At the time of writing of this book, the nature of the transition in two-dimensional hard disks (in the thermodynamic limit) has not been cleared up. It might be a first-order phase transition, or a continuous Kosterlitz–Thouless transition. It is now well understood that in two dimensions the nature of transition depends on the details of the microscopic model. The phase transition in hard disks could be of first order, but a slight softening-up of the interparticle potential would make the transition continuous. The question about the phase transition in hard disks—although it is highly model-specific—would have been cleared up a long time ago if only we disposed of Monte Carlo algorithms that, while respecting detailed balance, allowed us to move in a split second between configurations as different as the two configurations in Fig. 2.47. However, this is not the case. On a year 2005 laptop computer, we have to wait several minutes before moving from a crystalline configuration to a liquid one, for 100 particles, even if we used the advanced methods described in Subsection 2.4.1. These times get out of reach of any simulation for the larger systems that we need to consider in to understand the finite-size effects at the transition. We conclude that simulations of hard disks do not converge in the transition region (for systems somewhat larger than those considered in Fig. 2.47). The failure of computational approaches keeps us from answering an important question about the phase transition in one of the fundamental models of statistical physics. (For the transition in three-dimensional hard spheres, see Hoover and Ree (1968).)

2.5 Cluster algorithms

Local simulation algorithms using the Metropolis algorithm and molecular dynamics methods allow one to sample independent configurations for large systems at relatively high densities. This gives often very important information on the system from the inside, so to speak, because the samples represent the system that one wants to study. In contrast, analytical methods are often forced to extrapolate from the noninteracting system (see the discussion of virial expansions, in Subsection 2.2.2). Even the Monte Carlo algorithm, however, runs into trouble at high density, when any single particle can no longer move, so that the Markov chain of configurations effectively gets stuck during long times (although it remains, strictly speaking, ergodic).

In the present section, we start to explore more sophisticated Monte Carlo algorithms that are not inspired by the physical process of single-particle motion. Instead of moving one particle after the other, these methods construct coordinated moves of several particles at a time. This allows one to go from one configuration, a, to a very different configuration, b, even though single particles cannot really move by themselves. These algorithm methods can no longer be proven correct by common sense alone, but by the proper use of a priori probabilities. The algorithms generalize the triangle algorithm of Subsection 1.1.6, which first

went beyond naive pebble-throwing on the Monte Carlo heliport. In many fields of statistical mechanics, coordinated cluster moves—the displacement of many disks at once, simultaneous flips of spins in a region of space, collective exchange of bosons, etc.—have overcome the limitations of local Monte Carlo algorithms. The pivot cluster algorithm of Subsection 2.5.2 (Dress and Krauth 1995) is the simplest representative of this class of methods.

2.5.1 Avalanches and independent sets

By definition, the local hard-sphere Monte Carlo algorithm rejects all moves that produce overlaps (see Fig. 2.23). We now study an algorithm, which accepts the move of an independent disk even if it generates overlaps. It then simply moves the overlapped particles out of place, and starts an avalanche, where many disks are constrained to move and, in turn, tip off other disks. Disks that must move but which entail no other moves are called "terminal". For simplicity, we suppose that the displacement vector is the same for all disks (see Fig. 2.48 and Alg. 2.17 (`naive-avalanche`)).

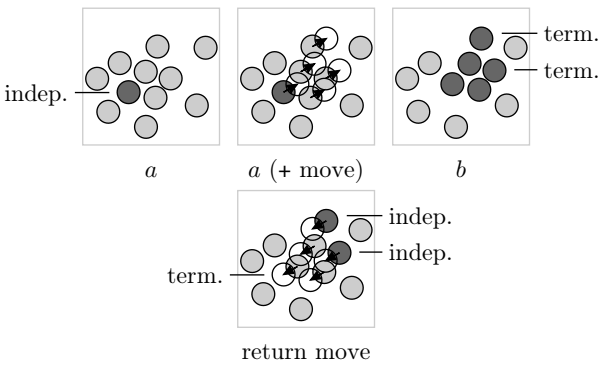

Fig. 2.48 Avalanche move $a \to b$ and its return move, with independent and terminal disks.

Detailed balance stipulates that, for hard spheres, the return move be proposed with the same probability as the forward move. From Fig. 2.48, it follows that the forward and the return move swap the labels of the independent and the terminal disks. In that example, the return move has two independent disks, and hence is never proposed by Alg. 2.17 (`naive-avalanche`), so that $a \to b$ must be rejected also. Only avalanche moves with a single terminal disk can be accepted. This happens very rarely: avalanches usually gain breadth when they build up, and do not taper into a single disk.

Algorithm 2.17 (`naive-avalanche`) has a tiny acceptance rate for all but the smallest displacement vectors, and we thus need to generalize it, by allowing more than one independent disk to kick off the avalanche. For

procedure `naive-avalanche`
input $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$
$k \leftarrow \mathtt{nran}\,(1, N)$
$\boldsymbol{\delta} \leftarrow \{\mathtt{ran}\,(-\delta, \delta)\,, \mathtt{ran}\,(-\delta, \delta)\}$
construct move (involving disks $\{k_1, \ldots, k_M\}$)
if (move has single terminal disk**) then**
  for $l = 1, \ldots, M$ **do**
    $\mathbf{x}_{k_l} \leftarrow \mathbf{x}_{k_l} + \boldsymbol{\delta}$
output $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$

Algorithm 2.17 `naive-avalanche`. Avalanche cluster algorithm for hard disks, with a low acceptance rate unless $|\boldsymbol{\delta}|$ is small.

concreteness, we suppose that avalanches must be connected and that they must have a certain disk l in common. Under this condition, there

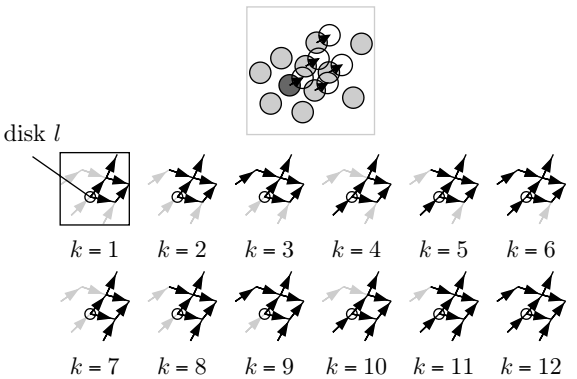

Fig. 2.49 The move $a \to b$, and all avalanches containing the disk l for a displacement $\boldsymbol{\delta}$. The avalanche $k = 1$ realizes the move.

are now 12 connected avalanches containing disk l (see Fig. 2.49). This means that the a priori probability of selecting the frame $k = 1$, rather than another one, is $1/12$. As in the study of the triangle algorithm, the use of a priori probabilities obliges us to analyze the return move. In the configuration b of Fig. 2.48, with a return displacement $-\boldsymbol{\delta}$, 10 connected avalanches contain disk l, of which one (the frame $k = 8$) realizes the return move (see Fig. 2.50). The a priori probability of selecting this return move is $1/10$. We thus arrive at

$$\mathcal{A}(a \to b) = \frac{1}{12} \quad \left\{ \begin{array}{c} \text{one of the 12 avalanches} \\ \text{in Fig. 2.49} \end{array} \right\},$$

$$\mathcal{A}(b \to a) = \frac{1}{10} \quad \left\{ \begin{array}{c} \text{one of the 10 avalanches} \\ \text{in Fig. 2.50} \end{array} \right\}.$$

These a priori probabilities must be entered into the detailed-balance

condition

$$\underbrace{\mathcal{A}(a \to b)}_{\text{propose}} \underbrace{\mathcal{P}(a \to b)}_{\text{accept}} = \underbrace{\mathcal{A}(b \to a)}_{\text{propose}} \underbrace{\mathcal{P}(b \to a)}_{\text{accept}}.$$

Detailed balance is realized by use of the generalized Metropolis algorithm

$$\mathcal{P}(a \to b) = \min\left(1, \frac{12}{10}\right) = 1.$$

It follows that the move $a \to b$ in Fig. 2.48 must be accepted with probability 1.

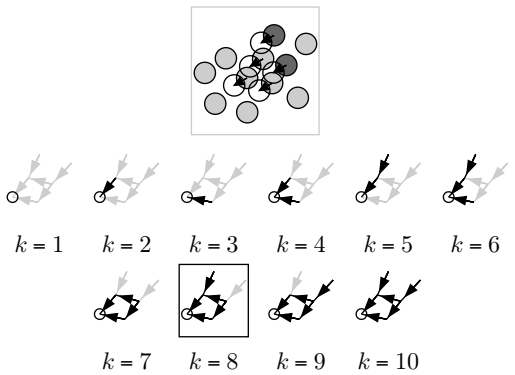

Fig. 2.50 Return move $b \to a$ and all avalanches containing disk l for a displacement $-\boldsymbol{\delta}$. The avalanche $k = 8$ realizes the return move.

The algorithm we have discussed in this subsection needs to count avalanches, and to sample them. For small systems, this can be done by enumeration methods. It is unclear, however, whether a general efficient implementation exists for counting and sampling avalanches, because this problem is related to the task of enumerating the number of independent sets of a graph, a notoriously difficult problem. (The application to the special graphs that arise in our problem has not been studied.) We shall find a simpler solution in Subsection 2.5.2.

2.5.2 Hard-sphere cluster algorithm

The avalanche cluster algorithm of Subsection 2.5.1 suffers from an imbalance between forward and return moves because labels of independent and terminal disks are swapped between the two displacements, and because their numbers are not normally the same. In the present subsection, we show how to avoid imbalances, rejections, and the complicated calculations of Subsection 2.5.1 by use of a pivot: rather than displacing each particle by a constant vector $\boldsymbol{\delta}$, we choose a symmetry operation that, when applied twice to the same disk, brings it back to the original position. For concreteness, we consider reflections about a vertical line, but other pivots (symmetry axes or reflection points) could also be used.

With periodic boundary conditions, we can scroll the whole system such that the pivot comes to lie in the center of the box (see Fig. 2.51).

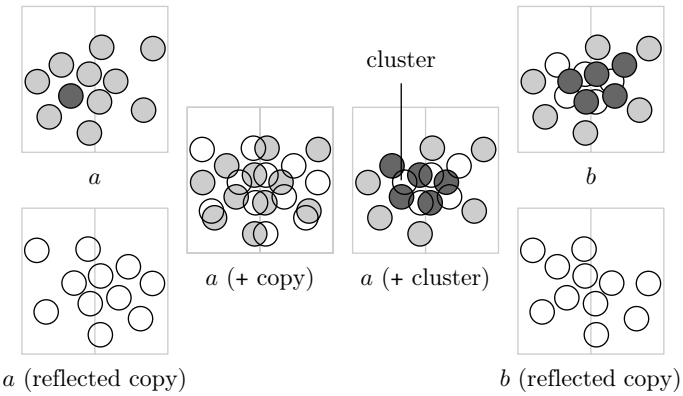

Fig. 2.51 Hard-sphere cluster algorithm. Some particles are swapped between the original configuration and the copy, i.e., they exchange color.

Let us consider a copy of the original system with all disks reflected about the symmetry axis, together with the original configuration a. The original and the copy may be superimposed, and disks in the combined systems form connected clusters. We may pick one cluster and exchange in it the disks of the copy with those in the original configuration (see configuration b in Fig. 2.51). For the return move, we may superimpose the configuration b and a copy of b obtained through the same reflection that was already used for a. We may pick a cluster which, by symmetry, makes us come back to the configuration a. The move $a \to b$ and the return move $b \to a$ satisfy the detailed-balance condition. All these transformations map the periodic simulation box onto itself, avoiding problems at the boundaries. Furthermore, ergodicity holds under the same conditions as for the local algorithm: any local move can be seen as a point reflection about the midpoint between the old and the new disk configuration. The basic limitation of the pivot cluster algorithm is that, for high covering density, almost all particles end up in the same cluster, and will be moved together. Flipping this cluster essentially reflects the whole system. Applications of this rejection-free method will be presented in later chapters.

To implement the pivot cluster algorithm, we need not work with an original configuration and its copy, as might be suggested by Fig. 2.51. After sampling the symmetry transformation (horizontal or vertical reflection line, pivot, etc.), we can work directly on particles (see Fig. 2.52). The transformation is applied to a first particle. From then on, we keep track (by keeping them in a "pocket") of all particles which still have to be moved (one at a time) in order to arrive at a legal configuration (see Alg. 2.18 (`hard-sphere-cluster`)). A single iteration of the pivot

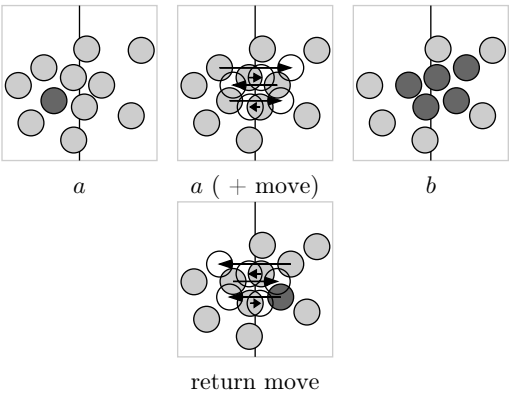

Fig. 2.52 Hard-sphere cluster algorithm without copies. There is one independent particle. Constrained particles can be moved one at a time.

cluster algorithm consists in all the operations until the pocket is empty. The inherent symmetry guarantees that the process terminates.

In conclusion, we have considered in this subsection a rejection-free cluster algorithm for hard spheres and related systems, which achieves perfect symmetry between the forward and the return move because the clusters in the two moves are the same. This algorithm does not work very well for equal disks or spheres, because the densities of interest (around the liquid–solid transition) are rather high, so that the clusters usually comprise almost the entire system. This algorithm has, however, been used in numerous other contexts, and generalized in many ways, as will be discussed in later chapters.

procedure `hard-sphere-cluster`
input $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$
$k \leftarrow$ `nran` $(1, N)$
$\mathcal{P} \leftarrow \{k\}$ (the "pocket")
$\mathcal{A} \leftarrow \{1, \ldots, N\} \setminus \{k\}$ (other particles)
while $(\mathcal{P} \neq \{\})$ **do**
  $i \leftarrow$ any element of $\mathcal{P}$
  $\mathbf{x}_i \leftarrow T(\mathbf{x}_i)$
  for $\forall\ j\ \in\ \mathcal{A}$ **do**
    if (i overlaps j) **then**
      $\mathcal{A} \leftarrow \mathcal{A} \setminus \{j\}$
      $\mathcal{P} \leftarrow \mathcal{P} \cup \{j\}$
  $\mathcal{P} \leftarrow \mathcal{P} \setminus \{i\}$
output $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$

Algorithm 2.18 `hard-sphere-cluster`. Cluster algorithm for N hard spheres. T is a random symmetry transformation.