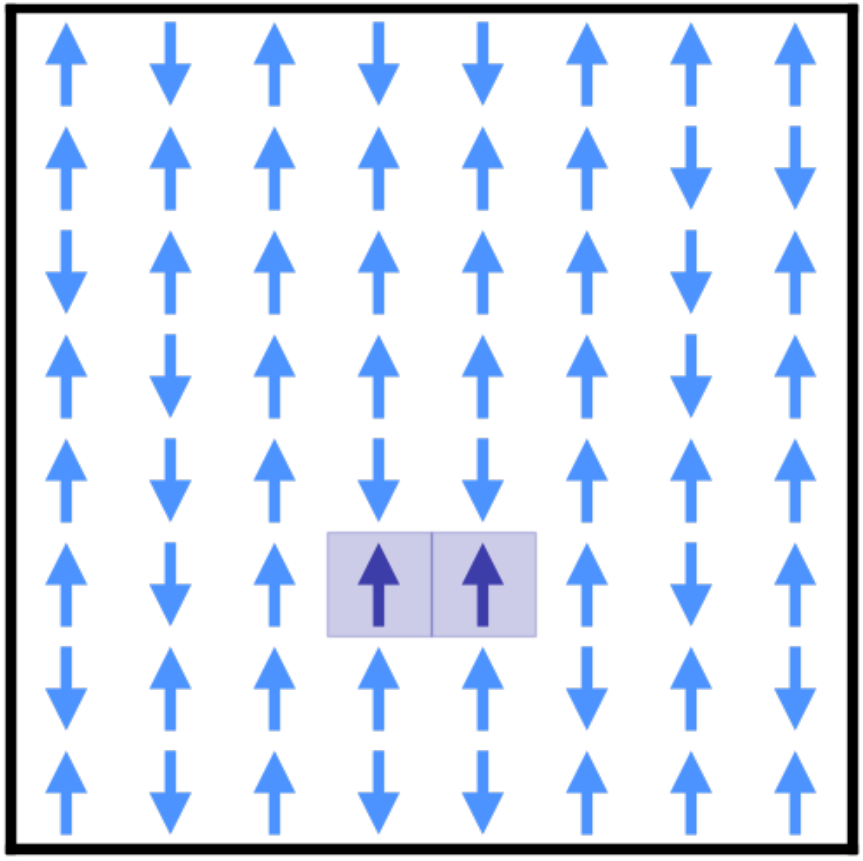
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



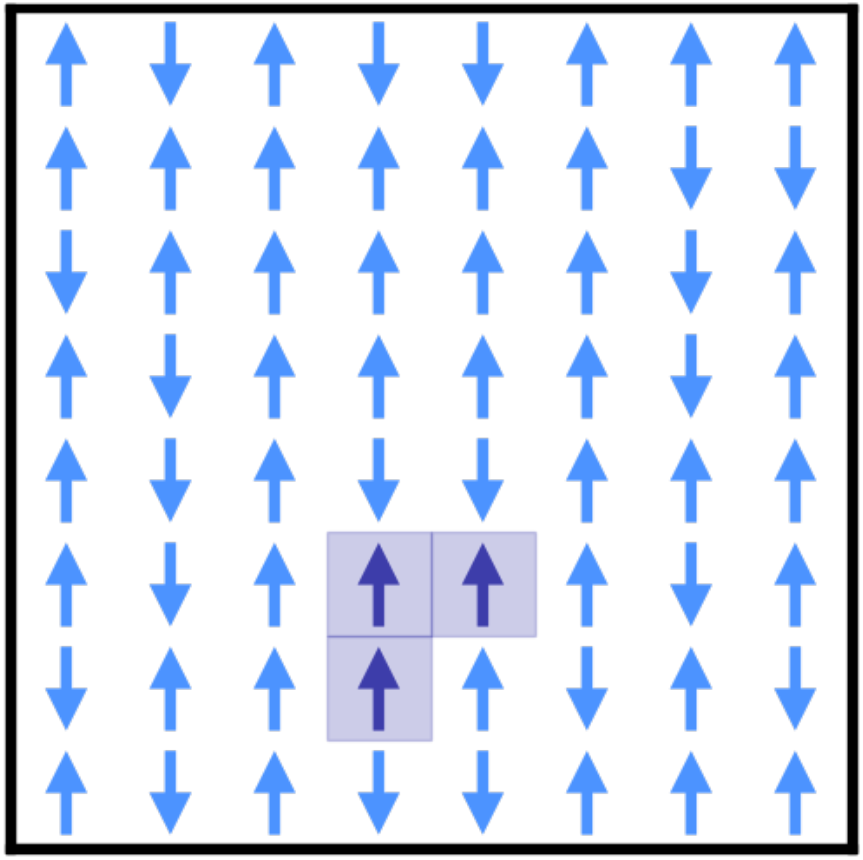
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



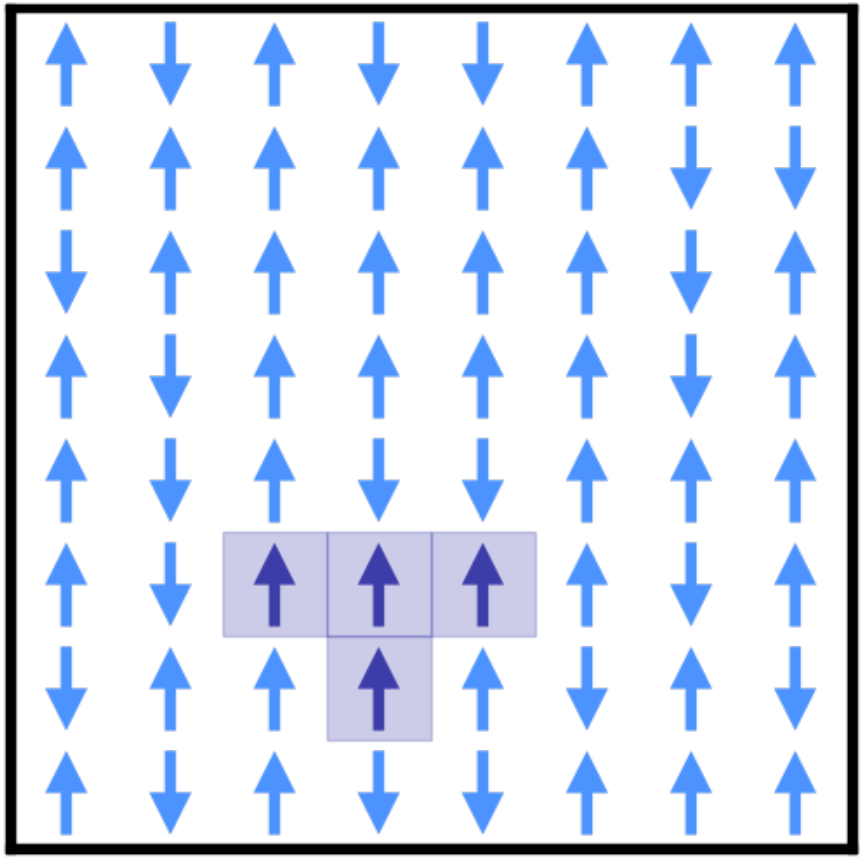
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



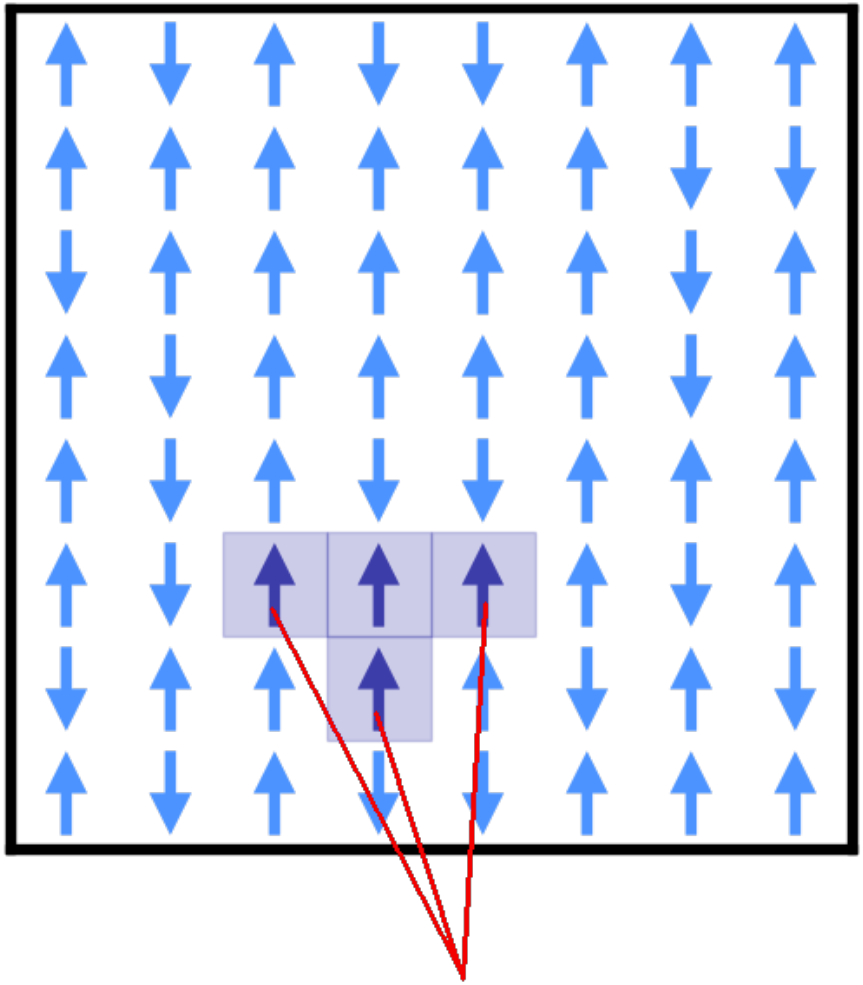
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)

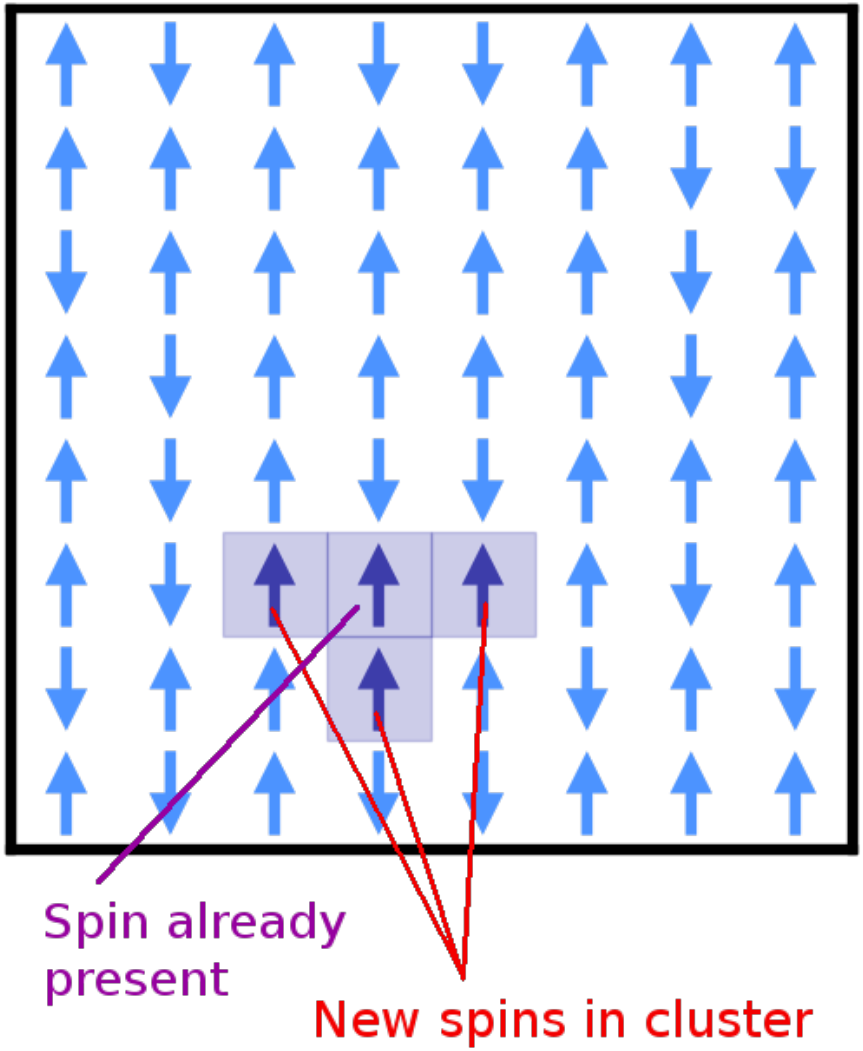


Département de Physique
École normale supérieure

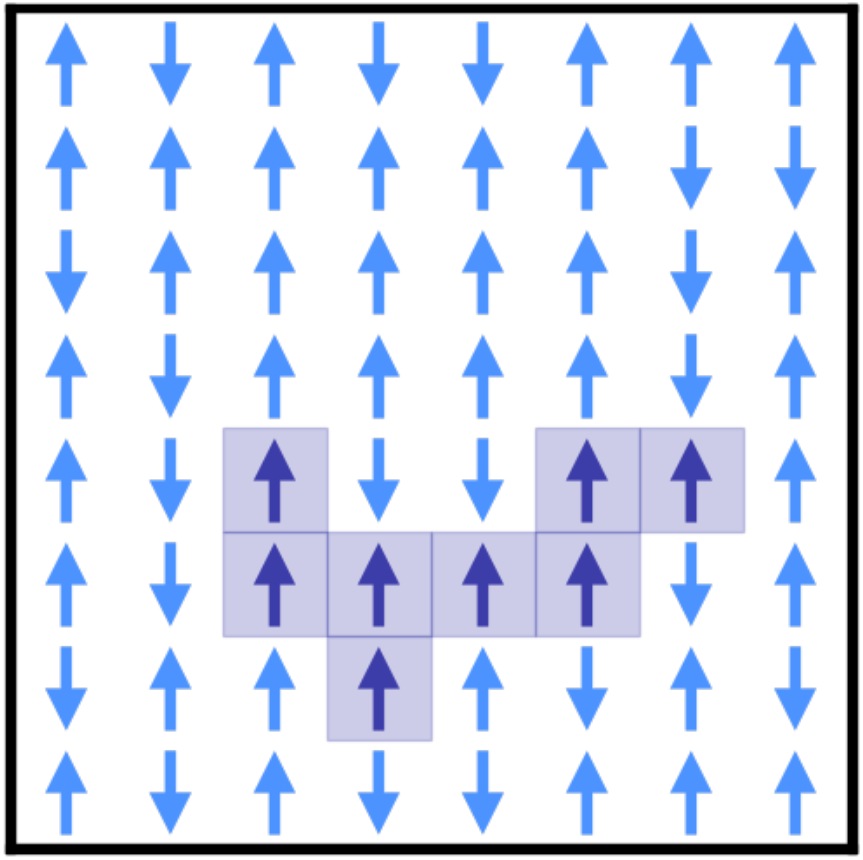
Cluster algorithm, probabilistic (Wolff, 1989)



Cluster algorithm, probabilistic (Wolff, 1989)

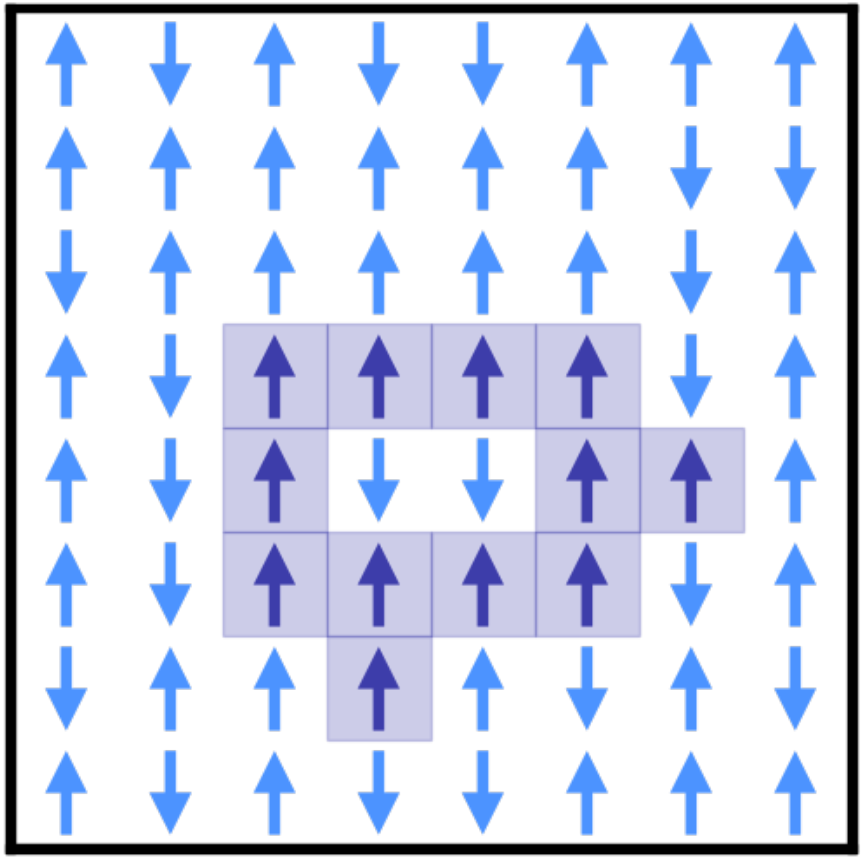


Cluster algorithm, probabilistic (Wolff, 1989)



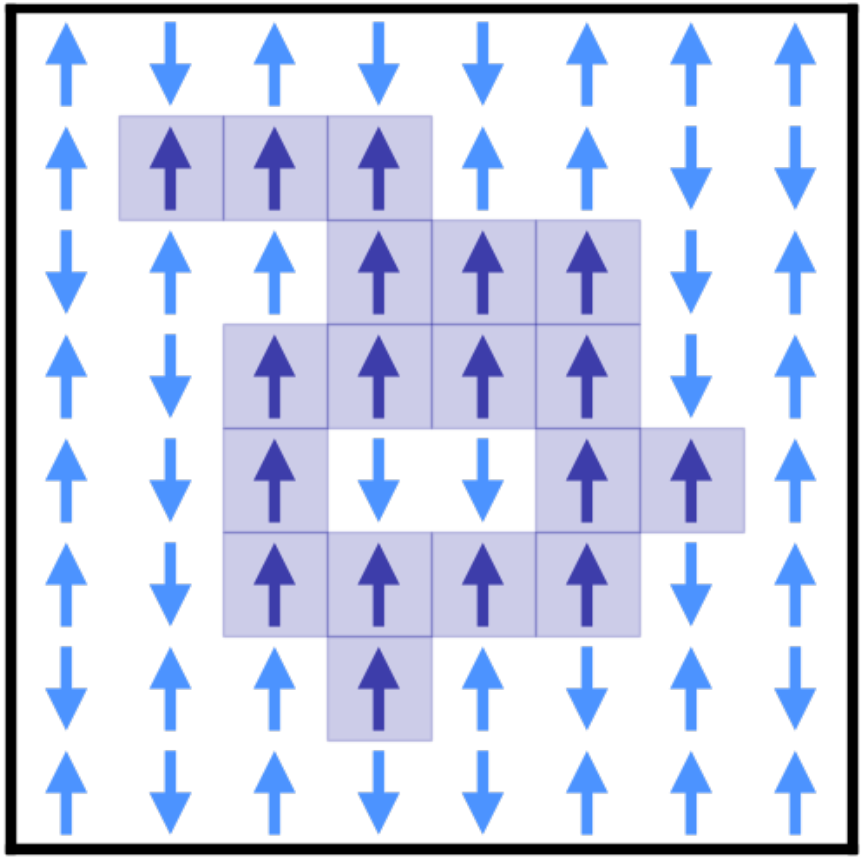
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



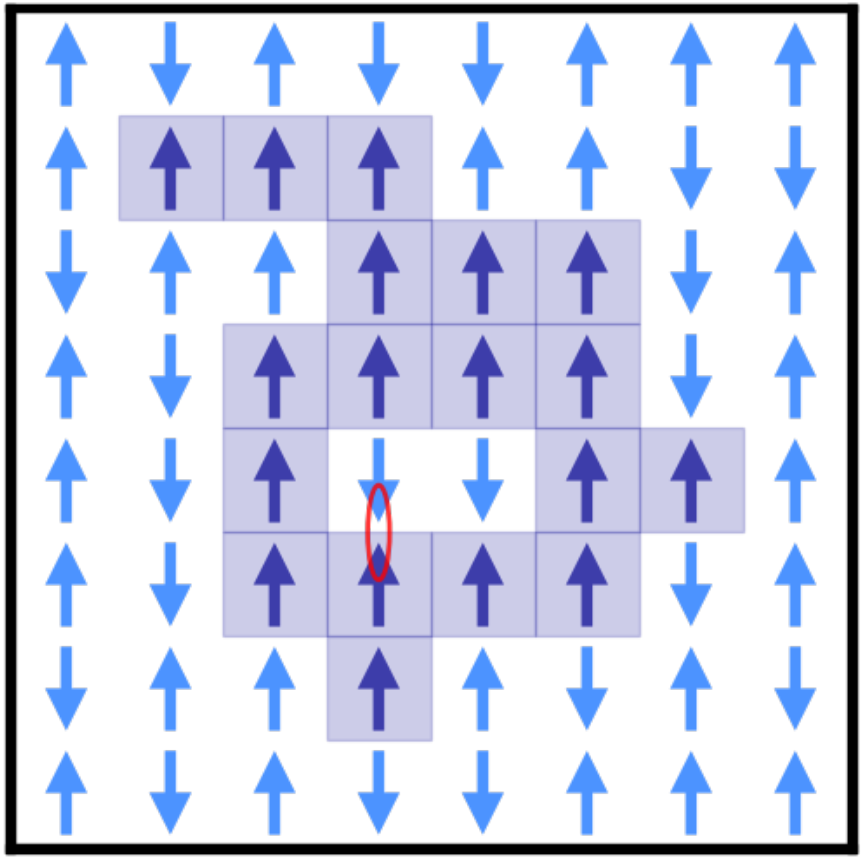
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



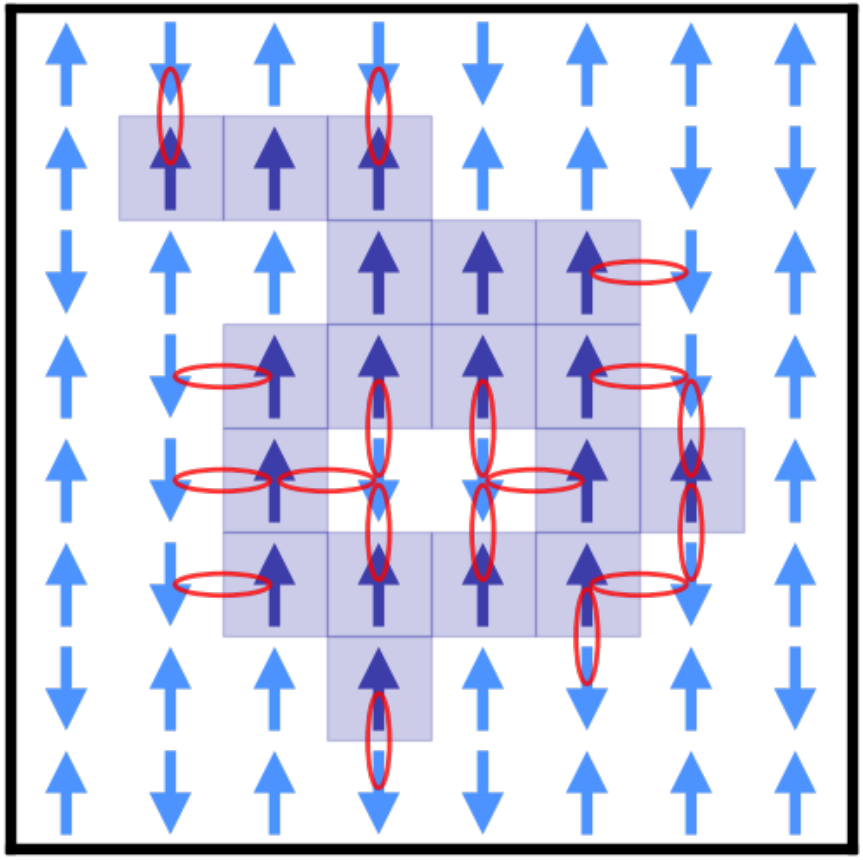
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



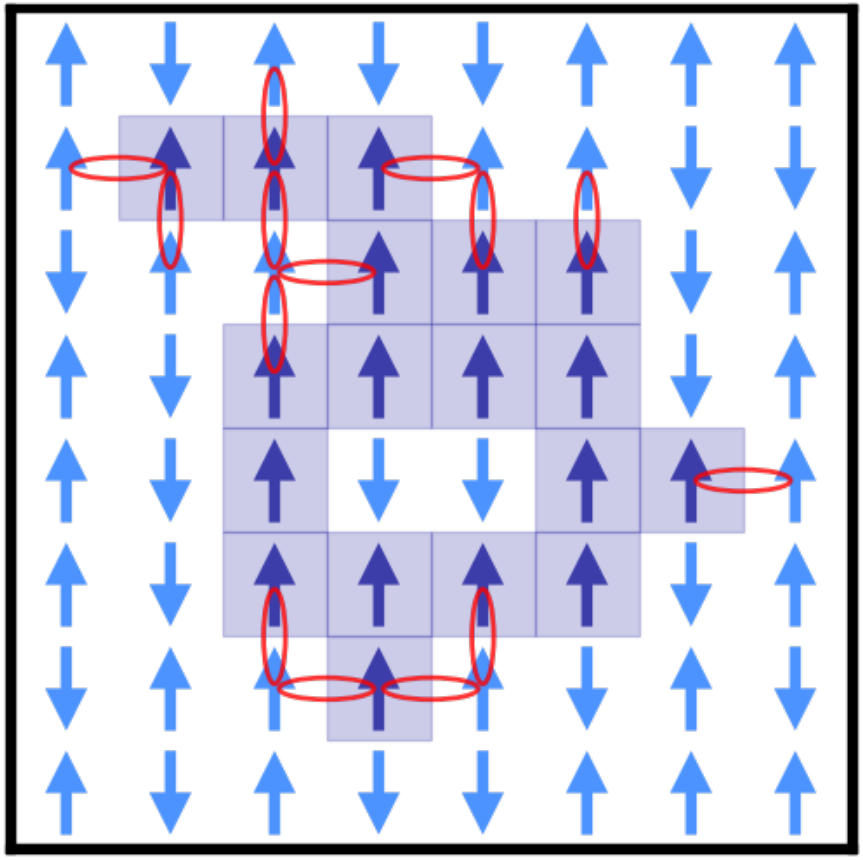
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



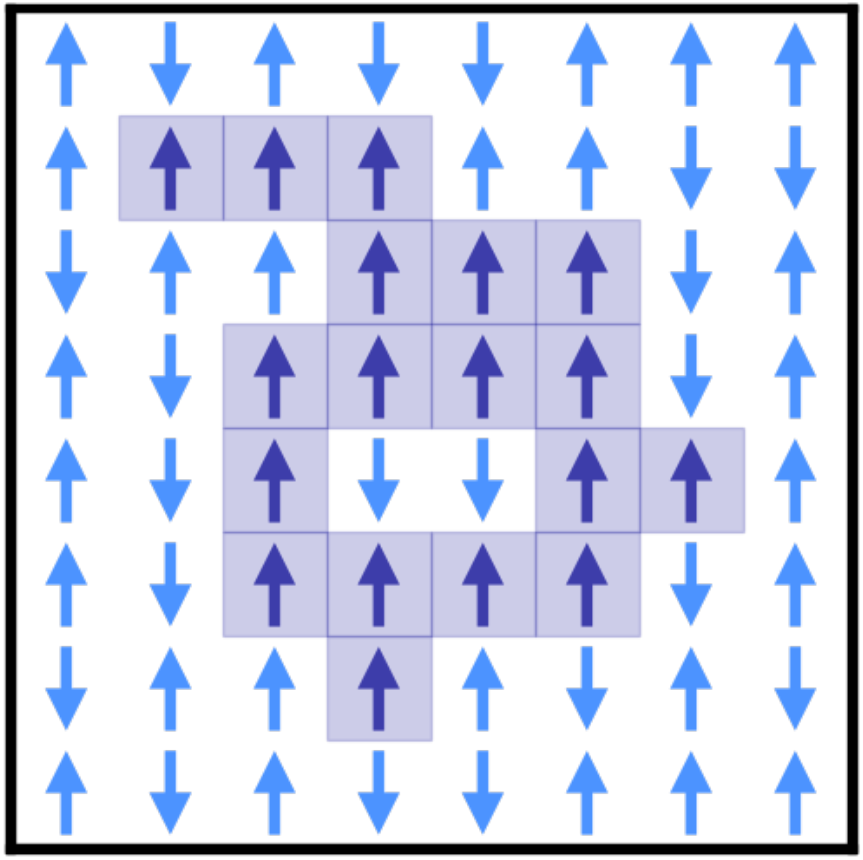
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



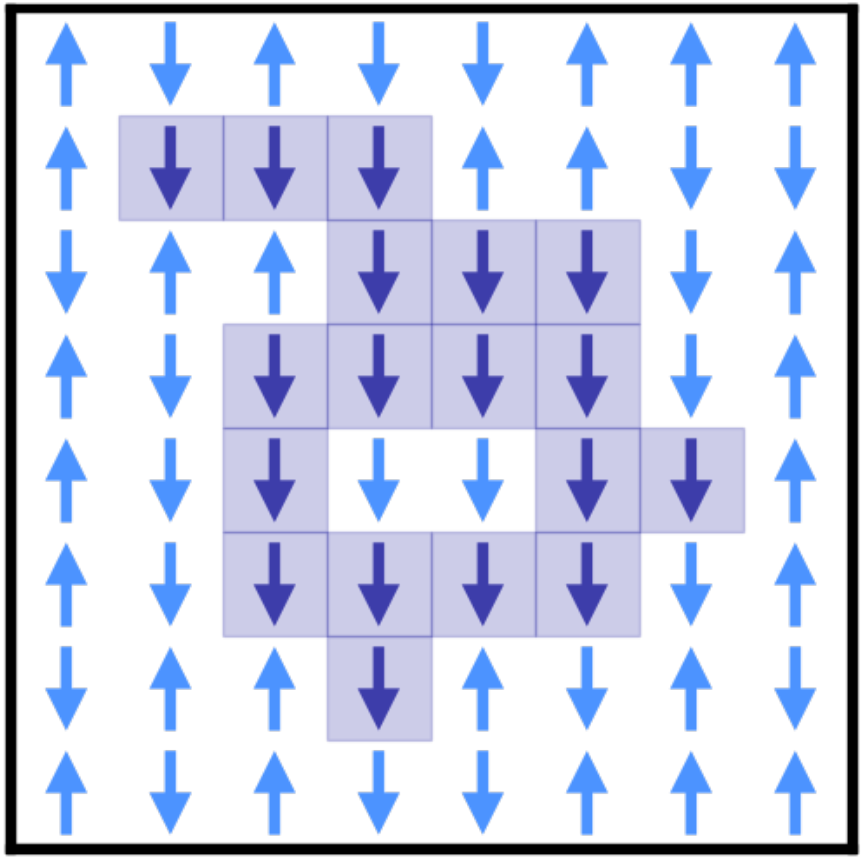
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



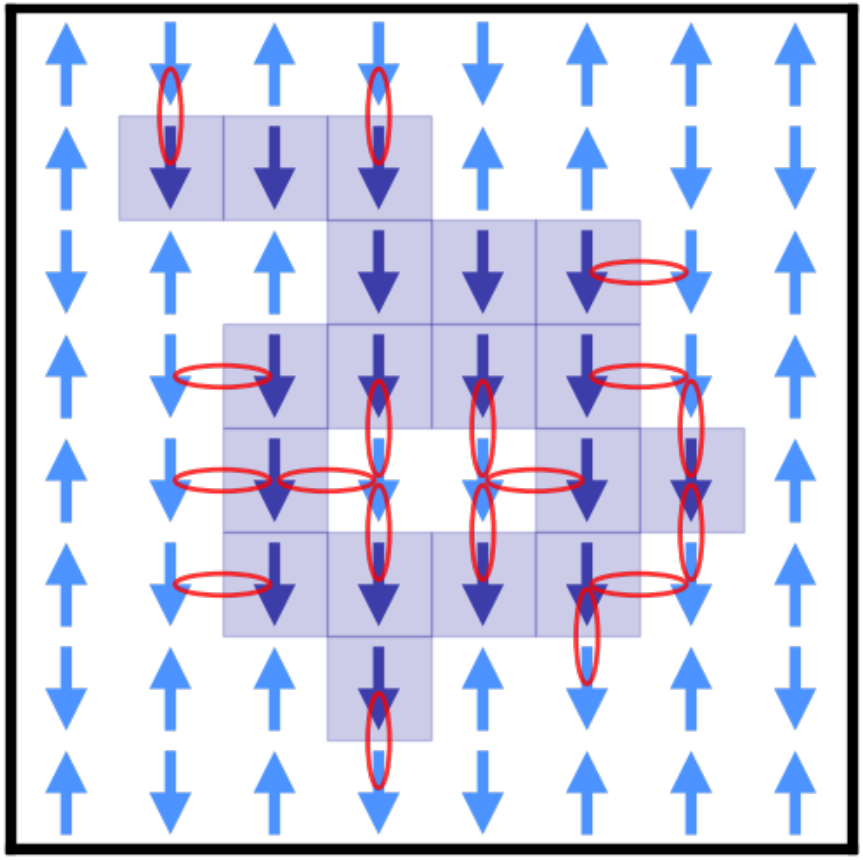
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



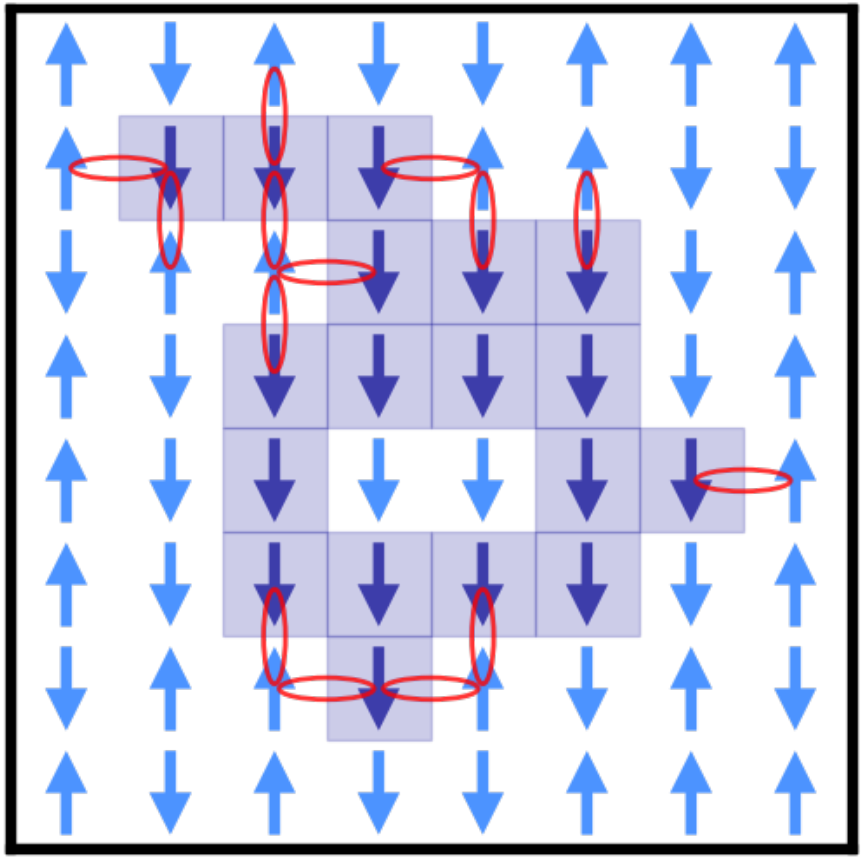
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



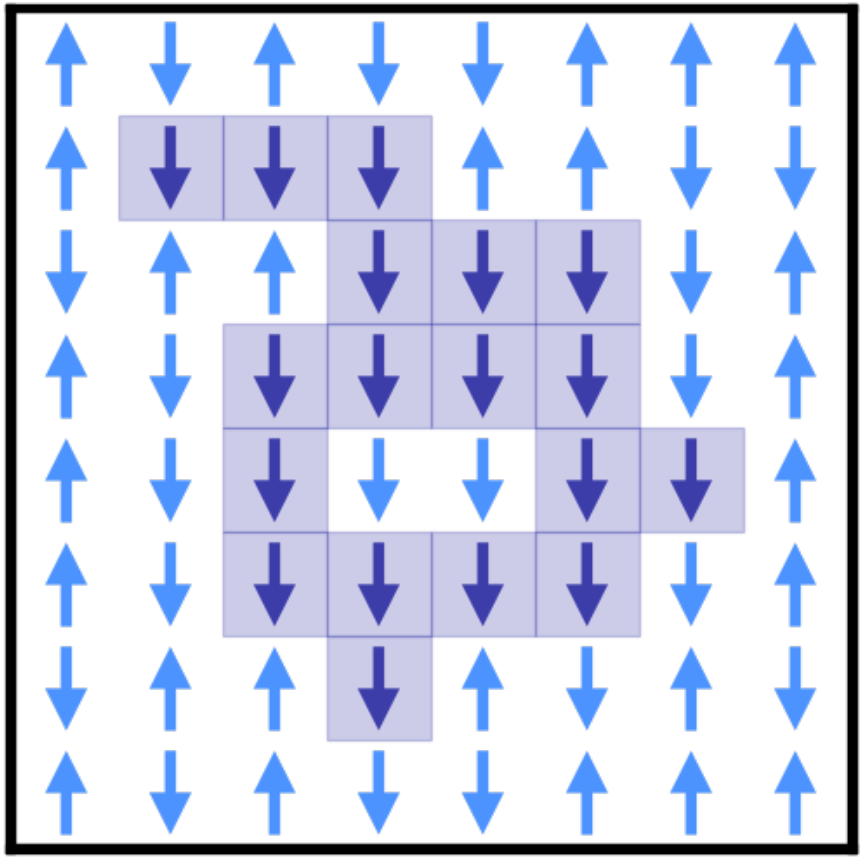
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



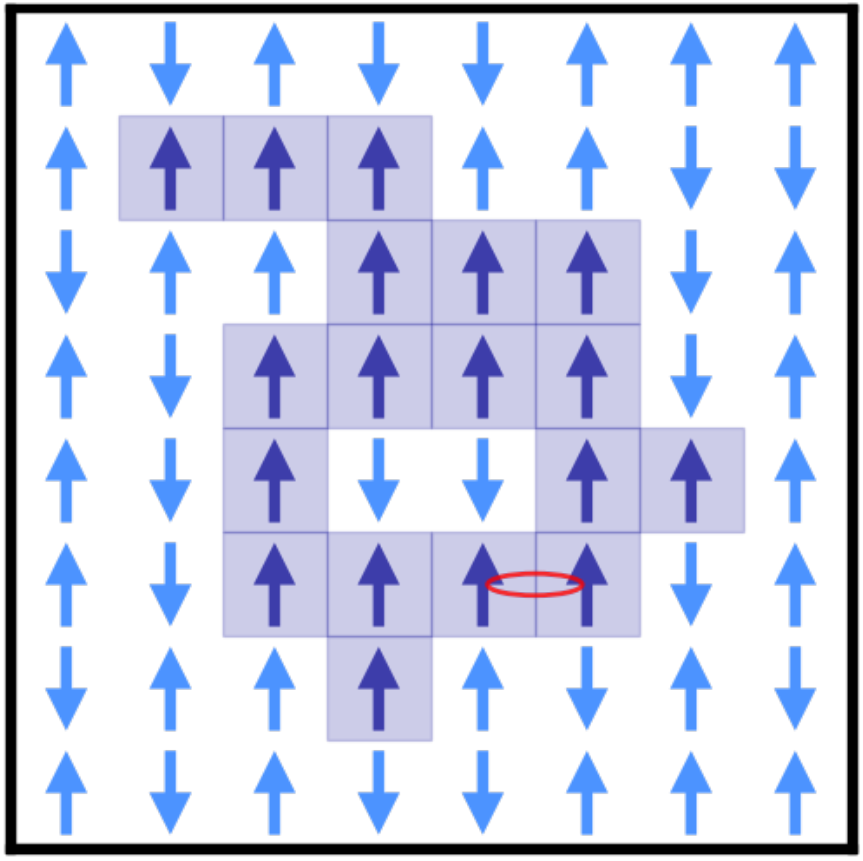
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



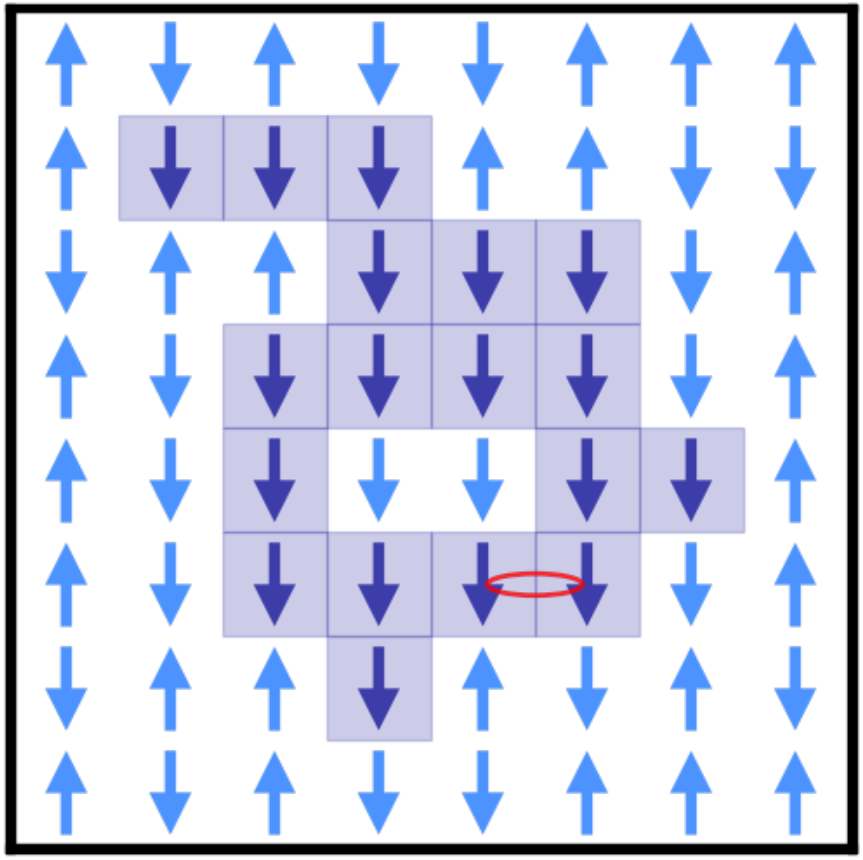
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



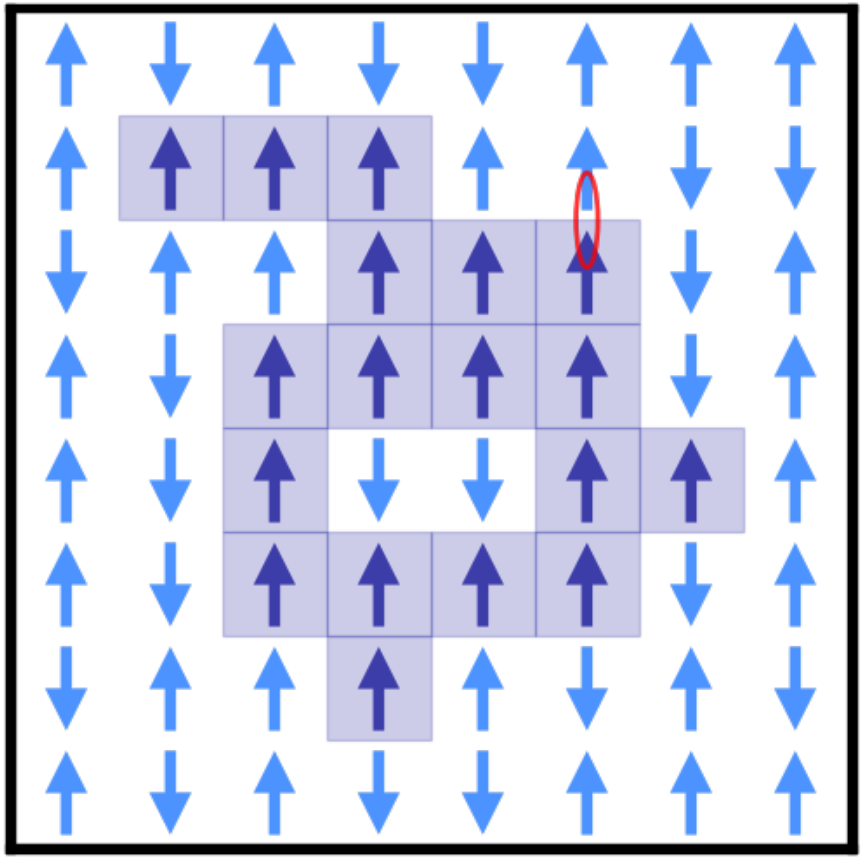
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



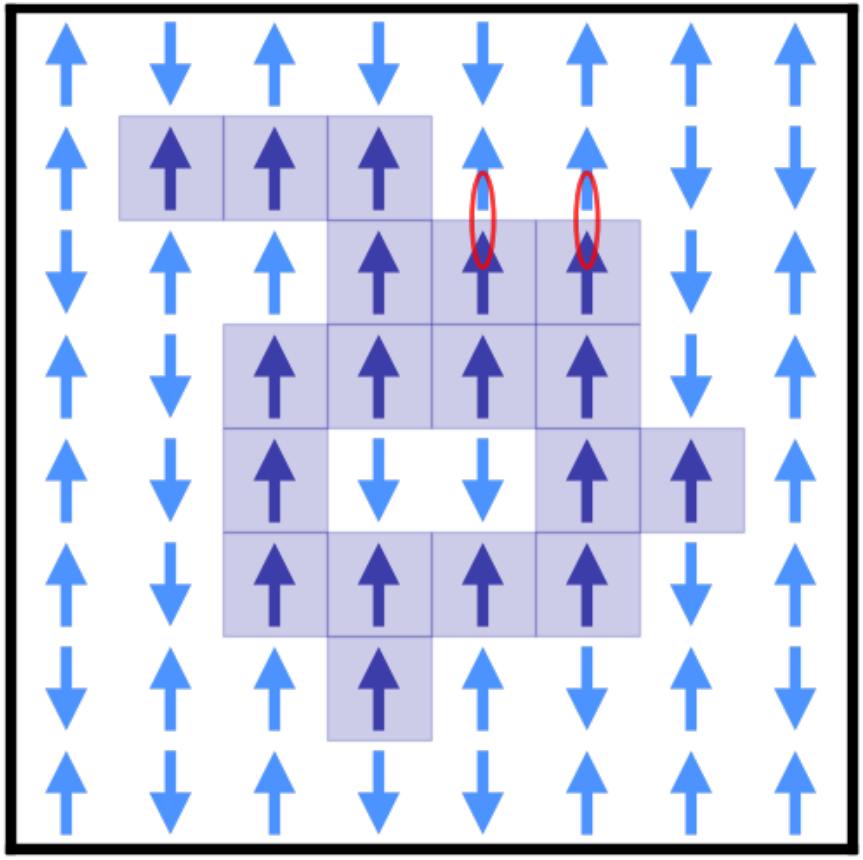
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



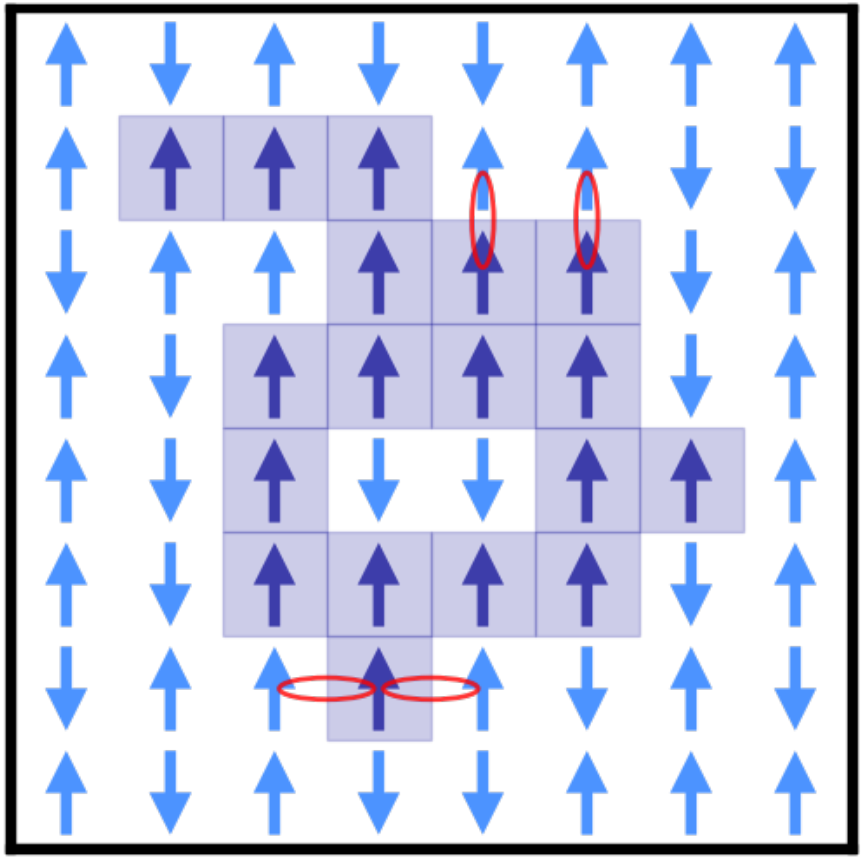
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



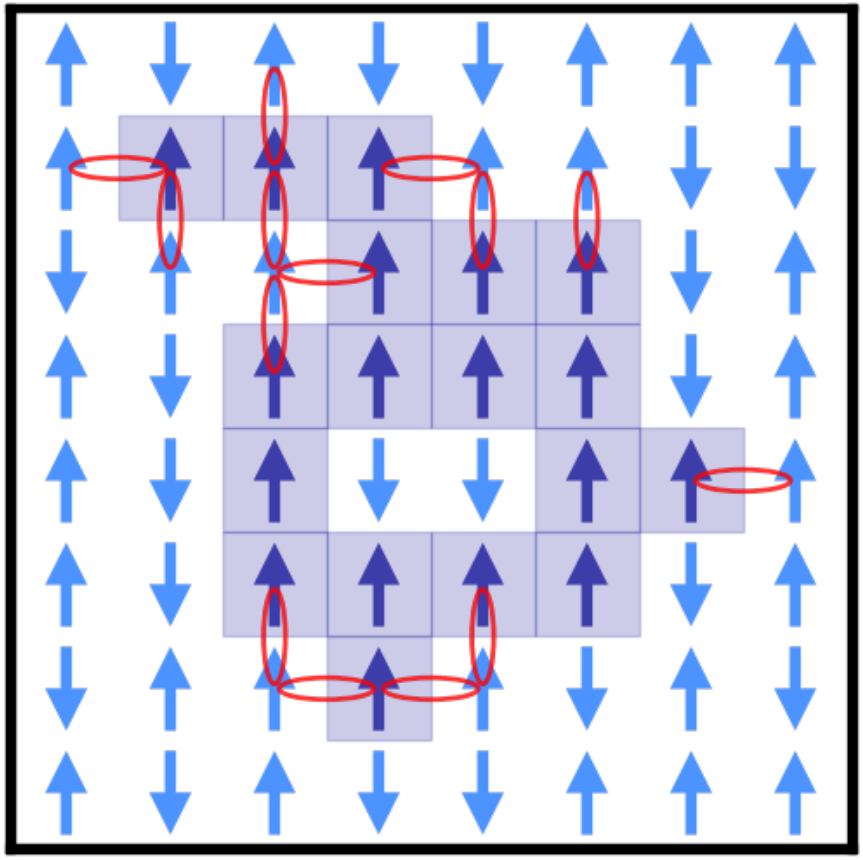
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



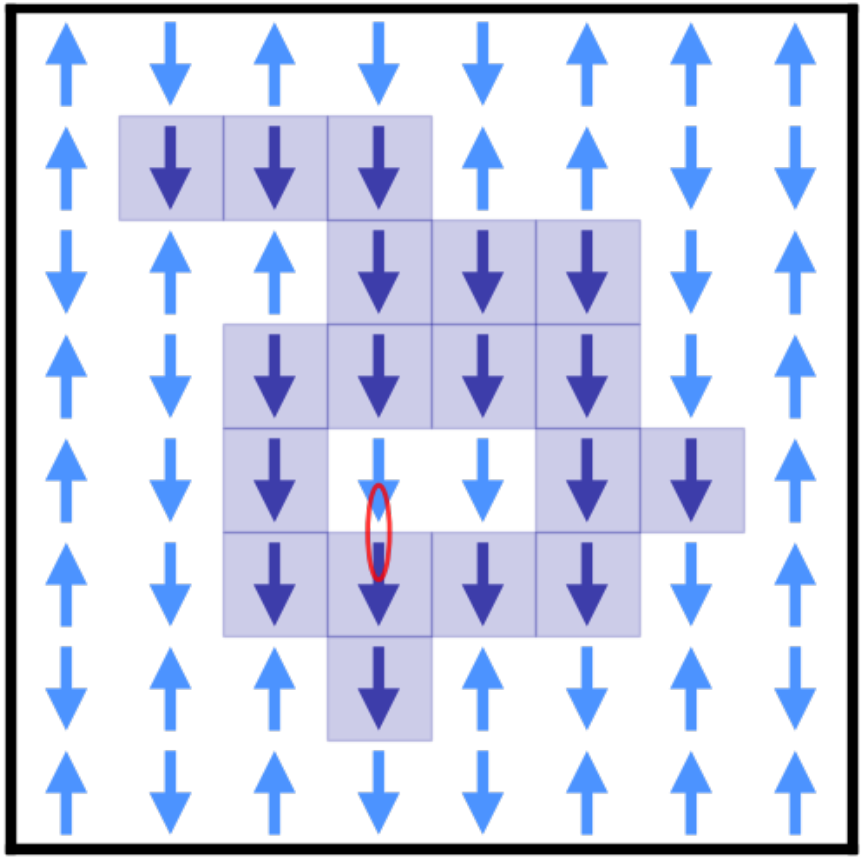
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)

$$\mathcal{A}(a \to b) \propto (1-p)^{14}$$

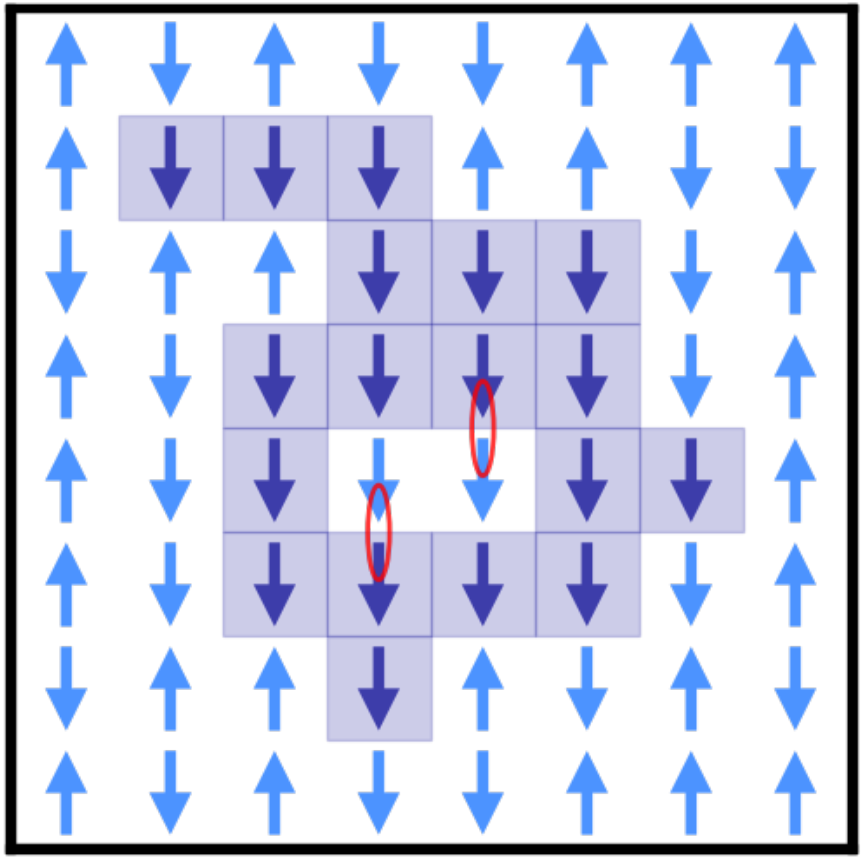
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



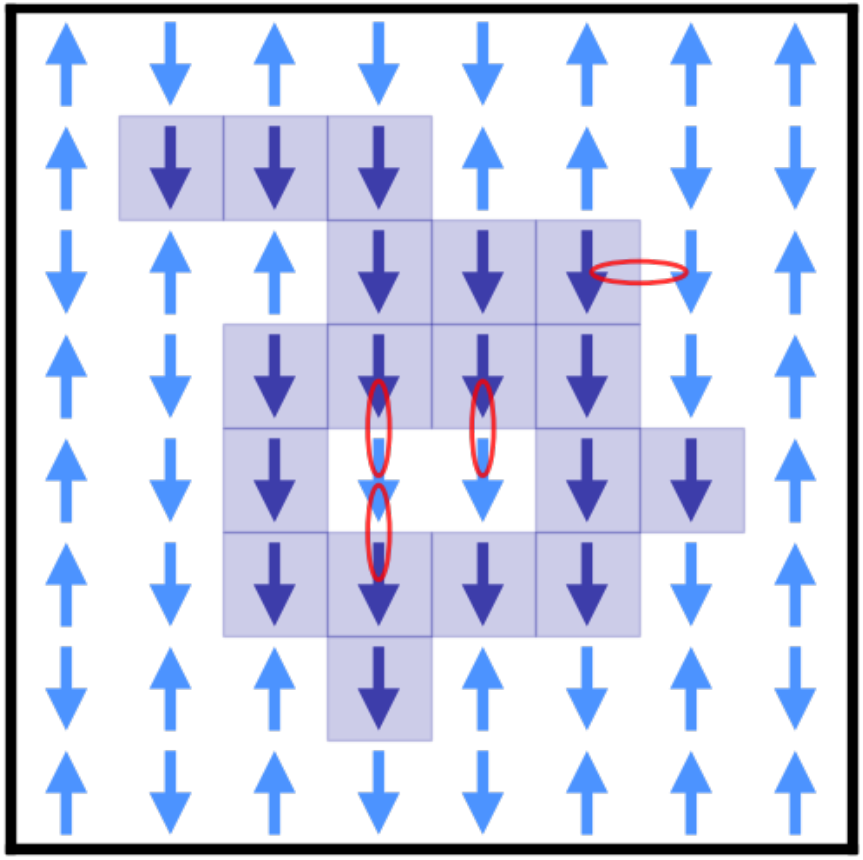
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



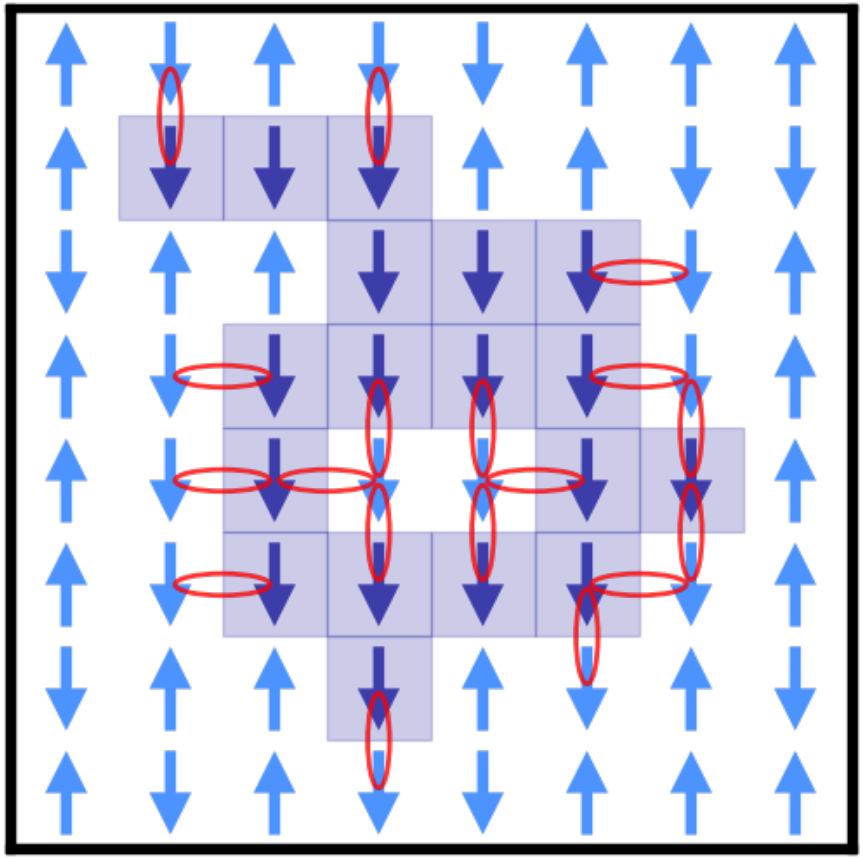
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



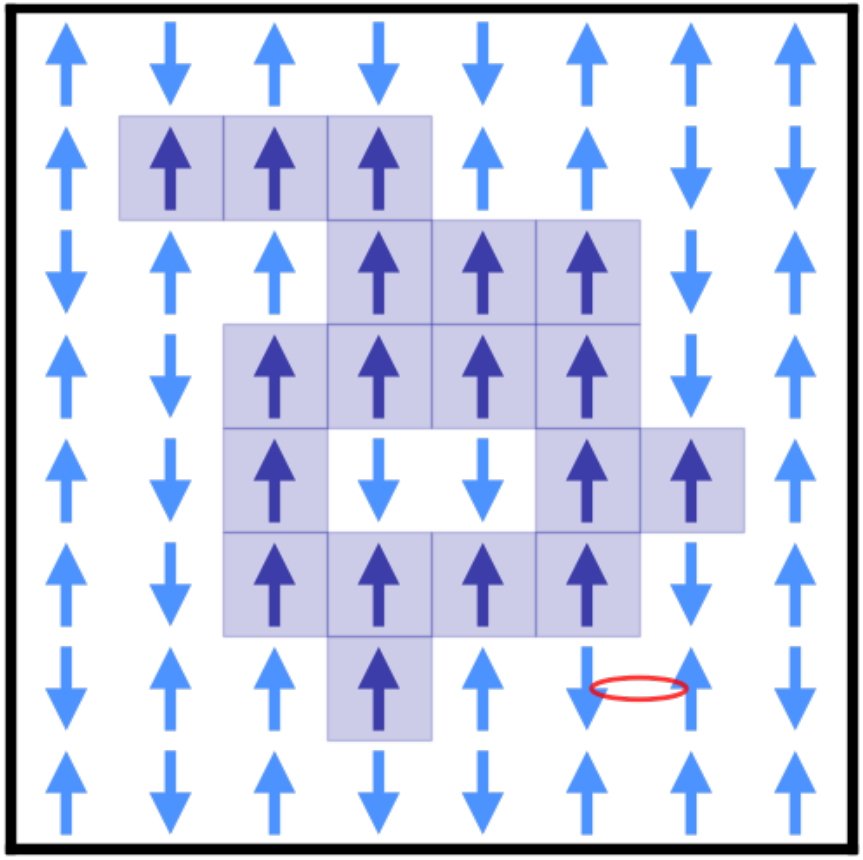
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)

$$\mathcal{A}(b \to a) \propto (1-p)^{18}$$

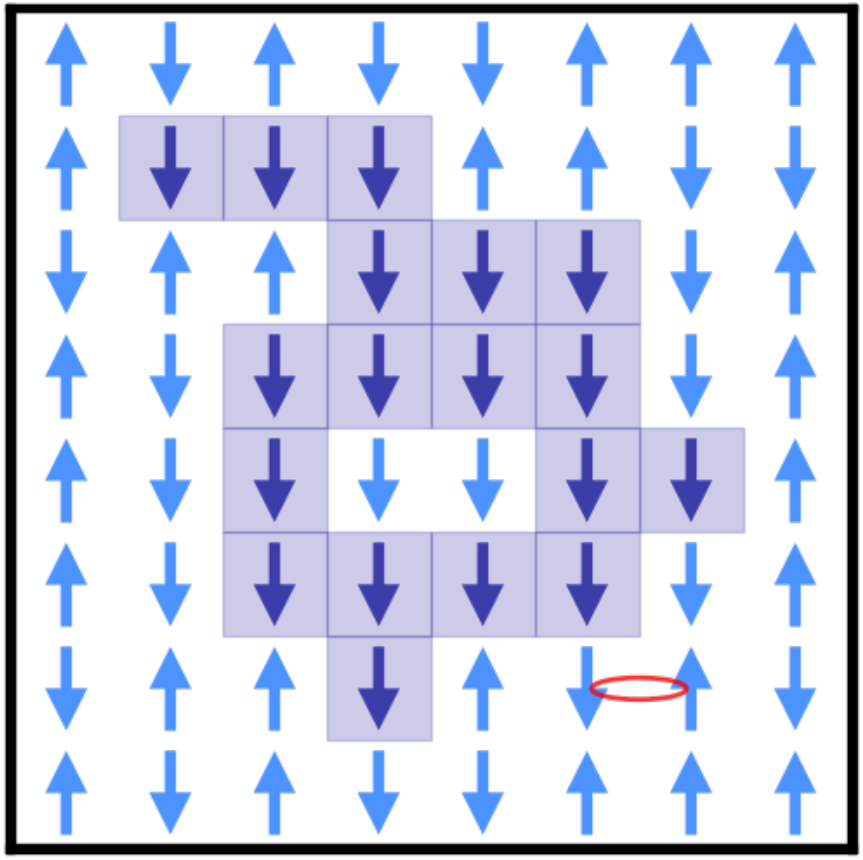
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



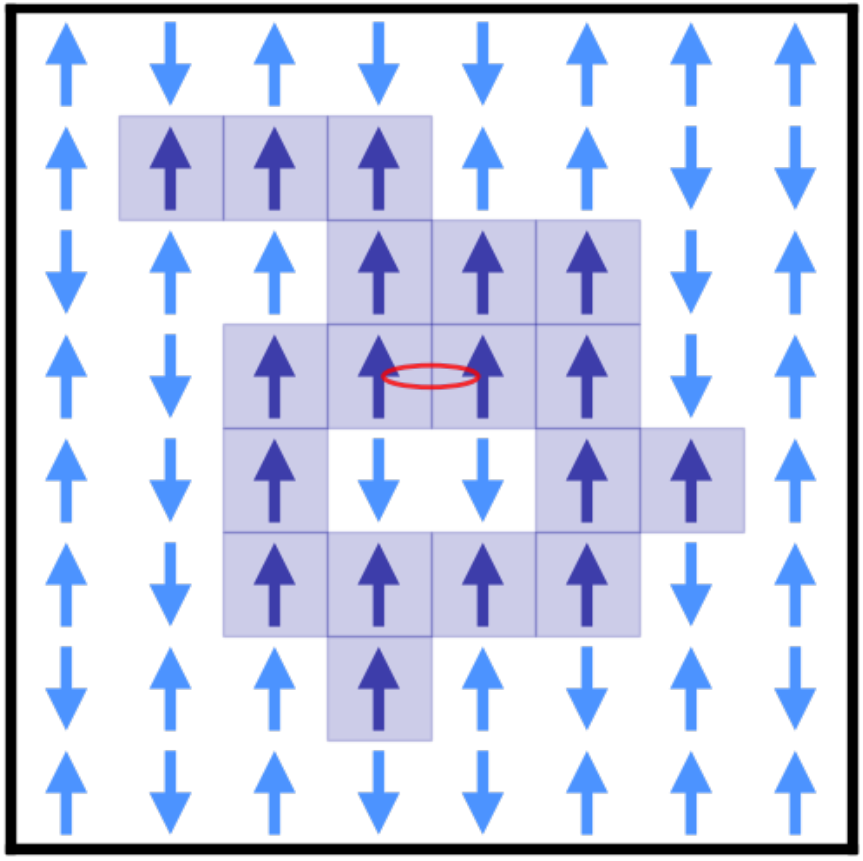
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



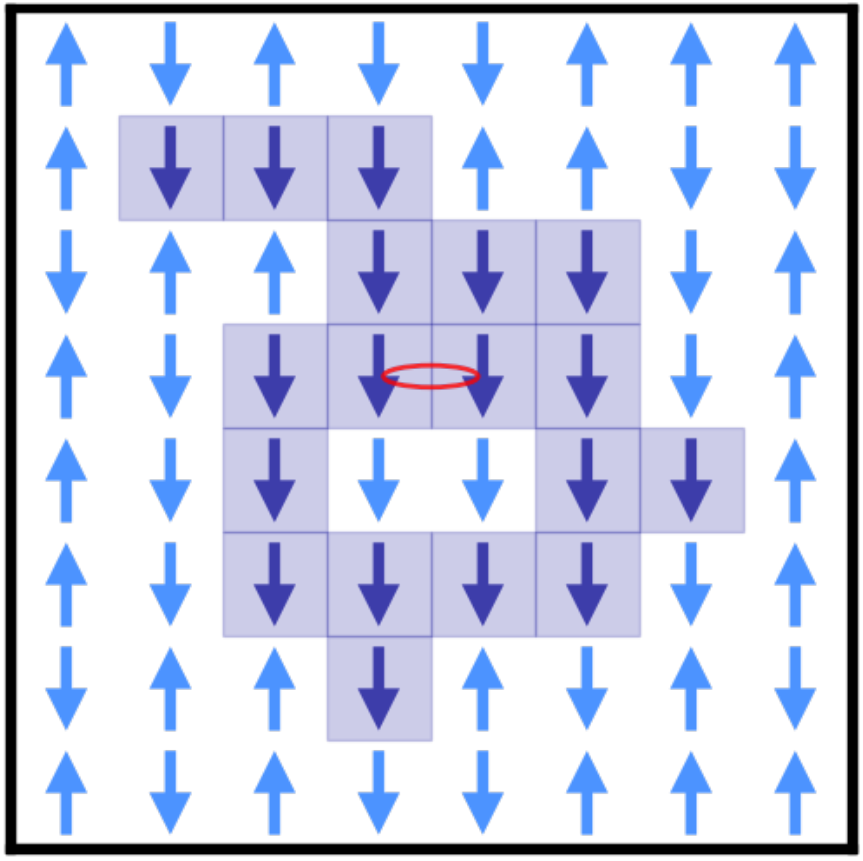
Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)



Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)

$$\text{energy of the boundary}$$

$$=$$

$$n_1 - n_2$$

Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)

$$\text{energy of the boundary}$$

$$=$$

$$n_2 - n_1$$

Département de Physique
École normale supérieure

Cluster algorithm, probabilistic (Wolff, 1989)

$$n_1 = 18, \quad n_2 = 14$$

Département de Physique
École normale supérieure

Detailed balance (Metropolis-Hastings)

$$e^{-\beta(n_1-n_2)}(1-p)^{n_2}p^{\mathrm{acc}}(a\to b) = e^{-\beta(n_2-n_1)}(1-p)^{n_1}p^{\mathrm{acc}}(b\to a)$$

Département de Physique
École normale supérieure

Detailed balance (Metropolis-Hastings)

$$e^{-\beta(n_1-n_2)}(1-p)^{n_2}p^{\mathrm{acc}}(a\to b) = e^{-\beta(n_2-n_1)}(1-p)^{n_1}p^{\mathrm{acc}}(b\to a)$$

$\pi(a)$ (depends on energy across boundaries)

Département de Physique
École normale supérieure

Detailed balance (Metropolis-Hastings)

$$e^{-\beta(n_1-n_2)}(1-p)^{n_2}p^{\mathrm{acc}}(a\to b) = e^{-\beta(n_2-n_1)}(1-p)^{n_1}p^{\mathrm{acc}}(b\to a)$$

$\mathcal{A}(a\to b)$ (probability to stop construction)

$\pi(a)$ (depends on energy across boundaries)

Département de Physique
École normale supérieure

Detailed balance (Metropolis-Hastings)

$$e^{-\beta(n_1-n_2)}(1-p)^{n_2}p^{\mathrm{acc}}(a\to b) = e^{-\beta(n_2-n_1)}(1-p)^{n_1}p^{\mathrm{acc}}(b\to a)$$

- $\mathcal{A}(a\to b)$ (probability to stop construction)
- $\pi(a)$ (depends on energy across boundaries)

$$p^{\mathrm{acc}}(a\to b) = \min\left[1, \frac{e^{-\beta(n_2-n_1)}(1-p)^{n_1}}{e^{-\beta(n_1-n_2)}(1-p)^{n_2}}\right]$$

Département de Physique
École normale supérieure

Detailed balance (Metropolis-Hastings)

$$e^{-\beta(n_1-n_2)}(1-p)^{n_2}p^{\mathrm{acc}}(a\to b) = e^{-\beta(n_2-n_1)}(1-p)^{n_1}p^{\mathrm{acc}}(b\to a)$$

$\mathcal{A}(a\to b)$ (probability to stop construction)

$\pi(a)$ (depends on energy across boundaries)

$$p^{\mathrm{acc}}(a\to b) = \min\left[1, \frac{e^{-\beta(n_2-n_1)}(1-p)^{n_1}}{e^{-\beta(n_1-n_2)}(1-p)^{n_2}}\right]$$

$$= \min\left[1, \left(\frac{e^{-2\beta}}{1-p}\right)^{n_2}\left(\frac{1-p}{e^{-2\beta}}\right)^{n_1}\right]$$

Département de Physique
École normale supérieure

Detailed balance (Metropolis-Hastings)

$$e^{-\beta(n_1-n_2)}(1-p)^{n_2}p^{\mathrm{acc}}(a\to b) = e^{-\beta(n_2-n_1)}(1-p)^{n_1}p^{\mathrm{acc}}(b\to a)$$

$\mathcal{A}(a\to b)$ (probability to stop construction)

$\pi(a)$ (depends on energy across boundaries)

$$p^{\mathrm{acc}}(a\to b) = \min\left[1, \frac{e^{-\beta(n_2-n_1)}(1-p)^{n_1}}{e^{-\beta(n_1-n_2)}(1-p)^{n_2}}\right]$$

$$= \min\left[1, \left(\frac{e^{-2\beta}}{1-p}\right)^{n_2}\left(\frac{1-p}{e^{-2\beta}}\right)^{n_1}\right]$$

$1-p = e^{-2\beta}$ magic value

Département de Physique
École normale supérieure

Detailed balance (Metropolis-Hastings)

$$e^{-\beta(n_1-n_2)}(1-p)^{n_2}p^{\mathrm{acc}}(a\to b) = e^{-\beta(n_2-n_1)}(1-p)^{n_1}p^{\mathrm{acc}}(b\to a)$$

$\mathcal{A}(a\to b)$ (probability to stop construction)

$\pi(a)$ (depends on energy across boundaries)

$$p^{\mathrm{acc}}(a\to b) = \min\left[1, \frac{e^{-\beta(n_2-n_1)}(1-p)^{n_1}}{e^{-\beta(n_1-n_2)}(1-p)^{n_2}}\right]$$

$$= \min\left[1, \left(\frac{e^{-2\beta}}{1-p}\right)^{n_2}\left(\frac{1-p}{e^{-2\beta}}\right)^{n_1}\right]$$

$1-p = e^{-2\beta}$ magic value

Département de Physique
École normale supérieure

Detailed balance (Metropolis-Hastings)

$$e^{-\beta(n_1-n_2)}(1-p)^{n_2}p^{\mathrm{acc}}(a\to b) = e^{-\beta(n_2-n_1)}(1-p)^{n_1}p^{\mathrm{acc}}(b\to a)$$

- $\mathcal{A}(a\to b)$ (probability to stop construction)
- $\pi(a)$ (depends on energy across boundaries)

$$p^{\mathrm{acc}}(a\to b) = \min\left[1, \frac{e^{-\beta(n_2-n_1)}(1-p)^{n_1}}{e^{-\beta(n_1-n_2)}(1-p)^{n_2}}\right]$$

$$= 1$$

$$1-p = e^{-2\beta} \qquad \text{magic value}$$

Département de Physique
École normale supérieure

cluster-ising.py

```
import random, math

L = 100
N = L * L
nbr = {i : ((i // L) * L + (i + 1) % L, (i + L) % N,
            (i // L) * L + (i - 1) % L, (i - L) % N)
                                        for i in range(N)}
T = 2.5
p  = 1.0 - math.exp(-2.0 / T)
nsteps = 10000
S = [random.choice([1, -1]) for k in range(N)]
for step in range(nsteps):
    k = random.randint(0, N - 1)
    Pocket, Cluster = [k], [k]
    while Pocket != []:
        j = random.choice(Pocket)
        for l in nbr[j]:
            if S[l] == S[j] and l not in Cluster \
                    and random.uniform(0.0, 1.0) < p:
                Pocket.append(l)
                Cluster.append(l)
        Pocket.remove(j)
    for j in Cluster:
        S[j] *= -1
```

Département de Physique
École normale supérieure

Running cluster-ising.py

Metropolis algorithm (reminder)

$$\pi(a)\,\mathcal{A}(a \to b)\,p^{\mathrm{acc}}(a \to b) = \pi(b)\,\mathcal{A}(b \to a)\,p^{\mathrm{acc}}(b \to a)$$

$$\mathcal{A}(a \to b) = \mathcal{A}(b \to a)$$

Département de Physique
École normale supérieure

Metropolis algorithm (reminder)

$$\pi(a)\,\mathcal{A}(a \to b)\,p^{\mathrm{acc}}(a \to b) = \pi(b)\,\mathcal{A}(b \to a)\,p^{\mathrm{acc}}(b \to a)$$

$$\mathcal{A}(a \to b) = \mathcal{A}(b \to a)$$

Département de Physique
École normale supérieure

Heatbath algorithm

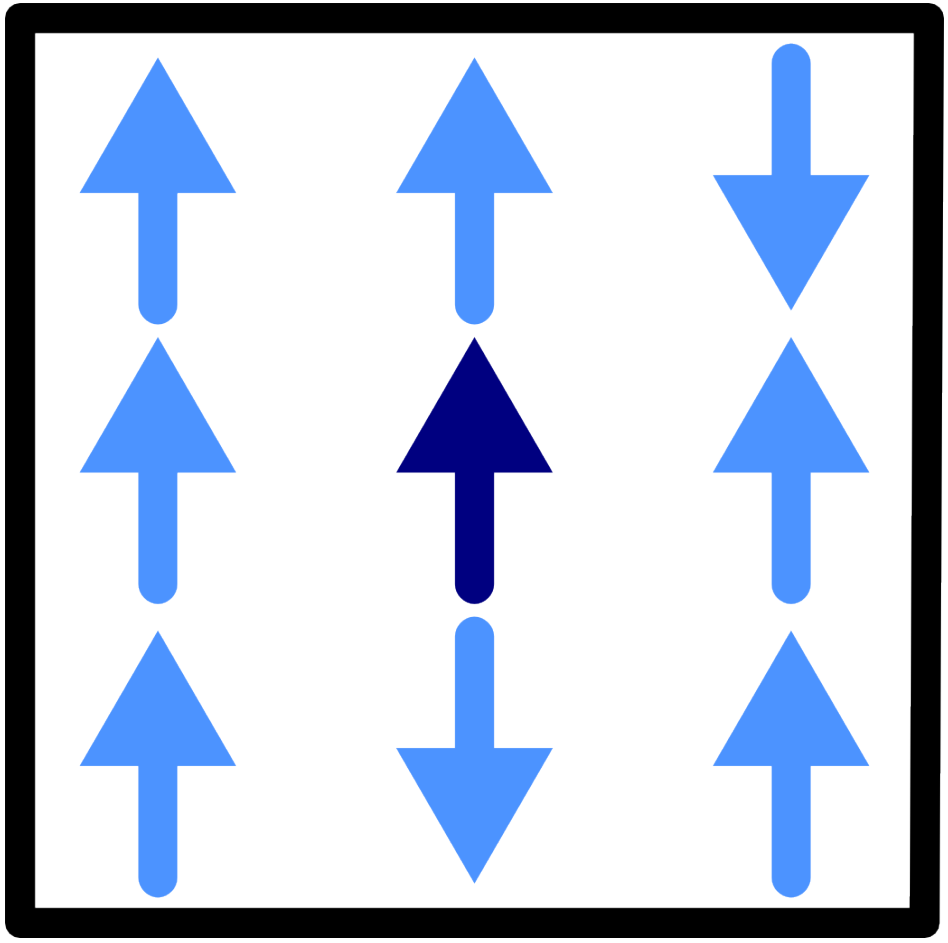
$$\pi(a)\,\mathcal{A}(a \to b)\,p^{\mathrm{acc}}(a \to b) = \pi(b)\,\mathcal{A}(b \to a)\,p^{\mathrm{acc}}(b \to a)$$

$$\mathcal{A}(a \to b) \propto \pi(b); \quad \mathcal{A}(b \to a) \propto \pi(a)$$

- Acceptance probabilities $= 1$
- Apply to subsystem

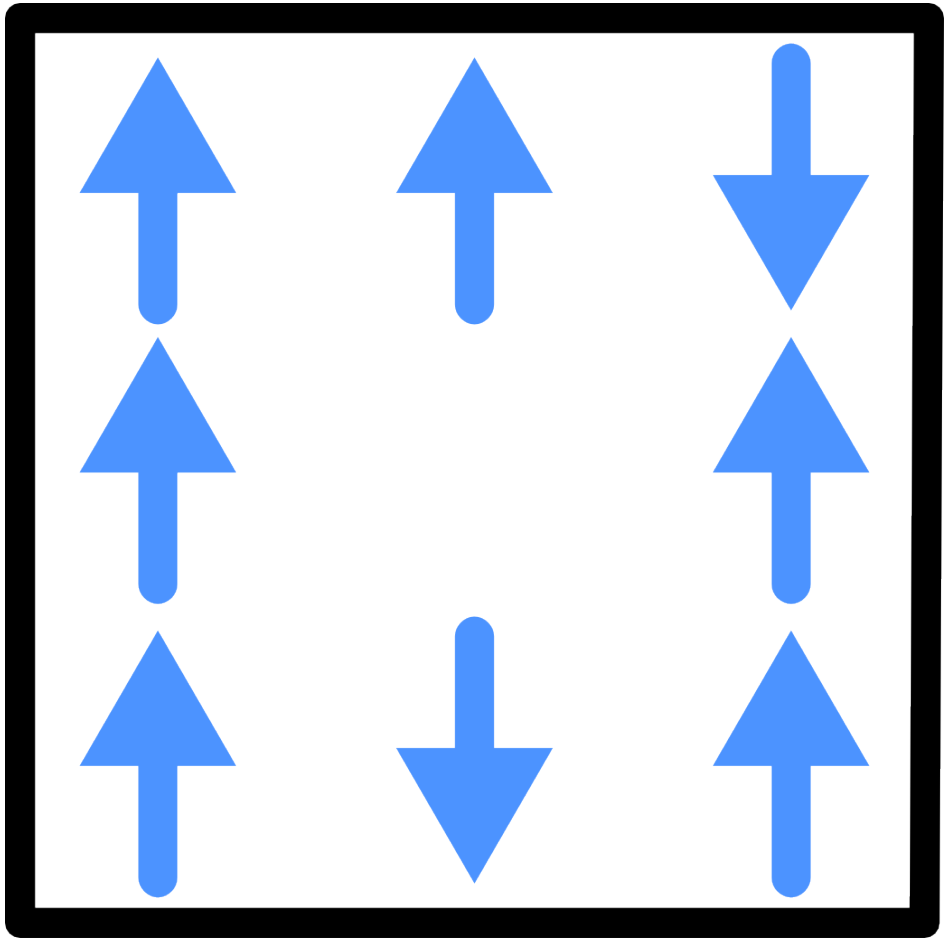
Département de Physique
École normale supérieure

Heatbath algorithm

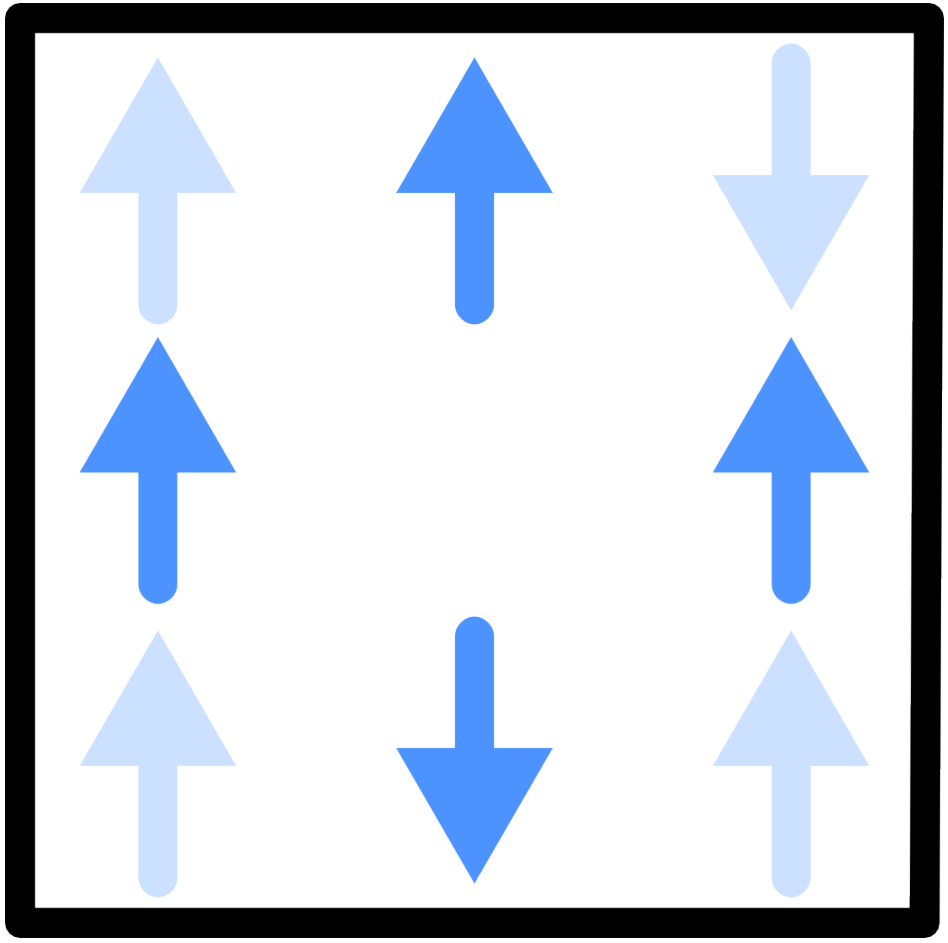


Département de Physique
École normale supérieure

Heatbath algorithm



Heatbath algorithm



Heatbath algorithm

$$\pi^+ = \frac{\mathrm{e}^{\beta h}}{\mathrm{e}^{-\beta h} + \mathrm{e}^{\beta h}}$$

$$\pi^- = \frac{\mathrm{e}^{-\beta h}}{\mathrm{e}^{-\beta h} + \mathrm{e}^{\beta h}}$$

Département de Physique
École normale supérieure

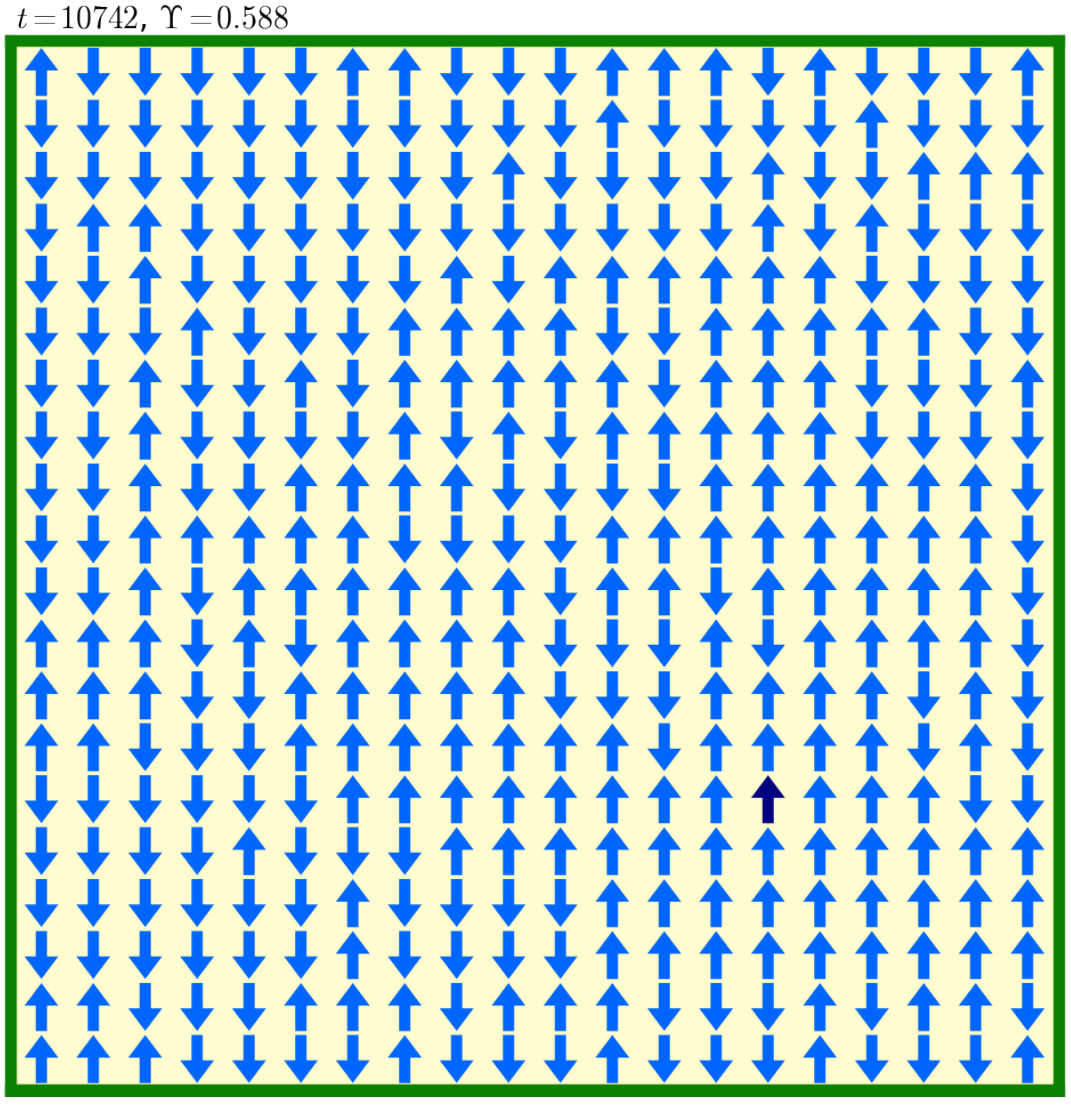
heatbath-ising.py

```
import random, math

L = 16
N = L * L
nbr = {i : ((i // L) * L + (i + 1) % L, (i + L) % N,
            (i // L) * L + (i - 1) % L, (i - L) % N) \
                                        for i in range(N)}
nsteps = 100000
beta = 0.5
S = [random.choice([1, -1]) for i in range(N)]
random.seed('123456')
for step in range(nsteps):
    k = random.randrange(N)
    Upsilon = random.uniform(0.0, 1.0)
    h = sum(S[nn] for nn in nbr[k])
    S[k] = -1
    if Upsilon < 1.0 / (1.0 + math.exp(-2.0 * beta * h)):
        S[k] = 1
```

Running heatbath-ising.py from all up

Running heatbath-ising.py from all down

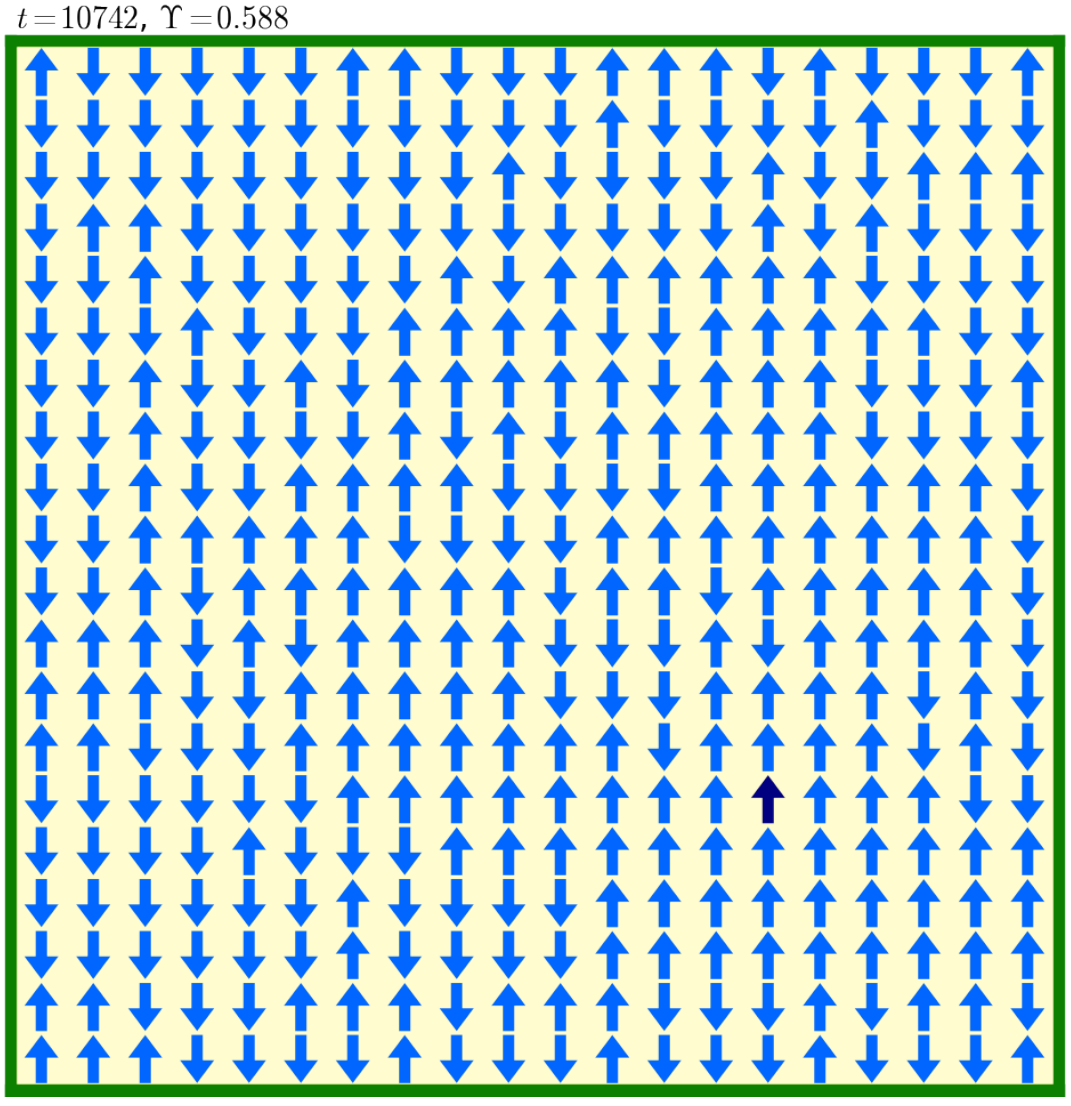
final configuration "up"



- Propp & Wilson 1996

Département de Physique
École normale supérieure

final configuration "down"



- Propp & Wilson 1996

Département de Physique
École normale supérieure

Coupling in heatbath-ising.py

```
import random, math

L = 16
N = L * L
nbr = {i : ((i // L) * L + (i + 1) % L, (i + L) % N,
            (i // L) * L + (i - 1) % L, (i - L) % N) \
                                        for i in range(N)}
nsteps = 100000
beta = 0.5
S = [random.choice([1, -1]) for i in range(N)]
random.seed('123456')
for step in range(nsteps):
    k = random.randrange(N)
    Upsilon = random.uniform(0.0, 1.0)
    h = sum(S[nn] for nn in nbr[k])
    S[k] = -1
    if Upsilon < 1.0 / (1.0 + math.exp(-2.0 * beta * h)):
        S[k] = 1
```

Conclusion

Concepts considered:

- Enumeration (Counting / Listing)
- Metropolis Markov-chain sampling
- Cluster algorithm & a priori probabilities
- Heat-bath algorithm
- Coupling

Algorithms considered:

- enumerate-ising.py (using Gray code)
- thermo-ising.py (using density of states)
- markov-ising.py (Metropolis algorithm)
- cluster-ising.py (Wolff algorithm)
- heat-bath-ising.py (and coupling)

Département de Physique
École normale supérieure