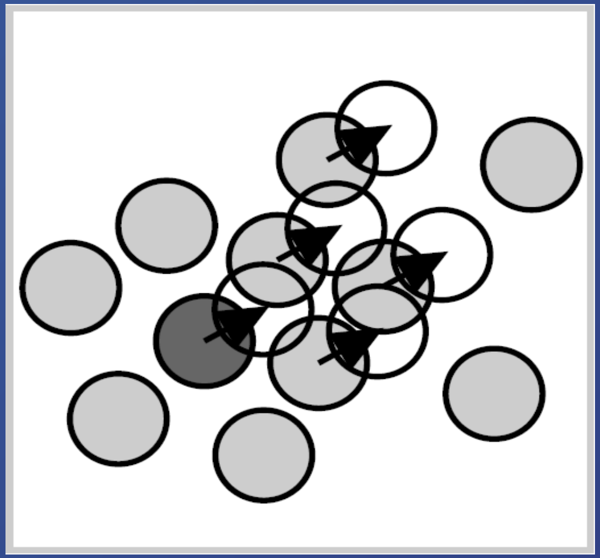
Cluster-Algorithms for off-lattice systems



Ludger Santen

Fachrichtung 7.1, Theoretische Physik
Universität des Saarlandes, Saarbrücken

FR PHYSIK

Cluster-Algorithms for off-lattice systems

- The Monte Carlo-Method: A reminder
- Cluster Algorithms
- Cluster-Algorithm of Dress & Krauth
- The Avalanche-Algorithm

FR PHYSIK

The Monte Carlo-Method

Ising-model:

$$\begin{matrix} + & + & - & - \\ + & - & - & + \\ - & - & + & - \\ + & + & + & + \end{matrix}$$

Hamiltonian:

$$H = -\sum_{\langle ij \rangle} s_i s_j$$

Magnetisation

$$\langle M_i \rangle = \frac{1}{Z} \sum_{\{S\}} s_i \exp(-\beta H(S))$$

Partition-function:

$$Z = \sum_{\{S\}} \exp(-\beta H(S))$$

Exact enumeration:

System-Size	**CPU-Time**
6x6	1 minute
7x7	6 weeks
8x8	7600 years
…	

➔ Exact enumeration only possible for small system sizes!

The Monte Carlo-Method

Naive method

- Random generation of configurations
- Estimate of expectation values from a finite number of configuration
- Example: Magnetisation

$$\langle m_i \rangle = p^{(1)} s_i^{(1)} + p^{(2)} s_i^{(2)} + p^{(3)} s_i^{(3)} + \ldots$$

$s_i^{(j)}$ magnetisation at site *i* in configuration *j*

Relative weight: $p^{(i)} = \exp(-\beta H(S^{(i)}))$

Problem of the method:

• A relative small fraction of the configurations determines the averages

• By unbiased sampling we always hit configurations with small weight

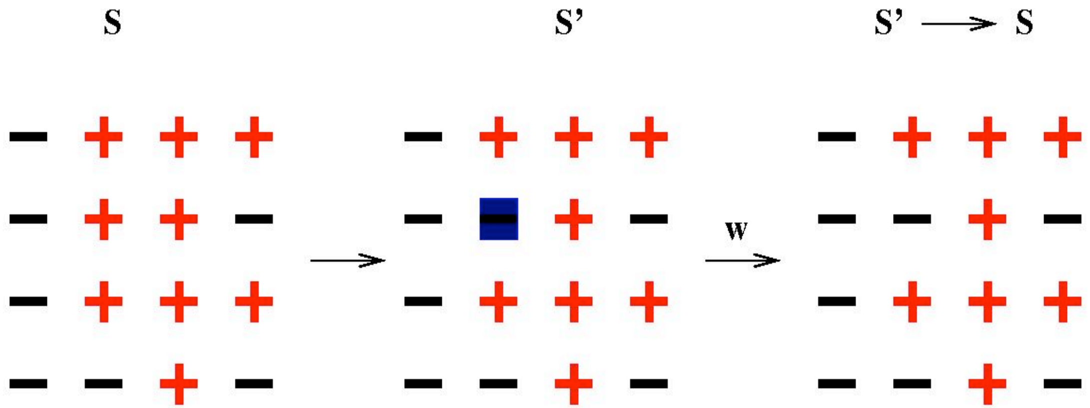
→ Biased sampling in order to hit the important ones

The Monte Carlo-Method: Importance Sampling

Idea: Generate configurationen with probability corresponding to their weight!

Realisation:

1. Generate a random configuration *S*
2. Select a site i and perform a spinflip
3. **If** spinflip lowers the energy *always* accept the move
   else accept with probability *w(S→S')*



Monte Carlo-Method: Detailed Balance

Complete algorithm: How to choose w(S→S')?

Configuration have to be generated according to their relative weight!
➔ Stationary probabilities of the stochastic process have Boltzmann-Weight

Master-equation:

$$\frac{\partial P(S,t)}{\partial t} = \sum_{S'} \left(w(S' \to S) P(S',t) - w(S \to S') P(S,t) \right)$$

Stationary solution

$$0 = \sum_{S'} \left(w(S' \to S) P(S') - w(S \to S') P(S) \right)$$

Detailed balance:

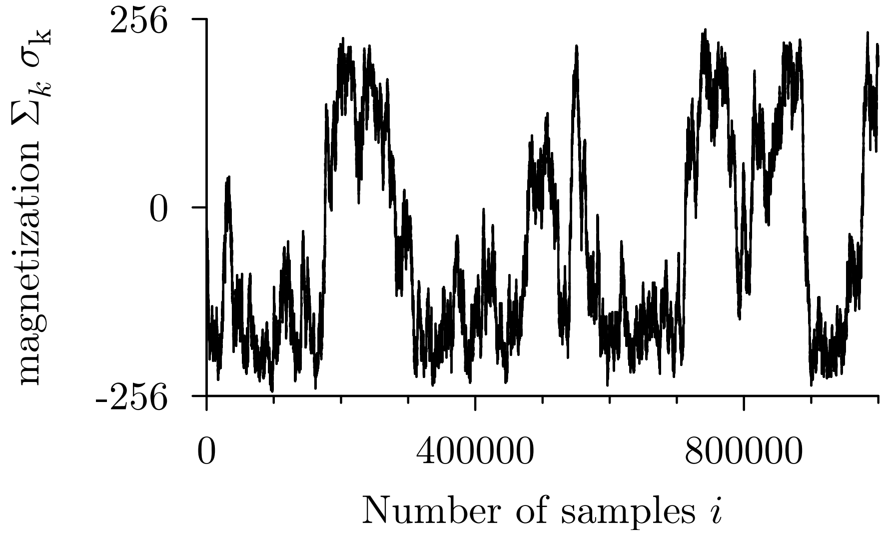
$$\frac{w(S \to S')}{w(S' \to S)} = \frac{P(S')}{P(S)} = \exp\left(-\beta \left(H(S') - H(S)\right)\right)$$

Metropolis

$$w(S \to S') = \begin{cases} \exp\left(-\beta \left(H(S') - H(S)\right)\right) & H(S') > H(S) \\ 1 & H(S') \le H(S) \end{cases}$$

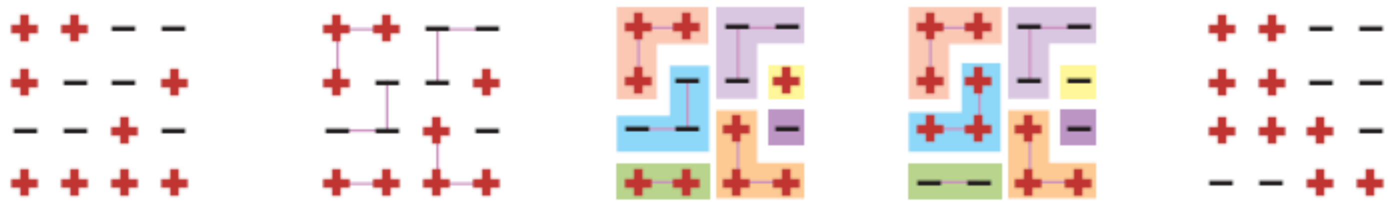
The Monte Carlo-Method

Problem of the algorithm:
Configurations depend on the history
➔ Configuration space is only partially explored



Cluster-Algorithms on a lattice

Improved Algorithms: Simultaneous update of many sites



Swendsen-Wang-Cluster-Algorithm:

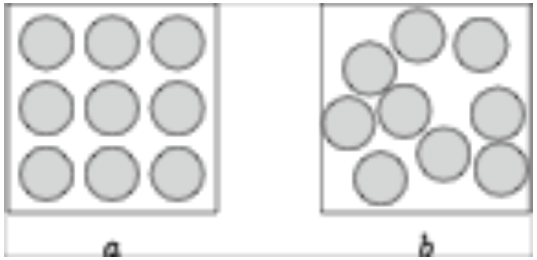
1. Set a bond with prob. $w_{ij} = 1 - \exp(-2\beta)$ neighboring spins (i,j)
2. Identification of clusters
3. Flip clusters with prob. ½
4. Delete bonds

→ Decorrelation of the configurationen + Importance Sampling

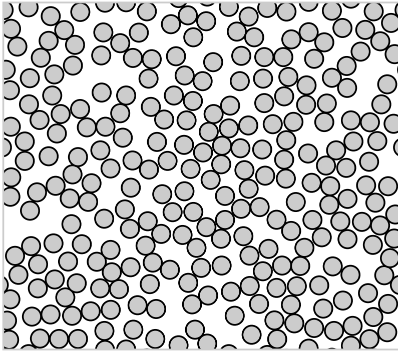
Off-Lattice Models

Hard disks as a model for 2d crystal

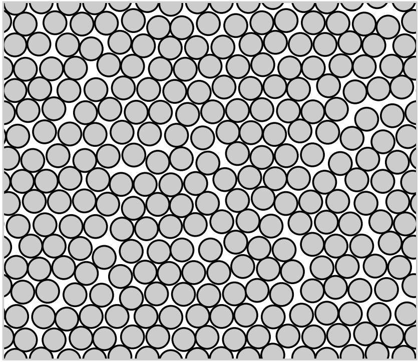
Configurations without overlap have **equal weight**



Phase transition: Liquid/crystal (bond orientational order)



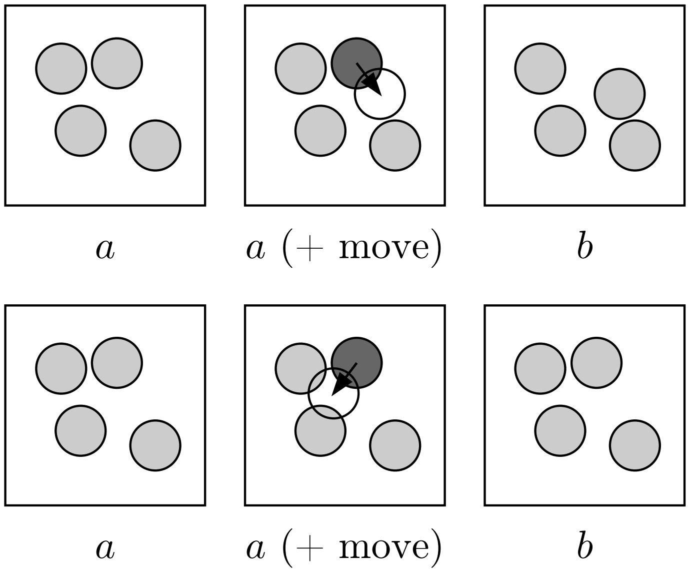
$\eta = 0.48$



$\eta = 0.72$

Off-Lattice Models

Local algorithm:



1. Select a particle and a displacement vector
2. Overlap? Reject; else: accept

Detailed balance

- All configurations without overlap have equal probality
- Inversion of the update: Inverse displacement vector, same particle

➔ Accept all moves leading to valid configurations

Off-Lattice Models

Local algorithm:

1. Select a particle and a displacement vector
2. Overlap? Reject; else: accept

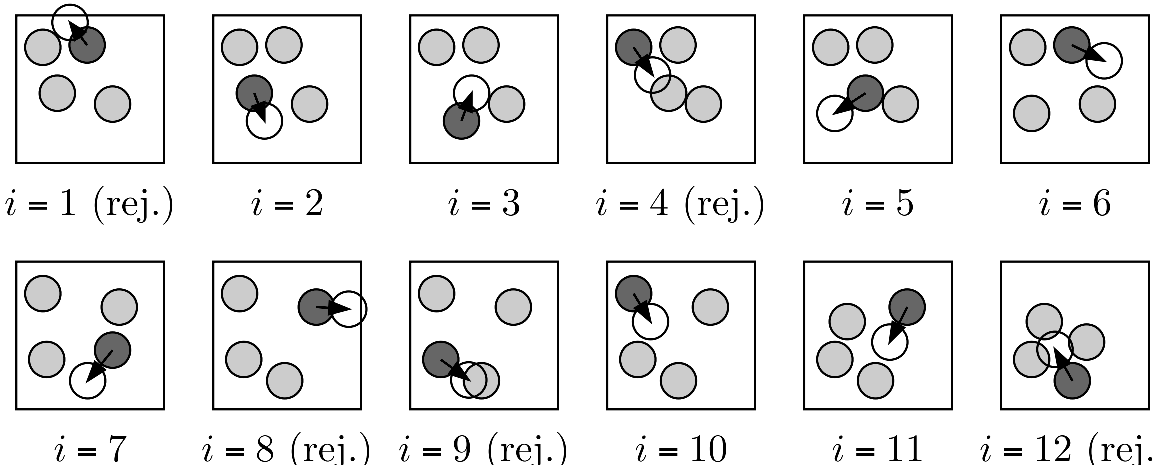
Detailed balance

- All configurations without overlap have equal probality
- Inversion of the update: Inverse displacement vector, same particle

➔ Accept all moves leading to valid configurations

Off-Lattice Models

Local algorithm:



1. Select a particle and a displacement vector
2. Overlap? Reject; else: accept

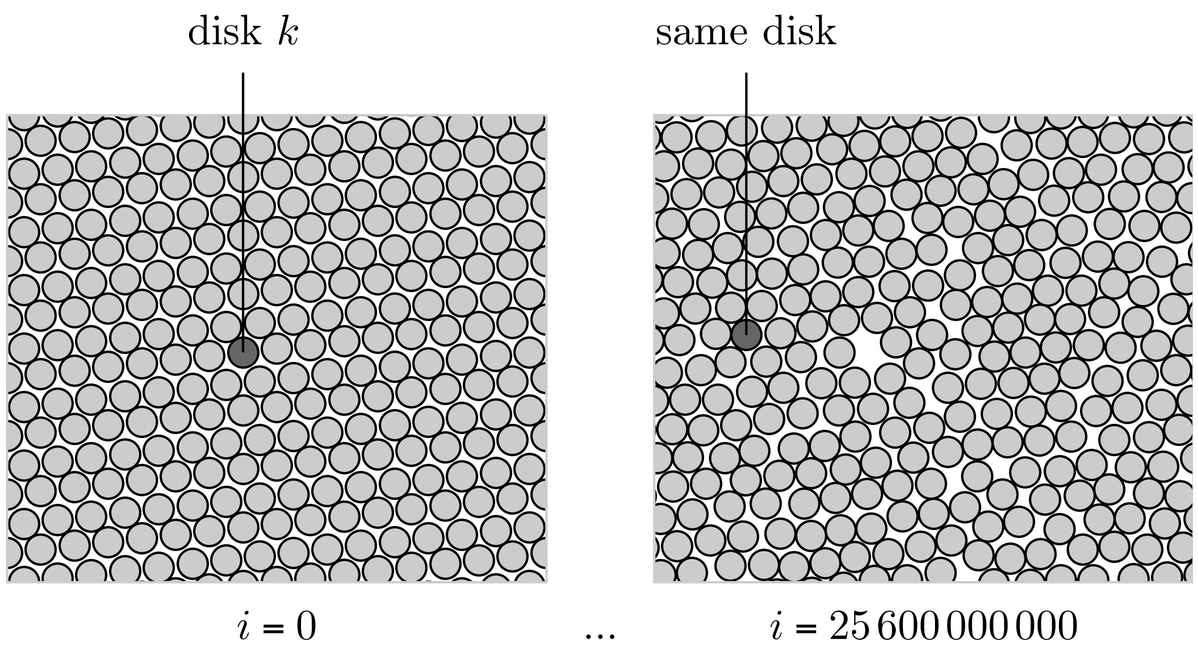
Detailed balance

- All configurations without overlap have equal probality
- Inversion of the update: Inverse displacement vector, same particle

➔ Accept all moves leading to valid configurations

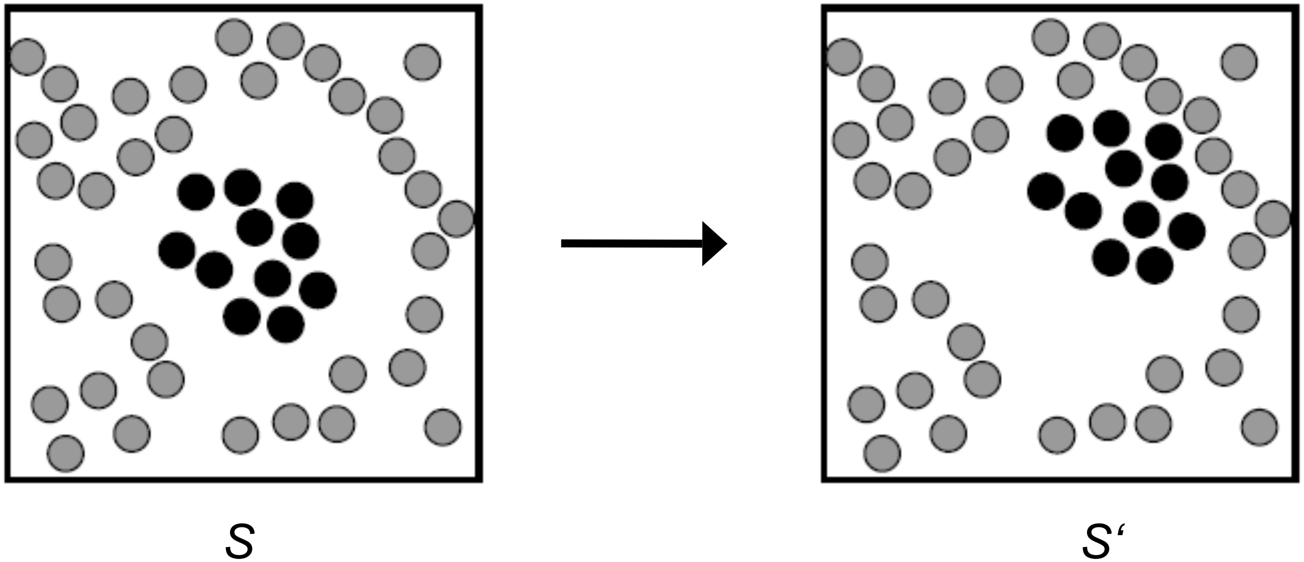
Off-Lattice Models

Problems of the local algorithm:



➔ Local algorithm is frozen at high densities!

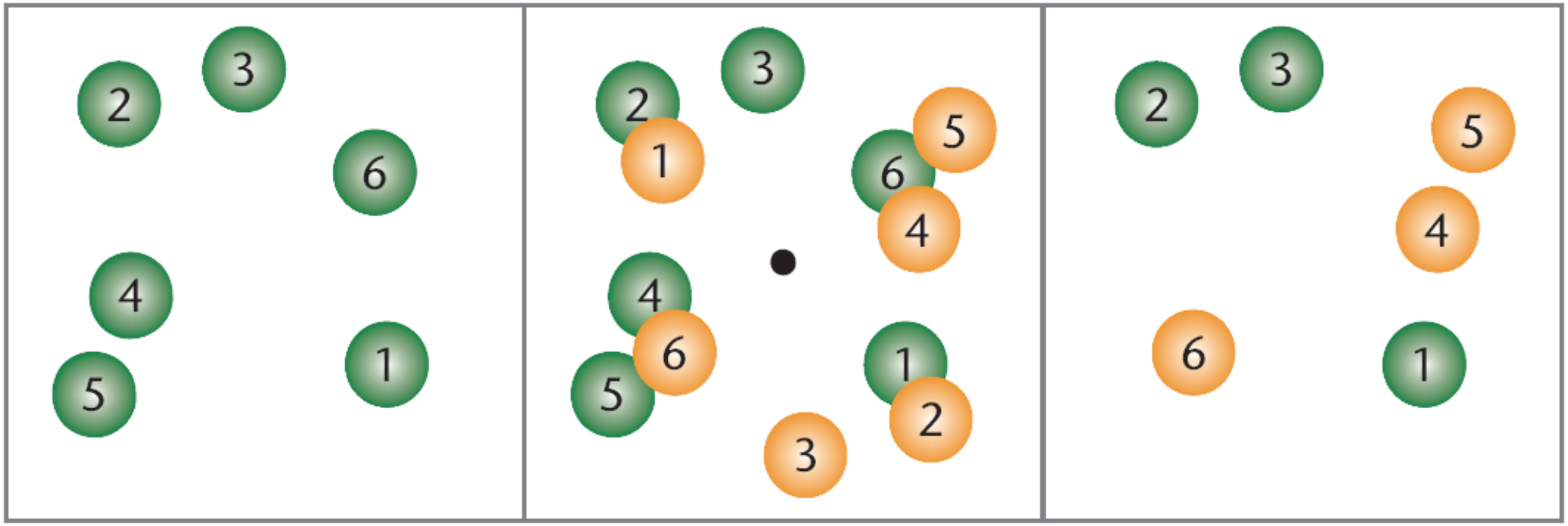
What is a Cluster?



Probability to select cluster in *S* much larger than in *S*‘
➔ Low acceptance-rate

Cluster-Algorithm of Dress and Krauth

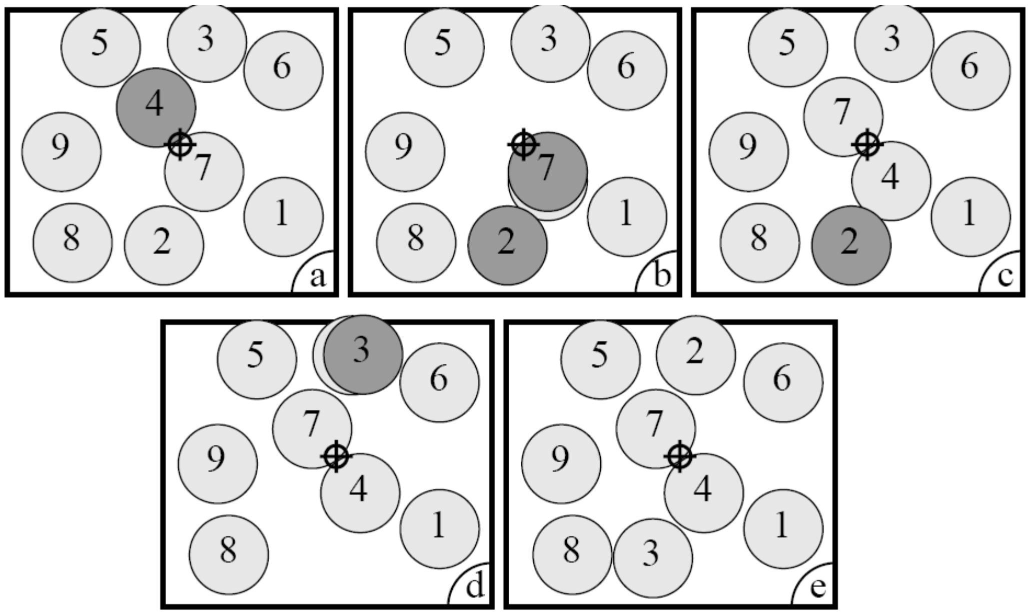
Looking for update with equal probability for cluster selection:



Idea: Exchange Clusters between configuration (S, green) and mirror image (S‘, orange)

Cluster-Algorithm of Dress and Krauth

Realisation:



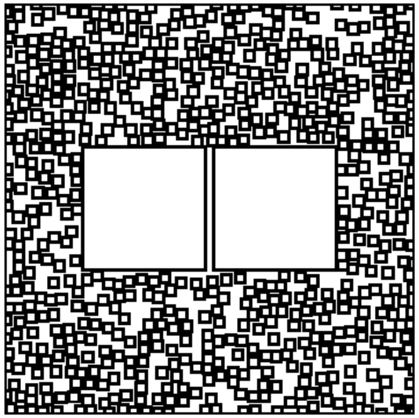
- Select an arbitrary pivot
- Reflect the particle position
- Iterate the procedure until a valid configuration is generated

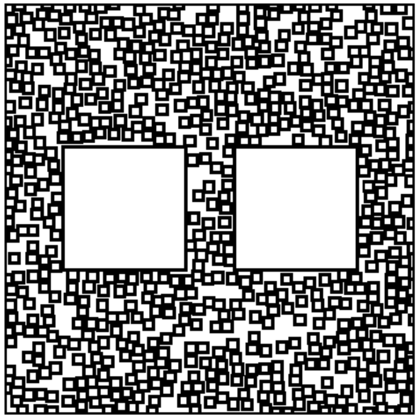
Cluster-Algorithm of Dress and Krauth

Applications of the cluster algorithm

Binary mixtures:

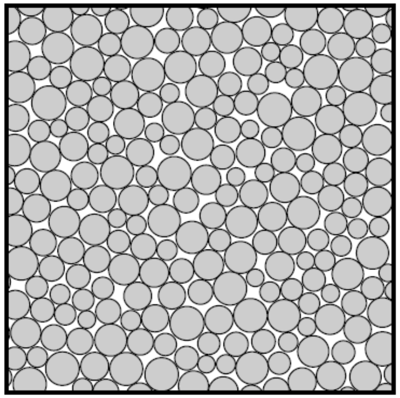
„Depletion Forces“





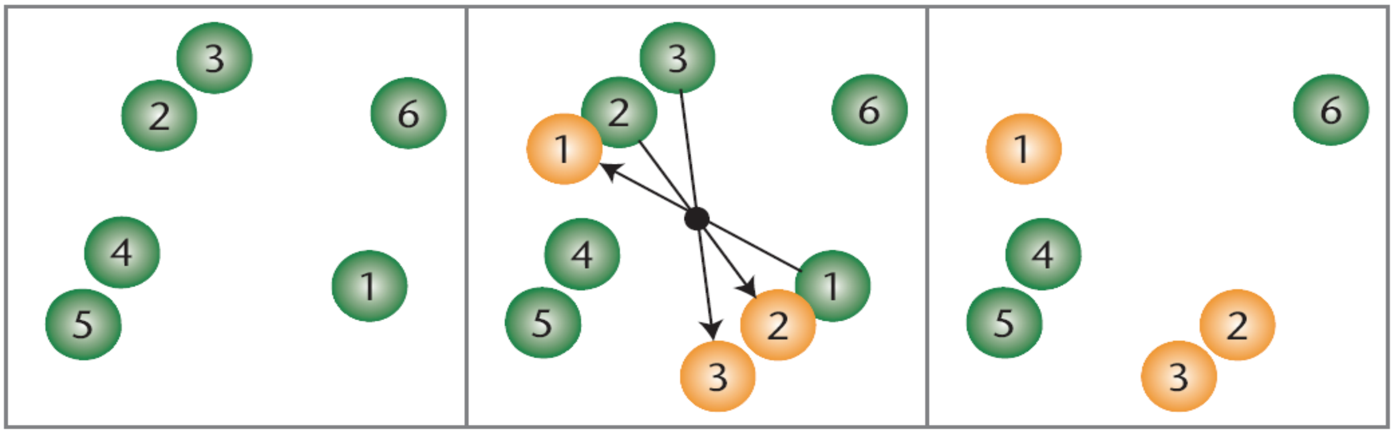
Polydisperse systems:

Equilibration beyond the dynamical transition



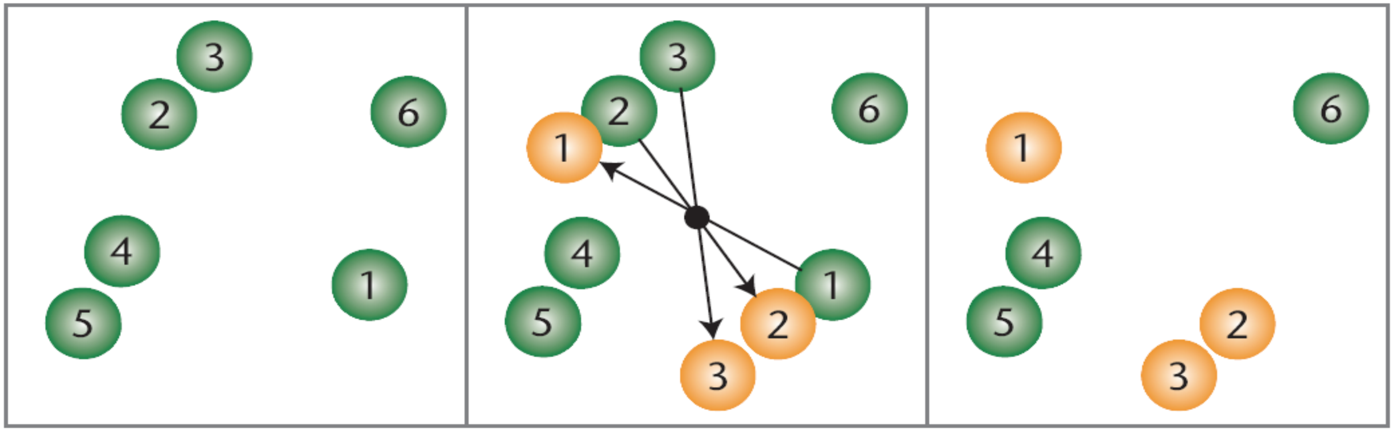
Cluster-Algorithm of Dress and Krauth

Extension: Rejection-free algorithm for interacting systems (Liu & Luijten)



Cluster-Algorithm of Dress and Krauth

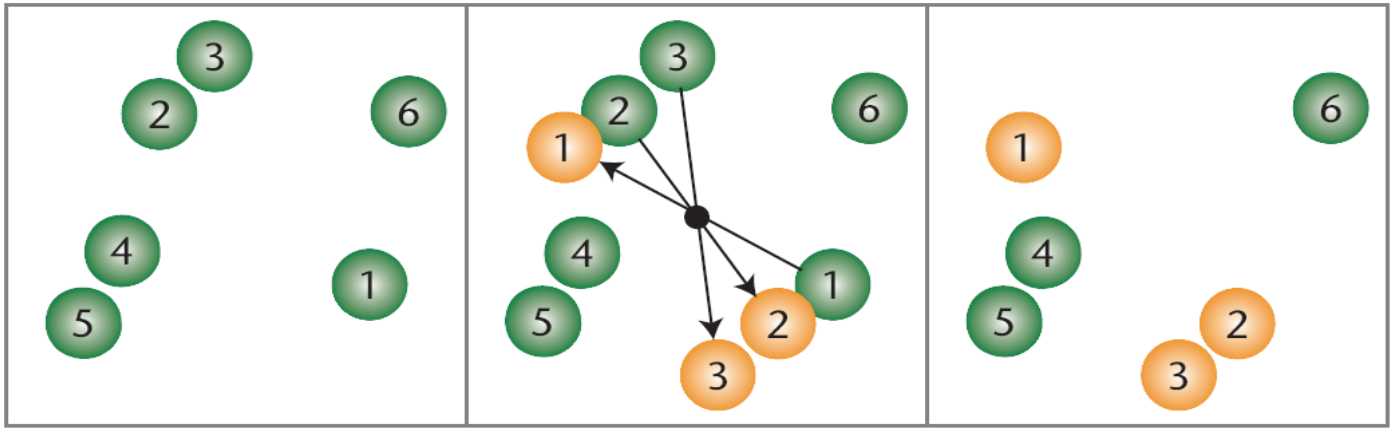
Extension: Rejection-free algorithm for interacting systems (Liu & Luijten)



- Start cluster-construction: Select a particle and a pivot
- Add a new particle with prob. $p_{ij} = \max[1-\exp(-\beta\Delta_{ij}),0]$.
- Add new particles to a list
- Flip particles until the list is empty

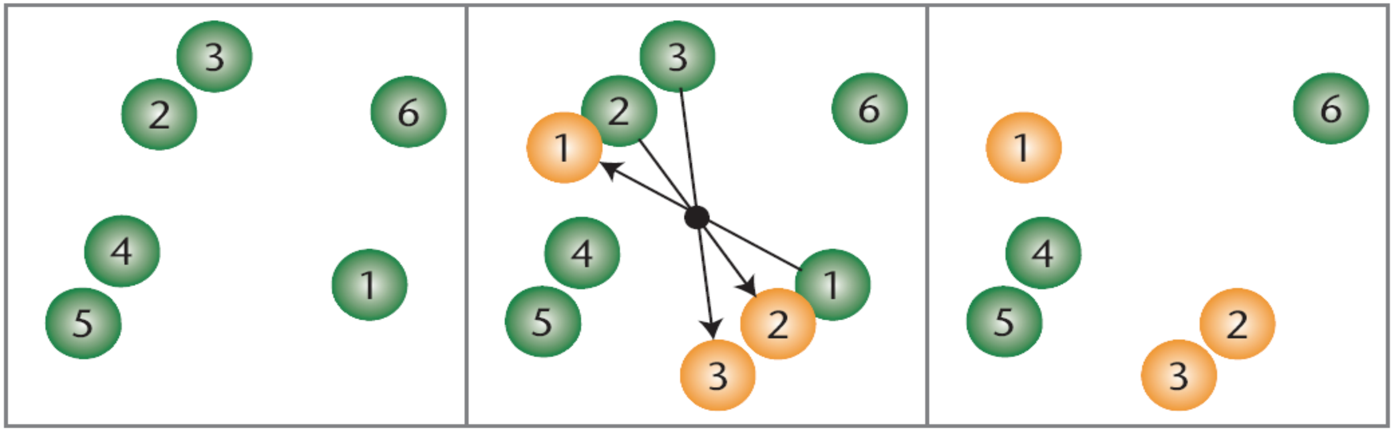
Cluster-Algorithm of Dress and Krauth

Extension: Rejection-free algorithm for interacting systems (Liu & Luijten)



Cluster-Algorithm of Dress and Krauth

Extension: Rejection-free algorithm for interacting systems (Liu & Luijten)



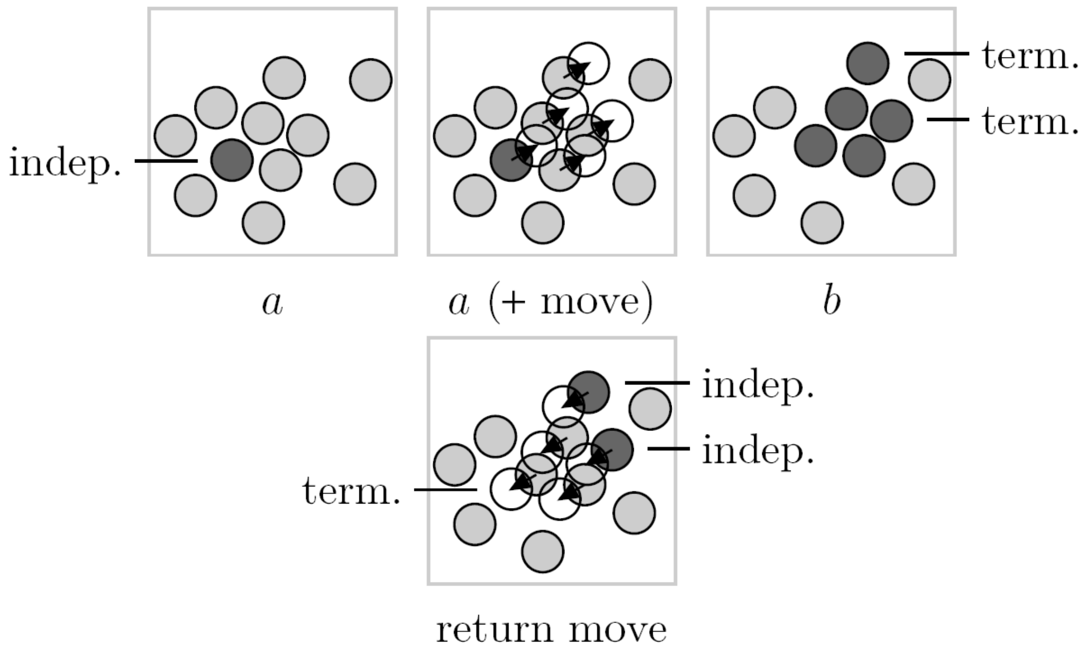
Problem: The algorithm is not efficient at high densities!

Avalanche algorithm

Idea: Generate cluster-moves by iteration of the elementary shift of the particles (avalanches)

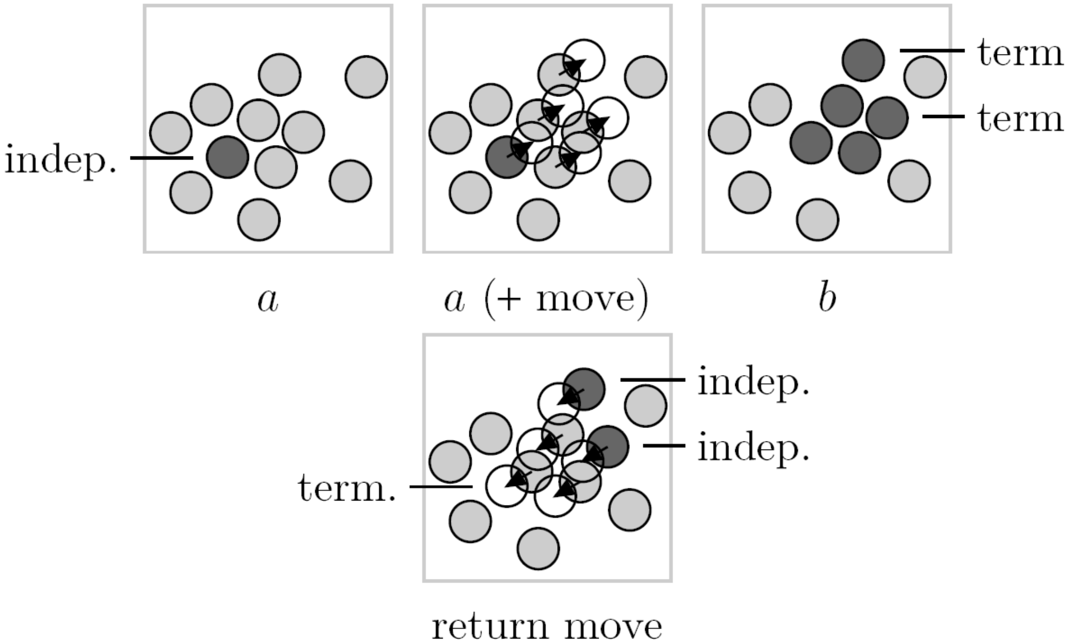
Avalanche algorithm

Idea: Generate cluster-moves by iteration of the elementary shift of the particles (avalanches)



Avalanche algorithm

Idea: Generate cluster-moves by iteration of the elementary shift of the particles (avalanches)



Naive algorithm:

- Random choice of a disk i and a displacement vector δ
- Iterate until a valid configuration is obtained
- Return-move?

Avalanche algorithm

Detailed Balance:

$$\Pi(a)\underbrace{A(a\to b)}_{\text{selection}}\underbrace{P(a\to b)}_{\text{acceptance}} = \Pi(b)\underbrace{A(b\to a)}_{\text{selection}}\underbrace{P(b\to a)}_{\text{acceptance}}$$

Avalanche algorithm

Detailed Balance:

$$\cancel{\Pi(a)}\underbrace{A(a\rightarrow b)}_{\textit{selection}}\underbrace{P(a\rightarrow b)}_{\textit{acceptance}} = \cancel{\Pi(b)}\underbrace{A(b\rightarrow a)}_{\textit{selection}}\underbrace{P(b\rightarrow a)}_{\textit{acceptance}}$$

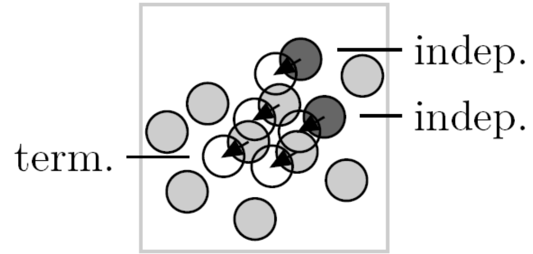
All (allowed) configurations have the same weight

Avalanche algorithm

Detailed Balance:

$$\cancel{\Pi(a)}\,\underbrace{A(a \to b)}_{\textit{selection}}\,\underbrace{P(a \to b)}_{\textit{acceptance}} = \cancel{\Pi(b)}\,\underbrace{A(b \to a)}_{\textit{selection}}\,\underbrace{P(b \to a)}_{\textit{acceptance}}$$

All (allowed) configurations have the same weight



return move

Problem:

Return-move b → a two particles needed to push the cluster (independent set)

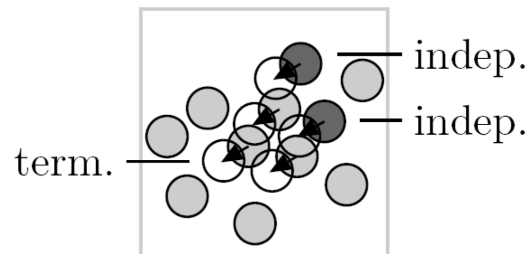
→Rejection of the Update

Avalanche algorithm

Detailed Balance:

$$\cancel{\Pi(a)}\,\underbrace{A(a\rightarrow b)}_{\textit{selection}}\,\underbrace{P(a\rightarrow b)}_{\textit{acceptance}} = \cancel{\Pi(b)}\,\underbrace{A(b\rightarrow a)}_{\textit{selection}}\,\underbrace{P(b\rightarrow a)}_{\textit{acceptance}}$$

All (allowed) configurations have the same weight



return move

Problem:

Return-move b → a two particles needed to push the cluster (independent set)

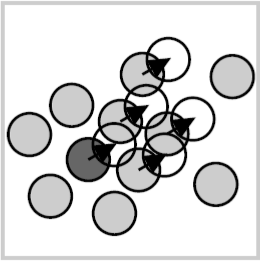
→Rejection of the Update

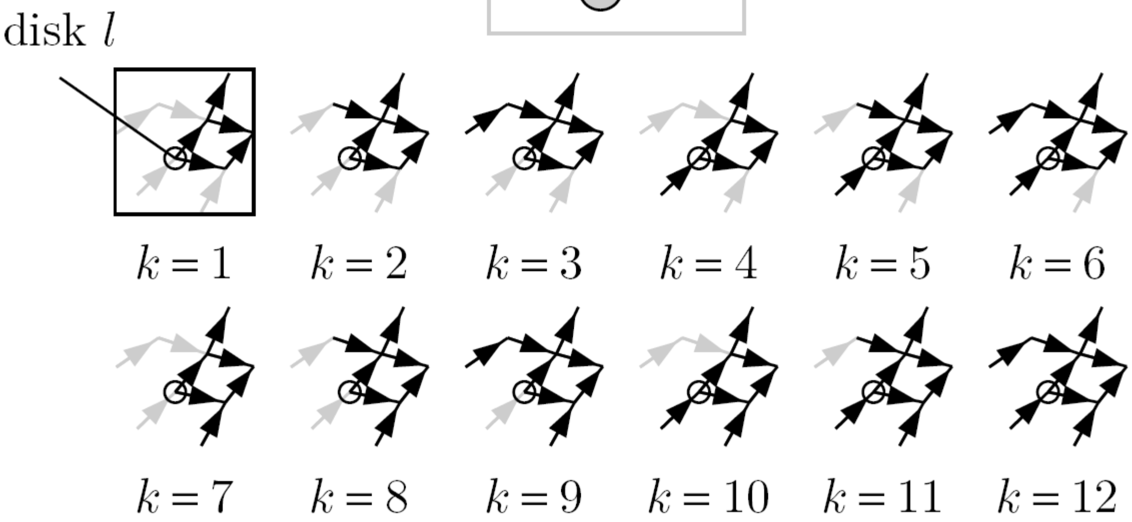
Solution (basic idea):

Generalisation of the cluster-construktion

The avalanche-algorithm

Move: a → b



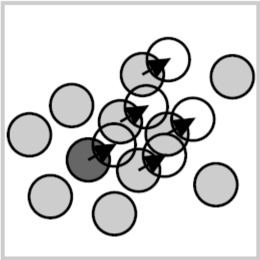


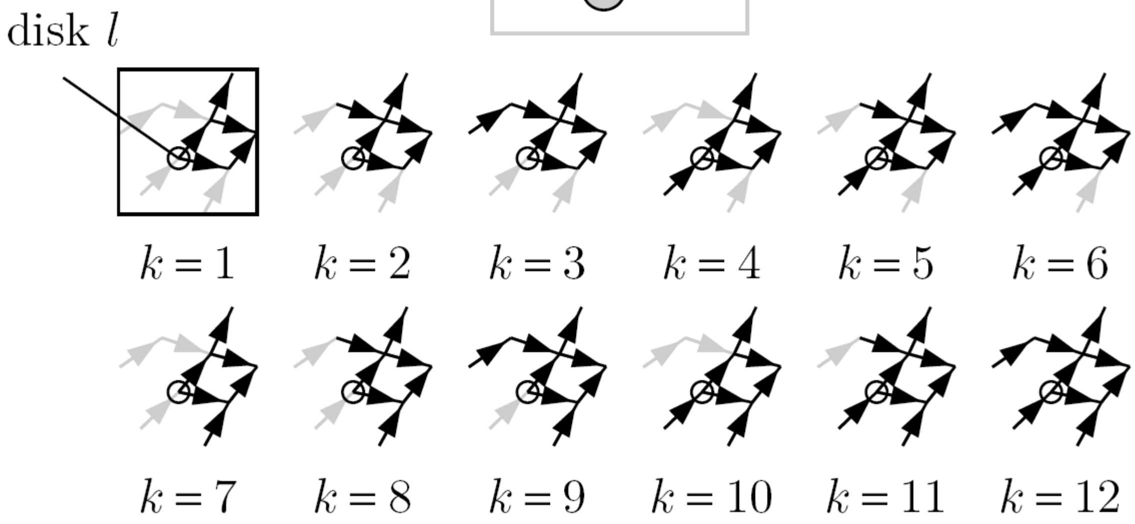
Prob. to select the cluster: A(a → b) = 1/12

The avalanche-algorithm

Recipe:
Construction of a graph, i.e. particles that overlap if shifted by ±δ and include disk *l*

Move: a → b



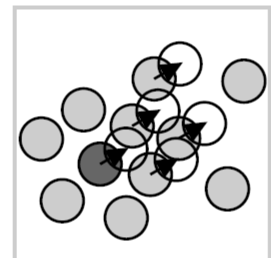


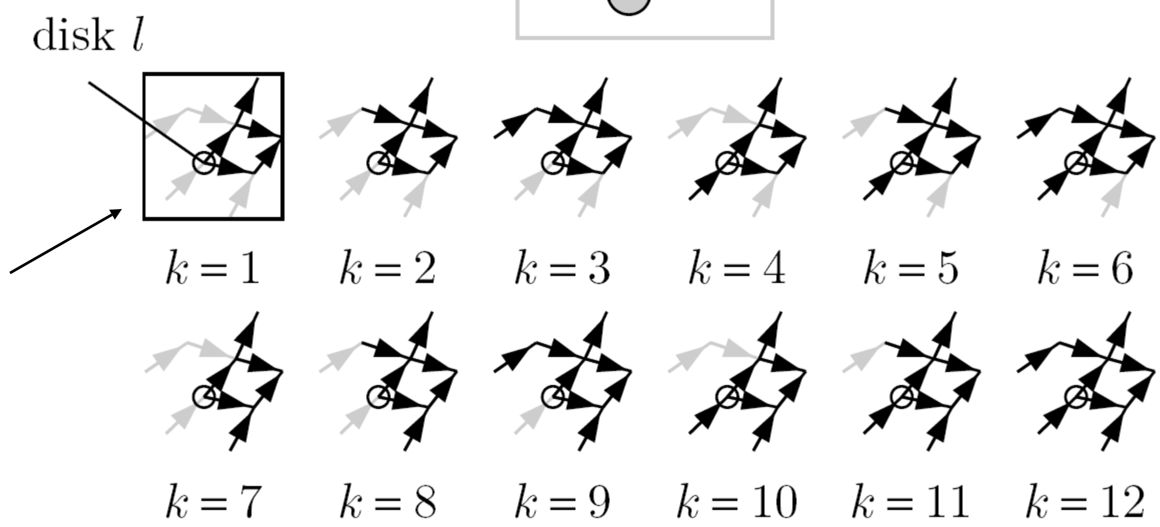
Prob. to select the cluster: A(a → b) = 1/12

The avalanche-algorithm

Recipe:
Construction of a graph, i.e. particles that overlap if shifted by $\pm\delta$ and include disk *l*

Move: a → b





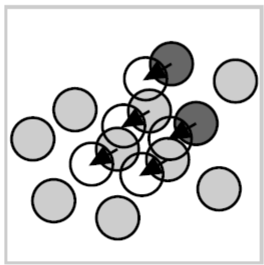
Generation of 12 Clusters is possible

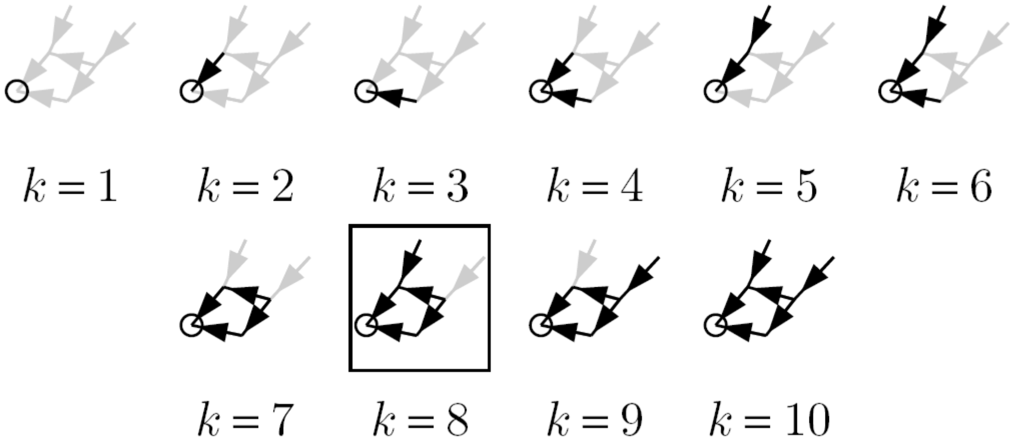
Prob. to select the cluster: $A(a \rightarrow b) = 1/12$

The avalanche-algorithm

Return-Move

b → a





Selection-Prob.: $A(b \rightarrow a) = \frac{1}{10}$

Detailed balance:

$$P(a \rightarrow b) = \min\left(1, \frac{A(b \rightarrow a)}{A(a \rightarrow b)}\right)$$

Here: $P(a \rightarrow b) = 1$

Avalanche algorithm

Elements of the algorithm:

(1) Construction of the graph
(2) Choice of the cluster

Avalanche algorithm

Elements of the algorithm:
(1) Construction of the graph
(2) Choice of the cluster

Principle advantages:
It is possible to select the ampliude
➔ *Tuning of the cluster-size!*
➔ *Efficient at large densities !?*

Avalanche algorithm

Elements of the algorithm:
(1) Construction of the graph
(2) Choice of the cluster

Principle advantages:
It is possible to select the ampliude
➔ *Tuning of the cluster-size!*
➔ *Efficient at large densities !?*

Problem the Algorithm:
Counting clusters is a ***NP-complete problem!***
Naive Algorithm: $\mathbf{2^N}$ ***possibilities*** *(N=# independent particles- im Graph)*

Avalanche algorithm

More efficient implementation:

i. Choose the displacement vector δ
ii. Select a particle with prob. *p* as a pivot particle
iii. Accept the cluster-move with prob. $P_{acc}(i \rightarrow f)$

A priori probability:

$$\mathcal{A}(i \rightarrow f) = p^{N_{ind}} (1-p)^{N_{\overline{C}}}$$

Acceptance probability:

$$P_{acc}(i \rightarrow f) = p^{N_{ind}^f - N_{ind}^i} = p^{\Delta}$$

Generalisation to individual probabilities is possible!

Problem the Algorithm:
High acceptance rate only if Δ is small!

Summary

Summary

- **Rejection-free cluster** algorithms slow at high densities

Summary

- **Rejection-free cluster** algorithms slow at high densities
- **Avalanche algorithm:** Tuning of the cluster-size possible; high rejection rates for large clusters

Summary

- **Rejection-free cluster** algorithms slow at high densities
- **Avalanche algorithm:** Tuning of the cluster-size possible; high rejection rates for large clusters
- Performance not comparable to lattice models

Summary

- **Rejection-free cluster** algorithms slow at high densities
- **Avalanche algorithm:** Tuning of the cluster-size possible; high rejection rates for large clusters
- Performance not comparable to lattice models
- For several systems perform cluster algorithms better than the standard algorithm

Summary

- **Rejection-free cluster** algorithms slow at high densities
- **Avalanche algorithm:** Tuning of the cluster-size possible; high rejection rates for large clusters
- Performance not comparable to lattice models
- For several systems perform cluster algorithms better than the standard algorithm
- Open question: How to choose the selection probabilities for the avalanche algorithm?

Thanks to

- Werner Krauth, LPS, ENS, Paris
- Markus Bellion, now at BaFin
- Heiko Rieger, Saarland University
- DFG for funding

Further reading:

W.Krauth, Statistical Mechanics: Algorithms and Computations, Oxford Univ. Press. (2006)
A. Buhot & W. Krauth, Phys.Rev. E 59, 2939 (1999)
L.Santen & W. Krauth, Nature 405, 550 (2000)
Liu and Luijten, Phys. Rev. Lett.. 92, 035504 (2004)
M.Bellion, W.Krauth, H.Rieger, L.Santen, in prep.