A parallel cluster algorithm for Monte Carlo simulations applied to model DNA systems

Jakob Schluttig

Institute for Theoretical Physics, University of Leipzig

eMail: jakob.schluttig@itp.uni-leipzig.de

Abstract:

Cluster methods are used in Monte Carlo simulations to decrease the autocorrelation time, i.e. the interval between statistically independent configurations, which becomes crucial close to critical points and phase transitions. The aim of this work is to build the basis for a Monte Carlo cluster algorithm for continuous two dimensional spin systems. First the Kornyshev-Leikin model potential is introduced which is applied to Monte Carlo simulations of DNA systems. Afterwards a purely geometrical technique for searching clusters is described. Furthermore it is extended to an energetic cluster criterion, which is the basis in Monte Carlo cluster methods. The scaling of the implementation is measured and analyzed. Finally it is used to study geometric clusters as a function of different DNA characteristics, e.g. the charge compensation parameter θ.

Kornyshev-Leikin pair potential for rigid helical molecules

It is well known that DNA forms close packed aggregates of various structures, e.g. in human chromosomes or viruses. Experimentally it was observed that short fragments form columnar aggregates which are suitable to study interactions, e.g. like charge attractions between molecules and global structures.

At first glance a whole DNA is far too complex to describe its interaction with other molecules in a closed analytical framework. However, A. A. Kornyshev and S. Leikin [1] described DNA molecules as long cylinders, carrying helical, continuous line charges on their surface, taking advantage of the symmetries in helical molecules. Thus it was possible to derive an exact formalism which can be used to calculate interactions between two stranded helical molecules like the DNA. The theory in [1] is formulated in a rather general way, so the potential that is finally used for the simulation had to be derived and adapted to the actual application. The whole interaction energy is obtained by a sum of three different terms. The first one, labeled as w_{cyl}, corresponds to the interaction between two homogeneously charged cylinders. w_{self} is a "self correlation" energy, which is due to correlated discrete surface charge distributions on each molecule. Ultimately w_{cross} is a "cross correlation" energy, which is caused by nonrandom alignment of discrete charges on the opposing molecules. The following formulae describe an energy density, where the energy is normalized to the persistence length L_p [2].

$$u(\phi, r) = \frac{(4\pi\sigma_0)^2}{\epsilon}\left[w_{\text{cyl}}(r) + w_{\text{self}}(r) + w_{\text{cross}}(r,\phi)\right] \tag{1}$$

$$w_{\text{cyl}}(r) = \frac{(1-\theta)^2}{2\kappa^2}\frac{K_0(\kappa r)}{[K_1(\kappa a)]^2} \tag{2}$$

$$w_{\text{self}}(r) = 4\sum_{n=1}^{\infty}\cos^2(n\tilde{\phi}_s)\frac{\sum\limits_{j=-\infty}^{\infty}\frac{I_{j-1}(\kappa_n a)+I_{j+1}(\kappa_n a)}{K_{j-1}(\kappa_n a)+K_{j+1}(\kappa_n a)}K^2_{n-j}(\kappa_n r)}{\left[K_{n-1}(\kappa_n a)+K_{n+1}(\kappa_n a)\right]^2\kappa_n^2} \tag{3}$$

$$w_{\text{cross}}(r,\phi) = 4\sum_{n=1}^{\infty}(-1)^n\cos(n\phi)\cos^2(n\tilde{\phi}_s)\frac{K_0(\kappa_n r)}{\left[K_{n-1}(\kappa_n a)+K_{n+1}(\kappa_n a)\right]^2\kappa_n^2} \tag{4}$$

$$\kappa_n = \sqrt{\kappa^2 + \left(\frac{2\pi n}{H}\right)^2} \tag{5}$$

Due to the very rapid convergence, the sums in eqs. 3 and 4 may be truncated at $n = 5$ and $-5 \leq j \leq 5$.

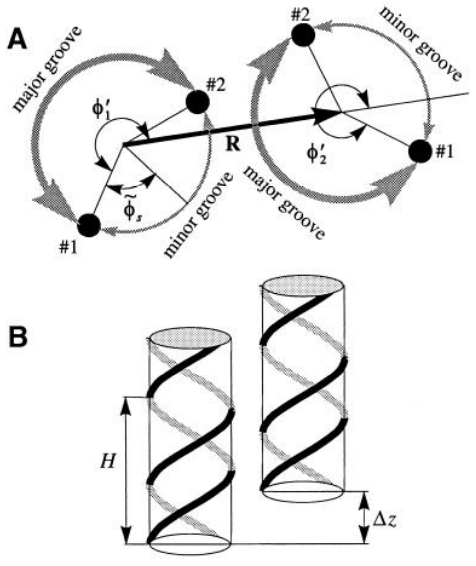


Fig. 1: Simple scheme of important structural values for two interacting DNA double strands.

The parameters appearing in eqs. 1-5 are briefly explained [5]:
First there are the structural parameters of the phosphate pattern (see fig. 1, taken from [4]): the helical pitch H, the azimuthal half-width of the minor groove $\tilde{\phi}_s$ and the hard-core radius a. Furthermore each DNA duplex carries the negative charge of phosphates with surface charge density σ_0 plus a compensating positive charge arising from adsorbed counter ions. The degree of compensation is described by the parameter θ, where $0 \leq \theta \leq 1$. eqn. 2 the term w_{cyl} vanishes if $\theta = 1$. The mobile counter ions in solution cause an exponential decay of the Coulomb interaction of the two helices for large separations. This exponential decay is parameterized by the inverse Debye screening length κ. The solution is also considered by its dielectric constant ϵ. Actually the dielectric constant and the Debye screening length are both temperature dependent and κ is also a function of ϵ. Although this is not taken into account here [3].

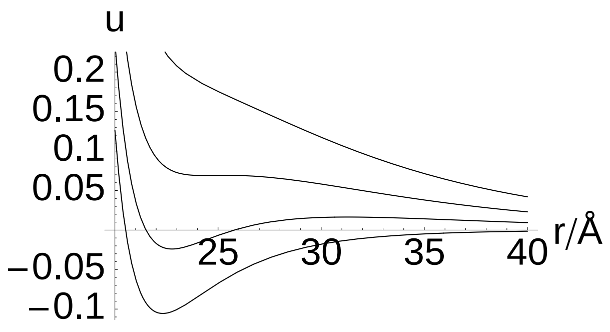
The simulations were carried out considering B-DNA structure. The proper parameters were taken from [4] and are collected in table 1.

L [Å]	a [Å]	H [Å]	$\tilde{\phi}_s/\pi$ [rad]	σ_0 [μ C/cm^2]	ϵ
500.0	9.0	33.8	0.4	16.8	80

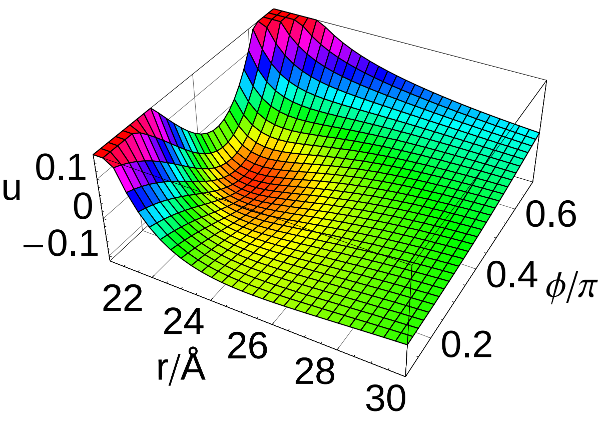
Table 1: Structural and chemical parameters for the DNA-B molecules.

From eqs. 2 to 4 it can be seen that there is a two dimensional potential energy landscape for a pair of DNA molecules depending on the distance of two strands $r = |\mathbf{R}|$ and the relative azimuthal orientation ϕ. The latter can be simply calculated as the difference of the respective angles ϕ'_1 and ϕ'_2 of the $5' \rightarrow 3'$ strand, relative to a reference direction. A shift in the axial direction Δz therefore translates into a different azimuthal orientation.

Due to a lack of time it was not possible to study the DNA aggregates with respect to different screening lengths κ^{-1}. For all discussions and measurements in this report it is fixed to $\kappa = 0.1\text{Å}^{-1}$. The following pictures should help to get an imagination of the potential energy and support the understanding of the expected effects.



a



b

Fig. 2: **a**: Kornyshev-Leikin potential at $\phi = 0.4\pi$ for $\theta = \{1.0, 0.9, 0.85, 0.8\}$, where $\theta = 1.0$ is the lowest plot and $\theta = 0.8$ the uppermost. **b**: Calculated with $\theta = 1.0$. The values of the potential energy are expressed in artificial units.

Fig. 2.**a** shows four qualitative different types of the interaction potential obtained for different values of θ. For high compensation θ the interaction of the two DNA strands is purely attractive for a nearly perpendicular azimuthal alignment until a certain equilibrium distance of about 22Å is reached. That is very short considering the hard core distance of 2×9Å$=18$Å. As the number of adsorbed counter ions decreases a local maximum arises which separates a condensed and a crystalline state, while the local minimum is still below zero and the energy barrier gets lower. For even lower values of θ the local minimum at short distances exceeds zero and thus the bound (clustered) state is obviously not any longer more preferable than the crystal state. Finally there is a certain - probably critical - point of charge compensation where the local minimum vanishes and the potential gets completely repulsive.

Keeping in mind that the potential energy landscape is not 1- but 2- dimensional (see fig. 2.**b**) it becomes clear that even in the case of $\theta = 1$ it is not sure wether all the DNA strands will go into the equilibrium distance and form one big cluster in the ground state. The preferred orientation between two molecules at short distances is $\approx \pi/2$ (cmp. fig. 2.**b**). A clustering of three molecules will result in an energetically frustrated configuration and therefore the formation of big clusters is not obvious. Also it can be seen from fig. 2.**b** that for large distances of two interacting strands the parallel orientation is favorable. Interestingly even if the system would be simulated on a lattice the potential would therefore still be density dependent. This makes it clear that there is a rich phase behavior to be expected. For $\theta = 0.7$, i.e. in the repulsive regime, this was studied for the ground state in ref. [5].

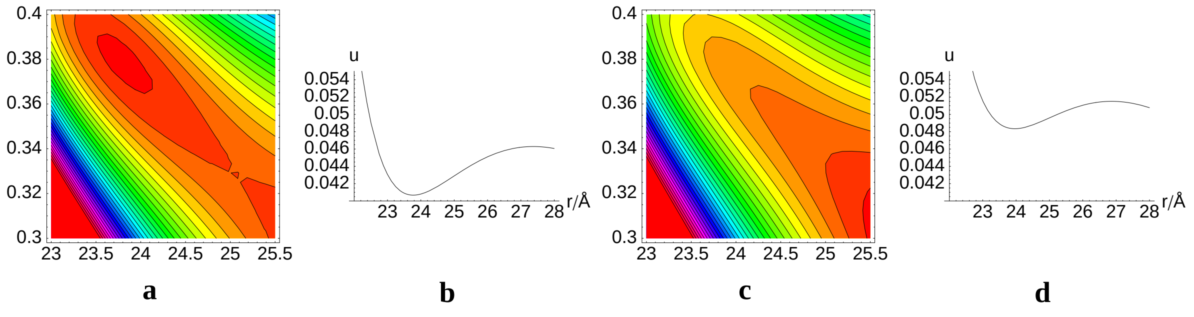


Fig. 3: These pictures illustrate the critical behavior of the Kornyshev-Leikin potential at $\kappa = 0.1$ and $\theta \approx 0.86$. Also it is shown that it is important to look for a local minimum with respect to r *and* ϕ. **a**, **b**: $\theta = 0.864$. **c**, **d**: $\theta = 0.860$. **b** and **d** are both obtained for $\phi = 0.38$.

Another important fact that results from the dependency of the orientation ϕ is visualized in fig. 3. The two plots **b** and **d** show the potential for fixed mutual orientations. They seem to be qualitatively similar. However, this is, which is proofen by **a** and **c** where it is obvious that for $\theta = 0.860$ there *is no* potential barrier as seen in fig. 3.**d** since the two strands can always loose energy by orientating parallel and seperate from each other at the same time. Whereas for $\theta = 0.864$ a real local minimum exists. This demonstrates that the critical value for θ must be in the range of $\theta_c \in [0.86, 0.864]$ and not at 0.85 as it could be expected from fig. 2.**a**.

Simulations

All simulations were carried out with the program *“SpinCG^{2d}”* by G. Sutmann [6]. As it is seen from eqs. 1-4, the only variables describing the interaction between two DNA molecules are their interaxial distance and their mutual orientation. This picture corresponds to a 2-dimensional spin system, where spins have three degrees of freedom (position, orientation), i.e. a kind of generalized X-Y-Model.

The starting configuration of DNA strands consits of a hexagonal structure with lattice constant d. This distance is related to the DNA density. Afterwards a mixture of down-hill and simulated annealing algorithm is performed, i.e. the temperature is decreased by a certain amount after *each* Monte Carlo step. In so doing the system is cooled down from $T(t_{\text{MC}} = 1) = 1000\text{K}$ to $T(t_{\text{MC}} = N_{\text{MC}}) = 30\text{K}$. N_{MC} is the number of Monte Carlo steps performed in a whole simulation and t_{MC} is the Monte Carlo time. This means that not only the system is given no time to equilibrate but in the whole simulation the forming of an equilibrium state is actually prevented. On the other hand as we are mostly interested in energetically favorable states the cooling to very low temperatures will definitely lead to ground state like structures of the aggregate. These are naturally somehow artificial since the DNA would probably change its configuration dramatically at such low temperatures. It can be assumed that for long simulation times N_{MC} the influence of the non equilibrating kind of the simulation can be neglected. In [2] the temperature was kept constant over a certain number of Monte Carlo steps and similar results were observed. During the simulation the size of the trial moves is adapted to have an acceptance rate of 0.5. Since the explicit evaluation of the potential is rather expensive it is interpolated during the simulation by a second order interpolation from a table. For the cross correlation energy the first 5 terms of the sum over n are interpolated and multiplied by the orientation dependent $\cos(n\phi)$.

Geometrical cluster search

Thinking of an efficient way to identify neighbors in a certain distance, which still has to be defined, it is most important to consider the continuous properties of the particles in the system on the one hand and the spatial decomposition of *"SpinCG^{2d}"* on the other. The latter is mostly significant for the parallelization. The whole program is written using MPI for distributed memory systems without memory replication, so the cluster search also has to deal with this distributed data handling.

The basic approach consists of:

1. identifying clusters sequentially and independently on each processor
2. communicate with other processors to link global clusters
3. scatter the whole linking information

First of all it is necessary to introduce a geometric cluster criterion. If the distance of two molecules exceeds a treshold length r_0 they are not considered as *neighbors*. Since the structures of aggregate configurations are extremely varying, it seems to be a good choice to correlate this criterion with the Kornyshev-Leikin potential and thus introduce a barrier dependent treshold length r_0. By some test runs it turned out that postulating r_0 as the distance where the potential barrier is overcome by $4/5$ provided acceptable results (see fig. 4). Because of the form of the potential it was always quite obvious which particles were in the short distance of the potential minimum, and which were in a kind of crystalline state with respect to each other. That is the clusters were clearly separated and the identification could be verified easily.

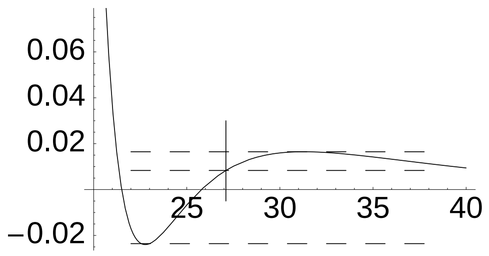


Fig. 4: The maximum distance of two particles belonging to one cluster is derived from the interaction potential.

Sequential local cluster identification

There is a variety of algorithms for searching clusters on a lattice. One of the most well known is probably the Hoshen-Kopelman algorithm [7]. A lattice system is particularly simple in that way, that firstly the particles (i.e. the lattice sites) are in a given order which makes them easy to address. Furthermore there is a definite number of neighbors at given positions. Both of these lattice properties are not given in the considered system. Since there is no way to order the particles systematically, it would be necessary to check every particle against every other particle which would result in a complexity of $\mathcal{O}(N^2)$. Even if the particles would be sorted by the x- or y-coordinate, the quadratic scaling behavior would probably be only decreased by a factor. This is of course undesirable. Thus the idea is to alter a lattice based algorithm to fit the needs without loosing its benefits.

Virtual lattice structure

This is accomplished by introducing a virtual lattice and sort the molecules into the lattice cells by a linked list (see fig. 5). In so doing a small number of particles belonging to one cell can be addressed nearly as fast as if they actually were particles in a lattice system. Furthermore this method makes it possible to check only pairs of molecules in a number of potential neighbor cells. But the structure of the lattice still needs some more investigation.

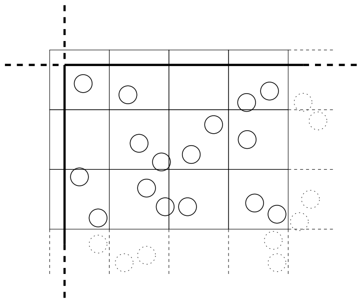


Fig. 5: Schematic view of the sorting of particles into a virtual lattice at the border of a processor.

It is plausible that the preferable structure is a square lattice. It is easy to address and the particles can be assigned efficiently to the cells. How to choose the lattice constant l? A very small l means that in one lattice cell there will be only few molecules and hence the step of checking each particle in one cell against any particle in another one will not be expensive although it has a complexity of $\mathcal{O}(N_c^2)$, where N_c is the number of particles in a cell. On the other hand for large l the number of cells in the whole lattice *and* also the number of possible neighbor cells decreases. The latter two factors are both scaling with $1/l^2$. Whereas the number of particles N_c is scaling with l^2. So the overall scaling with respect to l will be $1/l^2 \times 1/l^2 \times (l^2)^2 = 1$. From that it is not obvious why the introduction of a lattice should improve the performance.

It has to be noted that every particle has a certain size (hard core radius a in the viewed application) which gives an upper bound for the density of the system, i.e. there is a certain l where on average only one or two particles are located in each cell. Decreasing l even further would result in many empty lattice cells which still have to be checked as potential neighbor cells. On the other hand for very small l there are cells whose particles are always within the radius r_0 which avoids an explicit check of particle pairs within cells. But since for the specific example r_0 is not much larger than twice the hard core radius of one of the molecules it would surely not make sense to decrease l to such low values. So there is a lower bound of l. Also a minimum of 8 neighbor cells exists which always have to be checked because they have bordering corners or edges and could therefore contain molecules bound to molecules from the currently considered cell. So it is also apparent that it is no use in increasing l to very large numbers because the number of neighbor cells does not reduce anymore after reaching 8.

An important fact is that for a lattice constant smaller or equal to $r_0/\sqrt{2}$ it is *sure* that particles belonging to one cell are in the same geometric cluster (see fig. 6). Otherwise this has also to be checked which results in another N_c^2 step for every lattice cell! Taking all these actualities into account it seems to be a good choice to set $l = r_0/\sqrt{2}$. Still it is possible that for certain parameters like extremely low or high densities a change of the virtual lattice constant l could result in some speedup. From fig. 6 it can be seen that for this particular l there are 20 cells possibly containing neighbors. For checking each pair of cells only once it is enough to check 10 neighbors from each cell. There is no cell that definitely can only contain molecules in the neighboring distance, so every cell has to be checked.

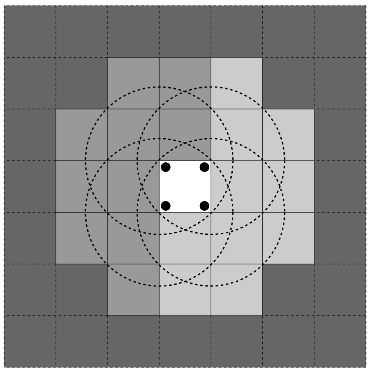


Fig. 6: Study of the virtual square lattice with lattice constant $l = r_0/\sqrt{2}$.

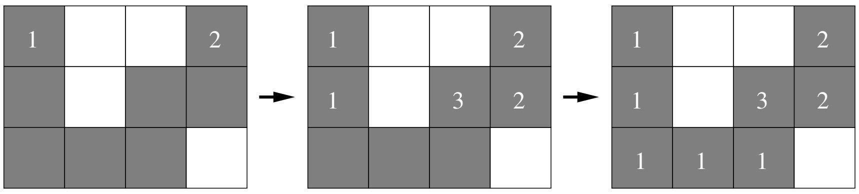
For the later communication with other processors it is mandatory that the lattice coordinates are somehow global. Otherwise the linking would get complicated. This is achieved by sorting the particles into a global grid structure, where single grid cells may belong to different processors. Technically this is realized by truncating the decimal places of the two dimensional floating point coordinates of each molecule, which are global, divided by the lattice constant $l = r_0/\sqrt{2}$. In Fortran code it would look like this:

```
lattice_coordinate=FLOOR(real_coordinate/lattice_constant)
```

This also explains the gap between the edge of the processor and the edge of the lattice in fig. 5.

Adapted Hoshen-Kopelman algorithm

The Hoshen-Kopelman algorithm works iterative by using a linked list. Fig. 7 is a schematic of the underlying idea. If two clusters are linked, the list entry of the one with the higher *proper* cluster label becomes a pointer to the smaller cluster label, which is represented by a negative integer number. The list entry of the other one contains a positive integer which is the total number of particles in the certain cluster, including all clusters which are linked with this one. Now the *proper* cluster label has to be find in the list by following these pointers until an entry equal or greater than 0 is reached.



list entry no.	1	2	3	1	2	3	1	2	3
value	1	1	0	2	3	−2	8	−1	−2

Fig. 7: Visualization of the Hoshen-Kopelman cluster search for next neighbors in a small square lattice.

The algorithm is capable of handling any number of potential next neighbor cells. Now some pseudo code will show how the actual sequential local cluster search is done.

```
DO FOR all lattice cells i
  DO FOR half of the possible neighbor cells j
    search for proper cluster label of j
    IF this label is not yet marked as linked with i
      IF there is a pair of molecules from i and j with distance<r_0
        mark label of j as linked with i -> j_l
      END IF
    END IF
  END DO
  IF number of linked clusters is
    0: i gets new cluster label
  >=1: find smallest proper label j_s from {j_l}
       link all other linked neighbors j_l and i to j_s in the list
       save sum of all cluster sizes as new size of j_s
  END IF
END DO
```

For the later parallelization it is important that the newly introduced cluster labels are globally unique. This is no problem since every processor has a non-ambiguous number for identification and there is an upper bound of particles that can be on one processor. So a new cluster label will be calculated like below:

```
cluster_label=local_counter+local_cpu_id*max_particles_per_cpu .
```

To provide the ability of linking local clusters to global ones the linked list array, which will be called `cluster_id` in the following, should have a dimension of

```
number_of_processors*max_particles_per_cpu
```

on every processor. This makes sure that also the linking information is somehow global from the beginning and can therefore be easily exchanged.

Parallel global cluster identification

In *"SpinCG^{2d}"* the whole two dimensional system is divided into N_{PE} domains, where N_{PE} is the number of processors on which the application runs. These domains are not stripes, which would mean that the effort of communication for the Monte Carlo simulation is not decreasing by higher amount of processors, but the program tries to make the domains as close to a square as possible. Hereby through adding more processors the edge length of each processor is reduced which is the driving factor for the communication.

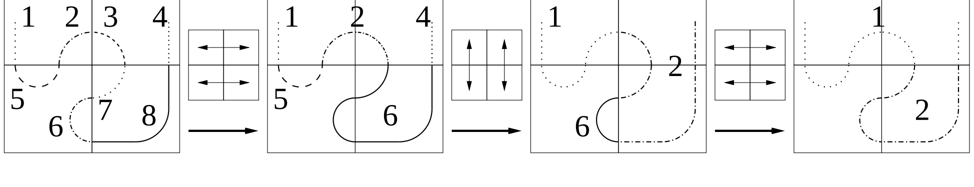


Fig. 8: With only next neighbor communication this example would need 5 communication steps until all of the local clusters are connected to one global one.

Now it has to be thought of a good strategy for identifying the global clusters which means clusters that span the domains of more than one processor. Fig. 8 depicts that it is not possible, or at least not efficient, to do this only by communication between next neighbor processors. The specific example shows that one could probably construct an artificial global cluster to get the necessity of any “desired” number of communication steps until the local clusters would be linked correctly. From that it is obvious that there has to be some kind of *all-to-all* communication.

To realize this with an acceptable scaling the idea is to use a communication tree. While descending to the root no information from the domain borders may be lost. This is achieved by introducing virtual domains as shown in fig. 9. One of two communicating processors is always the *master* which receives the whole border information from the *slave*. Afterwards it processes the bordering edge to link the clusters. This linking is done exactly like the identification of local clusters before. Thereafter it uses the rest of the received information to build up the virtual border of the domain containing the whole area of the two processors before. Since the bordering edge information is already translated into cluster linking pointers and it is obviously not part of the edge of the new virtual domain, it does not have to be sent in the next step of communication and can be rejected. Now the *master* is capable of communicating with other *masters* of the same level in the tree which will have similar dimensions relating to their virtual domains. Thus the number of communication steps is proportional to the logarithm of the number of processors in a certain direction.

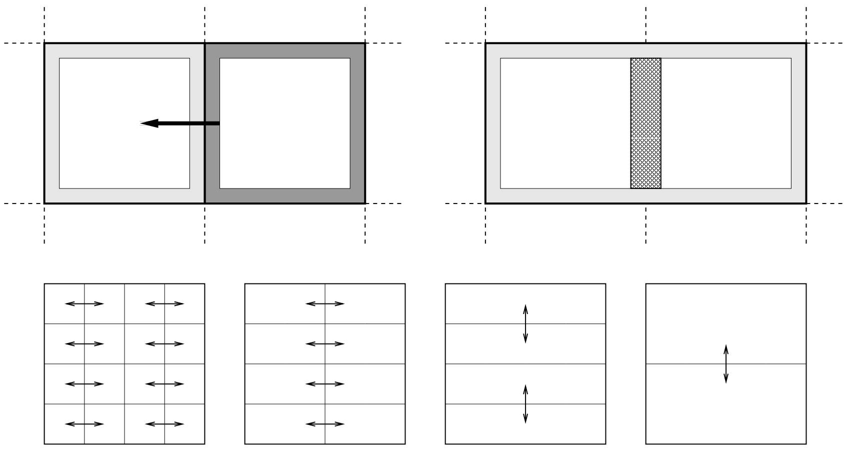


Fig. 9: During the communication process the only important thing is the edge of the domain of a processor. These edges grow and become the borders of virtual domains which include the real domains of many processors.

Since communication is very expensive it is important to reduce the information that has to be exchanged to a minimum. First of all there is no way to avoid sending the poisitions of the particles near the border of the processor domains. The appropriate array will be called `rxyz_border` in the following. Also it is necessary to have the correct cluster labels which are assigned to the bordering lattice cells. And finally the global linking cannot be done without partly knowing the entries of `cluster_id`, i.e. all entries dealing with cluster labels which exist on or which are pointed at from the lattice border. Therefore an array `cluster_id_border` is introduced, which dimension is two times the number of linked list entries as described before, one for the address and one with the actual value of each entry. To not loose any information during the communication especially concerning the finally broadcasted feedback which tells every processor the global labels of the clusters which are contained by it, it is necessary to always keep this information up-to-date with the global linking and *never* reject any of this data, even if it lies on a border inside a virtual domain. This makes up a difference to the other two communication arrays. To reduce the amount of distributed data it is useful to link the clusters lying on the edge of each processor directly to the proper labels before starting the communication. Therefore only proper labels and the labels that they point to, which is important for their size, have to be send. Fortunately it is always possible to calculate the dimension of the lattice on the edge of a virtual domain. That is the cluster labels in the lattice can be sent first because there is no uncertainty about the length of the array that has to be received. At the end of the array the size of the other two arrays can be stored, so absolutely no unnecessary information is exchanged.

The amount of communication is dependent on the overall edge length of virtual domains during the whole global cluster linking process. For rectangular areas the optimal form with respect to the edge length would be of course the square. So at first glance it should be faster to build up quadratic virtual domains instead of firstly linking global lines as it is done in the scheme in fig. 9. But after the global lines are linked, the vertical edges do not contain any relevant information for linking anymore. It is important that for a system with periodic boundary conditions this would be the point of time to apply periodic boundary conditions in horizontal direction. A short calculation example shall confirm the speed up coming from neglecting the vertical edges for further communication. Considering a number of 16 processors and fully quadratic domains with an edge length a. A communication tree which uses quadratic virtual domains if possible would have a whole edge length of $4a + 6a + 8a + 12a = 30a$ to be send while communicating. Using the approach shown in fig. 9 and rejecting the vertical edge information for the last two communication steps it is only $4a + 6a + 8a + 8a = 26a$. As looked at before the size of the linked list table is independent from the chosen type of communication since it is never shrinked.

The hexagonal setup of the system in *"SpinCG^{2d}"* leads to a system that expands more in the y-direction. For that reason it should even be actually faster to first build up global lines rather than global columns. Since afterwards the amount of communication is only dependent on the length of the lines, which is smaller than the length of columns would be, and does not increase anymore. Also the number of processors in the y-direction is greater than the decomposition in x-direction, that is why the rejection of the vertical borders can be done one communication step earlier under certain circumstances. Finally if we consider the architechture of a parallel computer, e.g. the *ZAMPaNo* [11], it is likely that neighboring processors in x-direction have shared memory, which makes the first one or two steps of communication fast in the case of first building global lines. But it would need some thorough investigations to proof this.

After all this there will be a *global master*. In its local array `cluster_id` it contains the whole global linking information. If it is enough for every processor to know only the size of clusters whose global labels originate from it, the *global master* must only send back the updated entries of the linked list array which originally came from the certain processor. If in contrast it is important that every processor knows the size of every cluster which can be found in its domain, than the whole linked list array of former edge cluster entries will have to be broadcasted to every processor.

Extension to capability of building clusters for a Monte Carlo cluster algorithm

An energetic cluster criterion

A very important fact about a Monte Carlo algorithm in general is, that it has to fulfill detailed balance. That is the probability of going from one state into another in a Monte Carlo step must be the same as the probability of getting back. In cluster algorithms this is ensured by the introduction of a certain probability to cut a geometric cluster into smaller parts dependent on energy and temperature as it was postulated by R. H. Swendsen and J. S. Wang in 1986 [8]. Actually every bond between two particles that geometrically belong to one cluster is cut by a probability calculated from their pair energy. The original algorithm dealed with systems of discrete degrees of freedom like the Potts spin models. U. Wolff extended this to a theory which could be applied to continuous spin systems like the $\mathcal{O}(n)$ models in general [9]. To compare two spins his approach uses a projection of the spins to a predefined direction. But still the system has to be simulated on a lattice and the energy is not distance dependent. It is not yet clear, how the criterion of detailed balance can be fulfilled for the cluster building for a system with positional and angular degrees of freedom exhibiting frustrated configurations.

Necessary modifications

Due to these considerations the present work was focused to build the basis of a Monte Carlo cluster algorithm. This implies some modifications of the previously described algorithm. The most important difference is that the cluster labels are obviously no longer associated with cells of the virtual lattice but with each molecule in the system since it must be possible that every molecule belongs to an independent cluster, even if two or more of them are assigned to one cell.

A short pseudo code shows the approach for the local cluster identification.

```
initialize every particle with a globally unique cluster label
all clusters have size 1
DO FOR every cell
  DO FOR every pair of molecules in this cell
    IF energetic criterion fulfilled
      get proper labels of both particles
      # ^ ^ important to do this here, can change in every step!
      link referring clusters to smaller proper label, sum size
    END IF
  END DO
  DO FOR every molecule inside this cell
    DO FOR all molecules in (half of) the neighboring cells
      IF both cluster criteria fulfilled
        get proper labels of both particles
        # ^ ^ important to do this here, can change in every step!
        link referring clusters to smaller proper label, sum size
      END IF
    END DO
  END DO
END DO
```

The previously described order of *(i)* do the linking within each cell and *(ii)* through the neighboring cells in the lattice; is not necessary. However it seems to be impossible to have real speed up at this spot. The checking has to be done for all molecules and cannot be stopped for a certain cell if one link is found as it was possible in the method described before. This is a serious loss in performance.

Thinking about the communication the overall scheme will be conserved while the data that have to be exchanged will be different. The `cluster_id_border` array is handled exactly as it was done in the purely geometric cluster search. It is clear that also the positions and orientations have to be sent for every particle on the border. These are saved in `rxyz_border` again. To identify the entries of the `rxyz_border` array with the linking information in `cluster_id_border` also the former, globally unique label of every particle is important (see the initialization part of the pseudo code). This means there will be an additional array with 1 integer value for each molecule in the virtual bordering area of a processor, containing the original label. Obviously none of the sizes of these arrays is given in advance. So it is a good way to let the receiving processor guess the dimension of the original label array (which has the least amount of data inside) plus an uncertainty and receive an array of that size with the amount of data in the other arrays attached to the end. It would be possible to send the dimensions of the arrays in an additional communication step before transmitting them, however it seems that some overhead cannot be avoided.

Scaling

After all the theoretical discussions about a good implementation of the desired functionality it is now time to look at real time measurements. All concerning measurements where carried out on the *ZAMPaNo* [11], a parallel computer with 8 compute nodes. One node consists of 4 processors and has 2GB shared memory. Therefore the memory model is only partly distributed. At first glance the communication tree should provide a *tree-like* scaling as it is described in [10]. That is the scaling is nearly linearly for low numbers of processors and saturates at a constant value for large numbers. It does not show the behavior of *all-to-all* communication where for large processor numbers the performance is getting worse.

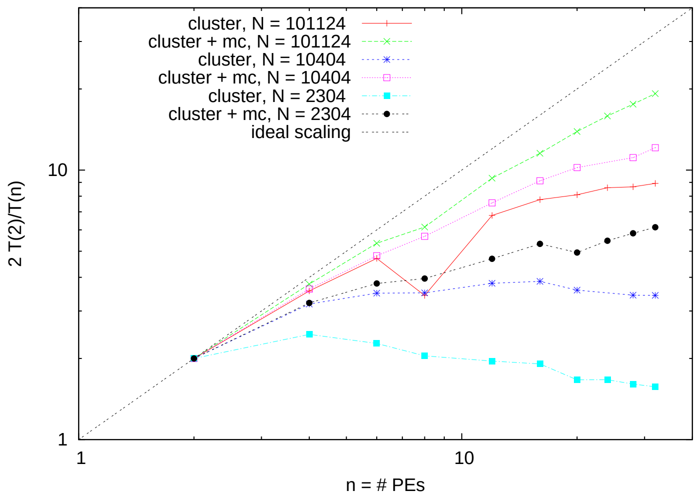


Fig. 10: Scaling of the cluster algorithm itself and attached to the Monte Carlo step in *"SpinCG^{2d}"* for different system sizes and processor numbers.

Fig. 10 shows the measured scaling of the cluster routine itself and in combination with the *"SpinCG^{2d}"* program. Since the latter has an outstanding scaling behavior it is easy to accept that the combination of both always shows a better scaling compared to the pure cluster search. However it is found that the described technique scales more like an *all-to-all* communication scheme. Remembering fig. 8 and the conclusion that some kind of *all-to-all* communication is necessary this is not really astonishing. The reason for the difference of the *tree-like* communication and the scheme in the existing case is that the amount of data that has to be transmitted is also rising with the number of processors since the overall edge length is rising. This is not the case in the underlying eqn. 107 in [10]:

$$c(N_p) = \log_2(N_p)\lambda + \sum_{n=1}^{\log_2(N_p)} \frac{2^{n-1}\chi}{N_p}\,, \tag{6}$$

where λ is the latency and χ the bandwith. To have a better view on the real complexity with respect to N_p the sum can be simplified to:

$$c(N_p) = \log_2(N_p)\lambda + \left(1 - \frac{1}{N_p}\right)\chi \tag{7}$$

Here N_p is the number of processors, $c(N_p)$ expresses the relative portion of communication with respect to communication. In eqn. 6 the term $2^{n-1}/N_p$ is a normalized amount of data to be send within each step. The overall amount is always 1, so in the last step of communication $1/2$ is sent. The edge length is growing with about $\log_2(N_p)$ in two dimensions. So it can be derived from 6 the leading behavior:

$$\begin{aligned} c(N_p) &= \log_2(N_p)\lambda + \sum_{n=1}^{\log_2(N_p)} \log_2(N_p)\frac{2^{n-1}\chi}{N_p} \\ &= \log_2(N_p)\lambda + \log_2(N_p)\left(1 - \frac{1}{N_p}\right)\chi \,. \end{aligned} \tag{8}$$

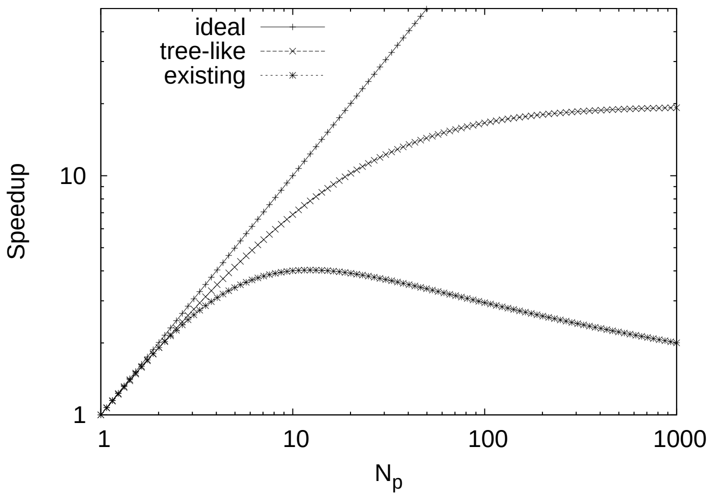


Fig. 11: Comparison of *tree-like* scaling and that one which is theoretically expected for the communication scheme used in the cluster search algorithm for $\lambda = 10^{-4}$ and $\chi = 0.05$.

Since it cannot be avoided that the sum of the edges of all processors grows by increasing the number of nodes, it is at least satisfactory that after the slope of speedup got negative it is slowly increasing again (see fig. 11). The most important fact is that the good scaling behavior of *"SpinCG^{2d}"* is not destroyed by implementing the cluster algorithm into it. For that reason fig. 12 shows the absolute running times of different routines called in *"SpinCG^{2d}"* and the cluster labeling algorithm. The "local cluster" search is done absolutely independently on every processor and therefore it exhibits a good scaling behavior. The critical part of the *"SpinCG^{2d}"* concerning the number of processors is the "spatial decomposition" which contains communication. Its time consumption is more or less constant, independent of the number of PEs. The two parts of the cluster algorithm containing communication are fast enough to not make the negative scaling a real problem. The most time consuming but absolutely parallel "interaction" part is probably slowed down by the "spatial decomposition" at nearly the same number of processors where the global cluster identification becomes important, maybe this happens even earlier.

Cluster size distributions

Besides the important fact that the cluster search can extend the functionality of the Monte Carlo simulation, it is of course also possible to measure cluster size histograms or study percolation in final configurations. Due to the limited time only the distributions of cluster sizes in simulation of DNA molecules was studied. The measurement was done by creating 100 final configurations of a system with $N = 5184$ DNA strands by *"SpinCG^{2d}"* starting from a density $\rho = 2/\sqrt{3}d^2$, where $d = 35.0$Å was chosen. From

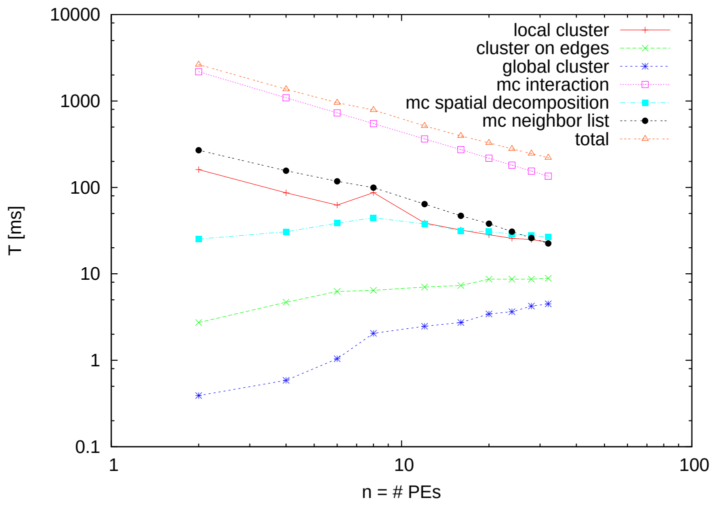


Fig. 12: The runtime of different routines called in *“SpinCG^{2d}”* and necessary steps of the cluster search for various numbers of processors and a system size of about 100000 molecules.

final configurations an average distribution of cluster sizes was obtained. Since every configuration was obtained by a definite number of Monte Carlo steps originating from the same initial set-up (a hexagonal lattice) the final systems were statistically absolutely independent and the error bars were calculated just as the standard deviation.

The measurements were carried out for different values of the charge compensation θ. The Debye screening length was always given by $\kappa^{-1} = 10\text{Å}$. As it was discussed in the beginning for this κ a phase transition is to be expected for $\theta \approx 0.86$. For smaller charge compensations the potential becomes completely repulsive which results in a kind of crystalline state of the system. For this reason there is no way to identify clusters, because it is even not possible to define a cluster criterion. Either all particles would be in one large cluster, or every particle would be identified to be separate from all others. Because of the repulsive form of the potential it is more reasonable to talk of a crystalline phase without any clusters. For higher values of the charge compensation θ the structure will be interesting because of the frustrations arising from the orientation dependent part of the Kornyshev-Leikin potential. It is interesting if there can be found some power law for the cluster size distribution in analogy to percolation on lattices for example.

Fig. 13.**a** shows a system with 324 DNA strands that was obtained by a simulation of 10000 Monte Carlo steps for $\theta = 0.95$. For this small number of molecules the picture is not capable to show the overall structure of such a system. But it depicts the qualitative fact that large clusters arise where the strands orient perpendicular to each other according to the minimum in the two dimensional potential energy landscape. Unlike in 13.**b**, where the same simulation was done with $\theta = 0.865$ close to the expected critical point. Most of the particles have a serious distance from each other while some of them gather into very small clusters at the flat minimum in the potential curve. This difference will be reflected in the cluster size histograms which allow qualitative statements about the structure of the system. In fig. 13.**c** and **d** the corresponding histograms to the structures in 13.**a** and **b** manifest this difference. For high charge compensation the probability of huge clusters is still in an acceptable range. Clusters at sizes greater than 2000 are likely to be percolating because they contain about half of all molecules in the system. The searched power law is well reproduced for long simulation times and the according exponent can be measured. For a rapid cooling and less Monte Carlo steps it can be observed a deviation from the power law at a certain point. Obviously this can be taken as a sign for too short simulation times. Near to the critical point (fig. 13.**d**) the slope of the cluster size curve changes dramatically (see

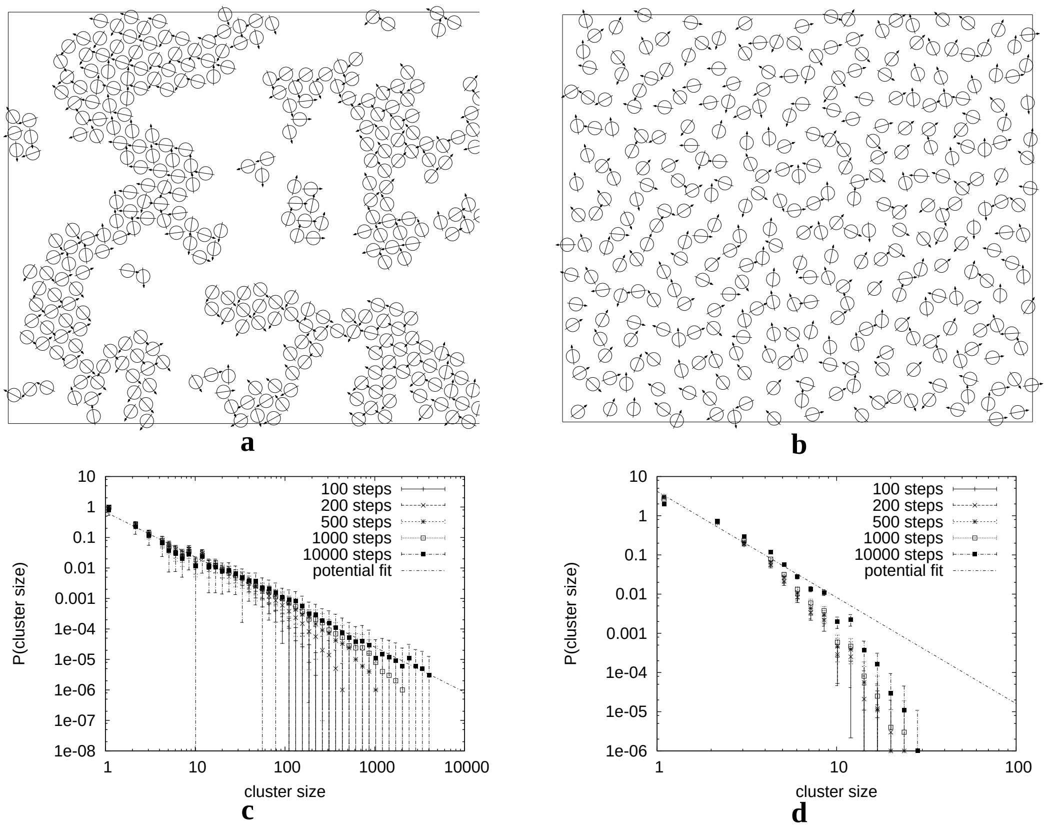


Fig. 13: **a**, **b**: Final configuration of a model DNA aggregate with 324 molecules for $\theta = 0.95$ (**a**) and $\theta = 0.865$ (**b**) after simulated annealing over 10000 steps. The vectors show the azimuthal orientation of a molecule. **c**, **d**: The according cluster size histograms, averaged over 100 final configurations of a system with 5184 particles.

also fig. 14). It can also be seen that the deviation from a power law which could be observed for short simulation times in fig. 13.**c** appears even for 10000 Monte Carlo steps in this case. Therefore for the rough estimation of an exponent only the first values where taken into account. This behavior seems to be similar to the critical slowing down which often occurs at phase transitions in spin simulations. It would be interesting to see the effect of a real cluster algorithm in the Monte Carlo step. May be it would help to increase the quality of the data.

Finally in fig. 14 the exponents of the cluster size distributions are plotted for several θ. Although the error bars are only obtained from the least square fit (double standard deviation) the qualitative drop of the exponent close to $\theta = 0.86$ is obvious.

Conclusion and Outlook

The introduced cluster search algorithm can be used to build a Monte Carlo cluster algorithm for continuous two dimensional off-lattice spin systems on the one hand, and the study of cluster distributions and percolation on the other. It is designed for parallel, distributed memory systems and does not need to replicate memory. Although a variety of optimizations were implemented it cannot be avoided that the scaling of the method can have negative slopes. But since its absolute run time is very short it does not have any considerable influence on the good scaling behavior of “*SpinCG^{2d}*” and can therefore extend the functionality of the existing code.

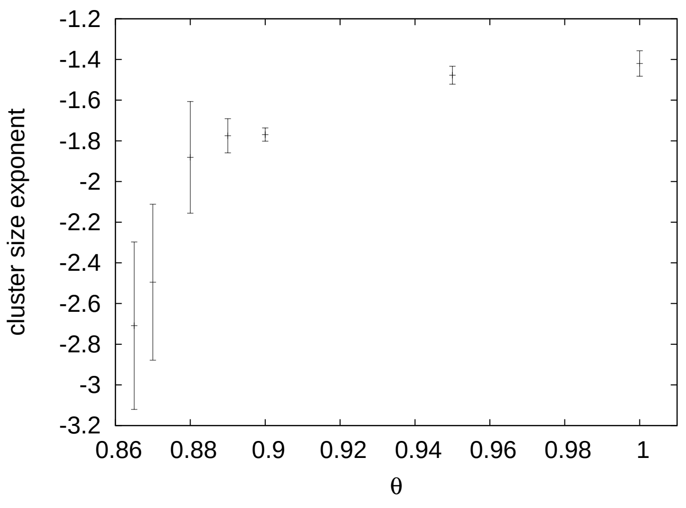


Fig. 14: The rough exponent of the cluster size distributions for different values of θ.

It will be very interesting to implement an energetic criterion which could make the method capable of lowering the autocorrelation times in the described systems. Measurements of autocorrelation times in the current and extended version would be necessary to proof that. Also the few measurements on system structure exponents should be carried out more thorough. An extension of the technique to three dimensions would be desirable and should not be complicated in principle.

Acknowledgments

I would like to thank PD Dr. Lippert and Dr. Esser for inviting me to the “Guest student program 2004” at the ZAM/NIC, Jülich research center, where this work was produced. Also i want to express my gratitude to my supervisor Dr. Godehard Sutmann, who did not only introduce me to a very interesting topic of research but also assisted me with the development, implementation, analysis and application of the described algorithm. I am grateful to the employees of the ZAM whose engagement was essential for the great experience I had here in Jülich, especially Jörg Striegnitz who taught us the basics of parallel programming.

References

1. A. A. Kornyshev and S. Leikin, J. Chem. Phys. **107**, 3656 (1997)
2. G. Sutmann, Monte Carlo Simulations of Columnar DNA Aggregates, to be published
3. G. Sutmann, Temperature induced structural transitions in columnar DNA aggregates, to be published
4. A. A. Kornyshev and S. Leikin, Proc. Natl. Acad. Sci. U.S.A. **95**, 13597 (1998)
5. H. M. Harreis, A. A. Kornyshev, C. N. Likos, H. Löwen, G. Sutmann, Phys. Rev. Lett. **89**, 018303-1 (2002)
6. G. Sutmann, *SpinCG^{2d} - a parallel Monte Carlo program for spin systems*, in preparation
7. J. Hoshen, R. Kopelman, Phys. Rev. B **14**, 3438 (1976)
8. R. H. Swendsen, J. S. Wang, Phys. Rev. Lett. **58**, 86 (1986)
9. U. Wolff, Phys. Rev. Lett. **62**, 361 (1988)
10. G. Sutmann, *Classical Molecular Dynamics*, in *Quantum Simulations of Complex Many-Body Systems: From Theory to Algorithms*, lecture Notes, J. Grotendorst, D. Marx, A. Muramatsu (Eds.), John von Neumann Institute for Computing, Jülich, 2002, Vol. 10, p. 211-254.
11. http://zampano.zam.kfa-juelich.de/