Speed-up of Monte Carlo simulations by sampling of rejected states

Daan Frenkel*

FOM (Fundamenteel Onderzoek der Materie) Institute for Atomic and Molecular Physics, Kruislaan 407, 1098 SJ Amsterdam, The Netherlands

Communicated by David Chandler, University of California, Berkeley, CA, October 27, 2004 (received for review March 24, 2004)

The Markov chain Monte Carlo method is an important tool to estimate the average properties of systems with a very large number of accessible states. This technique is used extensively in fields ranging from physics to genetics and economics. The rejection of trial configurations is a central ingredient in existing Markov chain Monte Carlo simulations. I argue that the efficiency of Monte Carlo simulations can be enhanced, sometimes dramatically, by properly sampling configurations that are normally rejected. This ''waste-recycling'' of microstates is useful in sampling schemes in which only one of a large set of trial configurations is accepted. It differs fundamentally from schemes that extract information about the density of macrostates from virtual Monte Carlo moves. As a simple illustration, I show that the method greatly improves the calculation of the order-parameter distribution of a two-dimensional Ising model. This method should enhance the efficiency of parallel Monte Carlo simulations significantly.

algorithms | statistical physics

Numerical simulations of many-body systems started in 1953 with the introduction of the Markov chain Monte Carlo (MCMC) scheme (1). This scheme was designed to estimate the average properties of systems (many-body systems are only one example) with a very large number of accessible states (''configurations''). To achieve this estimate, the algorithm generates a random walk through configuration space such that any individual state is visited with a frequency proportional to its (Boltzmann) weight. The desired estimate is then obtained as an unweighted average over visited states. In this article, I use the language of many-body systems obeying Boltzmann statistics, but my conclusions depend in no way on this restriction.

To visit states with the correct frequency, the MCMC algorithm generates random trial moves from the current (''old'') state (o) to a new state (n). In the case of molecular systems, for example, such a move might be the displacement of a single particle. These trial moves may be either accepted or rejected. The acceptance criterion is chosen such that the probability to find the system in state i is proportional to the Boltzmann weight $P_B(i)$. The standard ''dogma'' of MCMC simulations is that rejected states should not be included in the sampling. As a consequence, conventional Monte Carlo algorithms are rather wasteful because they tend to generate many trial states (often a majority) that are not used in the computation of averages. In this article, I argue that we can do better; the properties of rejected states can be included in the sampling. In the case in which many trial states are generated simultaneously, this approach may lead to a dramatic improvement in the statistical accuracy of Monte Carlo simulations.

To explain this approach, I briefly review the idea behind MCMC algorithms. Consider a thought experiment in which we apply a particular Monte Carlo rule not to a single state but to an equilibrium ensemble of states (i.e., a collection of systems in which every state occurs with a frequency proportional to its weight). Clearly, if a single Monte Carlo step is applied to every initial state, an equilibrium distribution of final states should result, which means that every state still occurs with the same frequency and, hence, that the total number of transitions from a given state o to all other states should be balanced by the number of transitions from these states to state o. To show that even an arbitrary initial distribution eventually relaxes to the equilibrium distribution (2), it is often convenient to impose a stronger condition (''detailed balance'') that requires that the number of transition from the old state o to a given final state n is exactly balanced by the number of transitions from n to o. Moreover, the algorithm should be ''ergodic,'' meaning that every microstate can be reached in a finite number of steps from every other microstate (2).

The original scheme proposed by Metropolis *et al.* (1) was constructed by using this condition of detailed balance. If $P_B(o)$ ($P_B(n)$) denotes the probability to find the system in state o (n) and $\alpha(o \to n)$ ($\alpha(n \to o)$) denotes the conditional probability to perform a trial move from o to n (n to o), then the probability $P_{\rm acc}(o \to n)$ to accept the trial move from o to n is related to $P_{\rm acc}(n \to o)$ by the following:

$$P_B(o)\alpha(o \to n)\, P_{\rm acc}(o \to n) = P_B(n)\alpha(n \to o)\, P_{\rm acc}(n \to o). \quad [1.1]$$

Metropolis *et al.* (1) made the assumption that $\alpha(o \to n) = \alpha(n \to o)$, and it then follows that

$$\frac{P_{\rm acc}(o \to n)}{P_{\rm acc}(n \to o)} = \frac{P_B(n)}{P_B(o)}. \quad [1.2]$$

Eq. **1.2** does not yet fix the acceptance probability. The choice of ref. 1 was as follows:

$$P_{\rm acc}(o \to n) = \min\left\{1, \frac{P_B(n)}{P_B(o)}\right\}. \quad [1.3]$$

There now exists a great variety of MCMC algorithms that differ in the way in which trial moves are generated (3, 4) (i.e., that use different choices for $\alpha(o \to n)$). In some cases, a biased generation of the trial move can greatly enhance the associated acceptance probability (e.g., ref. 4). In fact, there even exist techniques, such as the Swendsen–Wang cluster algorithm (5) and related schemes (a powerful off-lattice cluster algorithm is described in ref. 6), that have a 100% acceptance probability of trial moves. Yet, these algorithms are not free of waste. On the contrary, they may generate a large number of potential states n, only one of which is accepted, whereas the rest are rejected, either explicitly (7–14) or implicitly (5).

The detailed-balance condition mentioned above, together with the condition of ergodicity (e.g., ref. 2), is sufficient but not necessary to guarantee equilibrium sampling. In 1999, Manousiouthakis and Deem (15) showed that a weaker condition (''balance'') is both sufficient and necessary to guarantee that a particular Markov chain algorithm leads to equilibrium sam-

Abbreviations: CBMC, configurational bias Monte Carlo; MCMC, Markov chain Monte Carlo.

*E-mail: frenkel@amolf.nl.

pling. Leaving technicalities aside, the balance condition requires only that the Markov chain leave the equilibrium distribution invariant. By using this balance condition, it is now possible to formulate an algorithm that samples both rejected and accepted states, although (in general) with unequal weight. First, I briefly explain how the algorithm works, and then I give an explicit step-by-step description. Last, I explain why the algorithm satisfies the balance condition of ref. 15.

Methods

As its starting point, the current algorithm uses any valid MCMC algorithm. Such an algorithm generates trial moves that satisfy the condition of (super)detailed balance (see *Superdetailed Balance*). Let us consider a trial move that starts from an initial state o, with Boltzmann weight $P_B(o)$, and may end up in the original state o or in any of a set of trial states $\{n\}$. (For Metropolis Monte Carlo, there is only one o and one n, but in other algorithms there may be many.)

Now suppose that we replace a single trial move by a very large number of moves that sample the same subset of states $(o + \{n\})$ by using some form of importance sampling that satisfies detailed balance (e.g., Metropolis sampling). If we consider the limit at which L (the number of moves within the set $(o + \{n\})$) goes to infinity, then every possible final state n_i will be visited with a frequency proportional to $P_B(n_i)$. For any finite value of L, we could have accumulated averages in the normal way (i.e., as an unweighted average over visited states). However, we do not have to do this sampling explicitly because, in the limit $L \to \infty$, we already know the probability with which every state n_i will be visited, which is simply $P_B(n_i)/(\Sigma\ P_B(i'))$, where i' includes all possible states, including the original state o. At the end of the set of submoves, the system will be in a particular state f, with a probability $P_B(f)/(\Sigma\ P_B(i'))$. We can now dispense with the L submoves completely and replace them by a single "coarse-grained" move from state o to state f (dashed line segments in Fig. 1). State f is the initial state for the next coarse-grained Monte Carlo move. Fig. 1 shows the relation between the coarse-grained moves and the underlying sampling of all intermediate states.

For a given set of substates, the transition probabilities for coarse-grained moves between o and f satisfy detailed balance. Hence, at the level of these "macrosteps," we perform correct equilibrium sampling of the states connected by coarse-grained moves. One might argue that this algorithm does not correspond to a Markov chain because, during the submoves, the system remembers its previous states. However, there is another way of looking at the submoves. Consider a trial move starting from a state o. The first step in any Monte Carlo move is to generate at least one trial state f (for instance, the configuration that results if we move a single particle). By selecting state f as the final state of our trial move, we have established a temporary link between states o and f. As long as this link is present, state o is only connected to state f, and vice versa. Normally, we randomly generate new links at every Monte Carlo step, but although the updating of links must be a Markov process, it does not need to proceed at the same rate as the Markov chain for the submoves. Hence, whereas the overall sampling process is a Markov chain, between link updates, we sample a subpopulation of linked initial and final states. Note that the overall acceptance of trial moves in this algorithm is determined by $P_B(f)/(\Sigma\ P_B(i'))$, which is not the Metropolis rule but the so-called "symmetric" rule (16):

$$P_{\text{acc}}^{\text{sym}}\ (o\ \to\ f) = \frac{P_B(f)}{\Sigma_{\{i\}}\ P_B(i)}, \qquad \textbf{[2.1]}$$

where the set $\{i\}$ includes both states o and f. In other words, even if the underlying MCMC scheme to sample the set $o + \{n\}$) would be based on the Metropolis rule, the "coarse-grained" moves that we perform in practice satisfy a different rule. It is essential that the set of linked states be constructed as in a normal (biased or unbiased) Monte Carlo scheme. Otherwise, the algorithm will not satisfy detailed balance. This problem is well known in configurational bias Monte Carlo (CBMC) and related algorithms (4). As is explained in *Superdetailed Balance*, trial moves within a set of linked states should satisfy "superdetailed balance." This condition implies that the probability to generate the same set of linked states in both forward and reverse moves between states o and n should be such that detailed balance holds for trial moves within this set. In the case of biased sampling, the acceptance criterion (Eq. **2.1**) will not contain the bare Boltzmann weights but the weight functions that enter the Metropolis acceptance rule for that particular biasing scheme. For example, in the case of CBMC (7–9), the P'_B values in Eq. **2.1** would be replaced by the Rosenbluth weights of the old and new states. Averages of an arbitrary "measurable" quantity are then estimated as follows:

$$\langle A\rangle = \frac{1}{M}\sum_{m=1}^{M}\frac{\Sigma_{k\leftrightarrow m}\ P_B(k)A(k)}{\Sigma_{k\leftrightarrow m}\ P_B(k)}, \qquad \textbf{[2.2]}$$

where m labels the state of the system at the beginning of the mth coarse-grained step, and $k \to m$ denotes the set of substates linked to m during substate sampling (including m). Note that, for every coarse-grained move, we have to compute explicitly the weight $P_B(k)$ and the observable $A(k)$ for all $k \to m$.

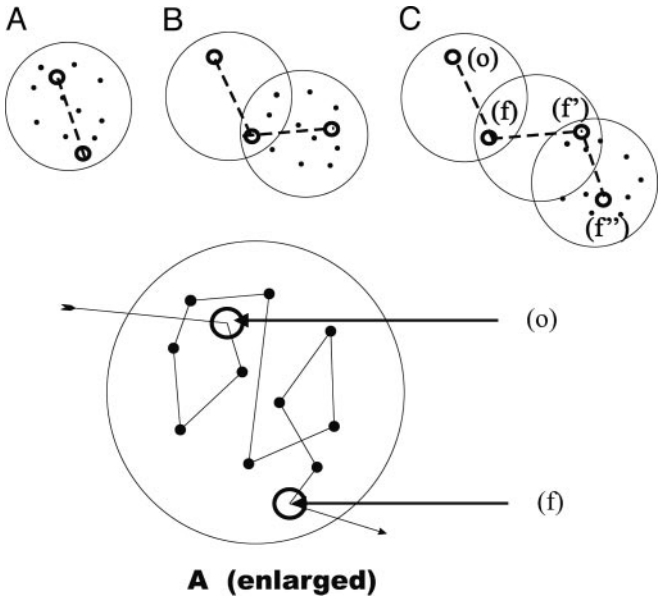

Fig. 1. Schematic representation of waste-recycling Monte Carlo. The large circles denote different sets of linked states. Initially, the system is in state *o* (open circle). As long as the set of linked states {*i*} is not changed, only the indicated points within the circle can be reached from state (*o*). The drawn line segments in the enlarged image *A* trace a possible path of the microscopic Monte Carlo. In practice, the number of steps is much larger (eventually, we consider the limit at which *L*, the number of steps between link changes, goes to infinity), such that every state is visited many times. Then, the fraction of the time that the system spends in a given state *s* is given by $P_B(s)/(\Sigma_{\{i\}}P_B i)$, where $P_B(i)$ denotes the (Boltzmann) weight of state *i*. The probability to end up in state *f* after *L* moves is given by Eq. **2.1**. (*B*) The next sequence of moves starts from the initial configuration (*f*). Another set of states (indicated in *B*) are linked to (*f*). After *L* Monte Carlo steps ($L \to \infty$), the system has moved to the state (*f'*) at the end of the dashed line segment. (*C*) Then, the linking changes again, and so on. The coarse-grained moves go from (*o*) → (*f*) → (*f'*) → (*f''*), etc.

In summary, the algorithm consists of the following steps:

1. Select a valid MCMC algorithm that generates trial states in a way that satisfies superdetailed balance.
2. Prepare the system under study in some initial state o, and set at zero the accumulators S_A that will we needed to estimate the average of property A.
3. Generate a set of trial states ($o + \{n\}$) by using the algorithm mentioned in step 1.
4. Compute the weights w_i [$I \in (o+ \{n\})$ that determine the acceptance probability of a move $o \rightarrow i$ in the MCMC algorithm of step 1 (e.g., for Metropolis sampling, $w_i = \exp(-U_i/kT)$)].
5. Update the accumulator S_A according to the following equation:

$$S_A \rightarrow S_A + \frac{\Sigma_i w_i A_i}{\Sigma_i w_i}.$$

6. If the total number of trial moves is less than a predetermined number M, select a final state $f \in (o + \{n\})$ with a probability $P_f = w_f/\Sigma_i w_i$. State f now becomes the original state o for the next move in step 3. Otherwise, go to step 7.
7. Compute estimates of the averages of all quantities A by using the following:

$$\langle A \rangle_{\text{est}} = \frac{S_A}{M}.$$

The algorithm that I describe satisfies the necessary "balance" condition (15). The reason is that the algorithm visits microstates with a probability that is dictated by the underlying MCMC algorithm. These probabilities are not affected by our choice to combine large numbers of such moves into a single coarse-grained move. However, it is important to realize that all averages that we compute, including fluctuations, should reflect the underlying Markov sampling of individual substates. Hence, for example, the average $\langle A^2 \rangle$ is as follows:

$$\langle A^2 \rangle = \frac{1}{M} \sum_{m=1}^{M} \frac{\Sigma_{k \leftrightarrow m} P_B(k) A^2(k)}{\Sigma_{k \leftrightarrow m} P_B(k)} \qquad \textbf{[2.3]}$$

and not

$$\frac{1}{M} \sum_{m=1}^{M} \left(\frac{\Sigma_{k \leftrightarrow m} P_B(k) A(k)}{\Sigma_{k \leftrightarrow m} P_B(k)} \right)^2.$$

Note that, even when the underlying MCMC algorithm satisfies detailed balance, the current algorithm only satisfies balance (15). The reason is that we perform steps in a fixed order. First we update the links, and then we sample the set $o + \{n\}$ and then update the links again, etc. The individual steps in this procedure satisfy detailed balance, but because they are carried out in fixed order, the composite move satisfies only the weaker balance condition.

As the simplest example, consider the Metropolis sampling of a fluid of atoms. An atom is selected at random, and a trial move is attempted from the old state o to the new state f. After one such trial move, the system will be either in state o or f. Normally, we would now consider a new trial move. However, in this scheme, we maintain the link between states o and f and carry out $M \gg 1$ Monte Carlo moves within this subset. These moves obviously satisfy detailed balance. As a consequence, the number of times M_i that the system is in state i (i is o or f) is $M \times P_B(i)/(P_B(o) + P_b(f))$. If we accumulate averages of a quantity A during these M moves, we get a contribution to the accumulator of A that is equal to the following:

$$\frac{1}{M}(M_o A_o + M_f A_f) = (P_B(o)A_o + P_B(f)A_f)/(P_B(o) + P_b(f)).$$

At the end of this sequence of moves, the system is in state i with a probability $P_B(i)/(P_B(o) + P_b(f))$. Rather than carry out the M moves explicitly, we directly compute the contribution of these M moves to the average. However, because the sampling could have been done explicitly, the validity of the original Monte Carlo algorithm guarantees the validity of the coarse-grained algorithm. This conclusion also holds if the number of linked states is larger than two.

Use of the current algorithm is limited when implemented in conventional ("Metropolis") Monte Carlo simulations, but it becomes very powerful in algorithms that generate many trial configurations simultaneously, as is the case for all parallel Monte Carlo simulations but also for many algorithms that are used in the context of polymer simulations. Examples are the CBMC method mentioned above (4), the recoil-growth algorithm (12–14), and the Dynamic Pruned Enriched Rosenbluth Method scheme (11). These schemes distinguish between old and new trial configurations. Normally, the old trial configurations are generated only to compute the acceptance probability. However, with the symmetric scheme, they may be selected. Yet the real advantage of this method shows up only when we generate a very large number of trial configurations in parallel.

Results and Discussion

For an illustration, consider the Swendsen–Wang cluster algorithm that was designed to sample a class of simple lattice models with discrete spins (5). This algorithm uses a scheme due to Fortuin and Kasteleyn (17) to divide the spin system into clusters of connected spins that are subsequently flipped as a unit with a 50% probability (for convenience, we limit the discussion to the case of an Ising spin model). The rule to construct these clusters has been designed such that any of the resulting states can be automatically accepted. However, if the system has been subdivided into n clusters, there are 2^n possible final states, and traditionally we accept only one such state. Hence, the algorithm is not waste-free.

With this approach, we can do much better because we can sample all possible states that result from flipping the clusters. As an example, consider the sampling of the probability density $P(S)$ that describes the probability to find the system with a total spin S. In the conventional Swendsen–Wang algorithm, every move m generates one value of S. During the simulation, we simply keep track of the number of times that a particular value of S is visited. Consider this approach. Let us denote the value of the spin of cluster i by s_i. The total spin of the system is as follows:

$$S = \sum_{i=1}^{n} s_i. \qquad \textbf{[3.1]}$$

We can now compute the characteristic function $f(k)$ defined as follows:

$$f(k) = \langle \exp(ikS) \rangle, \qquad \textbf{[3.2]}$$

where the angular brackets denote thermal averaging. By using the fact that

$$\exp(ikS) = \prod_{i=1}^{n} \exp(iks_i), \qquad \textbf{[3.3]}$$

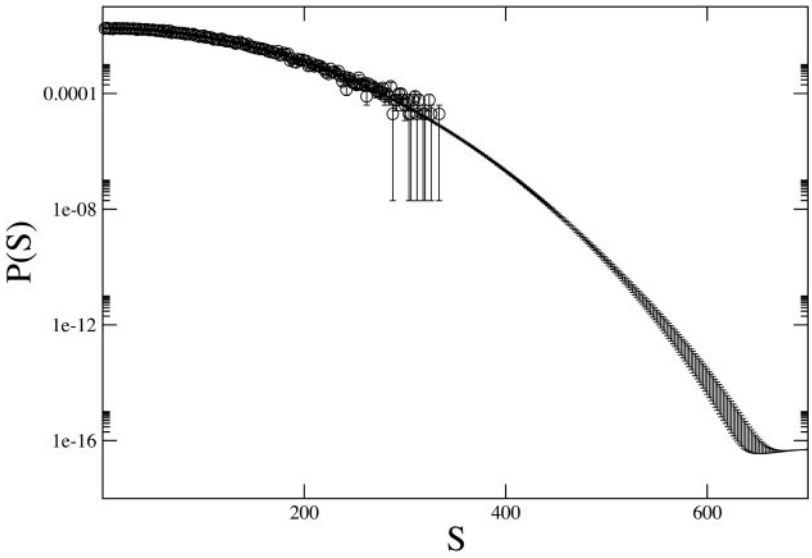

Fig. 2. Probability density $P(S)$ of the total magnetization S of a two-dimensional system of Ising spins on a lattice of 32×32 spins. As an example, we show the results for $J/(k_BT) = 0.3$ (i.e., a supercritical temperature). The total length of both runs was quite short (50,000 cluster moves). The open circles indicate the results obtained through regular (Swendsen–Wang) sampling. The results obtained with the method proposed in the text are indicated only by their error bars. Clearly, the two techniques agree where they can be compared. However, the waste-recycling method makes it possible to extend the calculations to values of $P(S)$ that are 10^{10} times smaller than those that can sampled with conventional means. Eventually [for $P(S) \sim 10^{-16}$], machine precision limits the accuracy of this method.

we can now sum over all 2^n cluster flips to obtain a submove average for the total cluster move m. We denote this submove average by f_m:

$$f_m = \prod_{i=1}^{n} \cos(ks_i). \qquad [3.4]$$

Our estimate for $f(k)$ is then

$$f_{\text{est}}(k) = \frac{1}{M} \sum_{m=1}^{M} f_m, \qquad [3.5]$$

and the corresponding estimate for $P(S)$ follows by Fourier transformation. Note that, in the high-temperature limit, this approach yields the exact $P(S)$ in a single Monte Carlo step.

Fig. 2 compares the resulting estimate for $P(S)$ with the estimate obtained with the conventional approach. Waste-recycling reduces the statistical error in $P(S)$ by >10 orders of magnitude. We stress that the proposed approach improves the sampling of equilibrium states, but the speed at which it approaches equilibrium still depends on the nature of the underlying Monte Carlo trial moves [in the case of Fig. 2, the Swendsen–Wang cluster moves (5)].

The example discussed above may seem rather extreme. However, very similar techniques may be used to speed up other cluster-based algorithms (6) and even normal Monte Carlo simulations. For example, consider a system of particles with intermolecular interactions that can be truncated beyond a finite cut-off distance r_c. If such a system is divided into cells with a diameter larger than r_c, only particles in adjacent cells can interact. It is well known that in such systems, one can perform parallel Monte Carlo simulations if the parallel trial moves are performed in cells that are not adjacent. However, we can do much better with the approach described here. Suppose that we do n trial moves in parallel; then, the number of possible trial configurations is 2^n. With the characteristic-function approach sketched above, we can sample the properties of all of these states simultaneously. Clearly, this sampling holds great promise for parallel simulations; if every cell is attributed to a single processor, we can envisage that the effective sampling speed of the simulation will increase exponentially with the number of processors. Note that this exponential gain applies to the sampling of histograms. This gain is useful because histogram-reweighting techniques are increasingly being used in many applications of Monte Carlo simulations (18). Of course, the accuracy of the histograms that are collected in the manner described above still depends crucially on the efficiency with which the underlying Monte Carlo Markov chain explores configuration space. For example, if the system could explore only a limited set of microstates, then the results of the calculation would be very precise but not accurate.

In conventional Monte Carlo simulations, umbrella-sampling techniques are used often to facilitate the sampling of such histograms (e.g., ref. 4). In this technique, the sampling is biased to obtain good statistics on parts of the histogram that would otherwise not be sampled. The best biasing function is the one that would make the biased histogram appear flat. However, to construct such a weight function, one already needs a good estimate of the unbiased histogram. This technique offers this possibility and, therefore, should greatly enhance the efficiency of multihistogram umbrella-sampling simulations.

It should be stressed that the present algorithm samples microstates, and that it differs fundamentally in philosophy from algorithms that compute the density of macrostates. Refs. 19–21 give a good discussion of various existing approaches. If the aim of the simulation is only to sample the density of macrostates, then these techniques are computationally as efficient as the technique presented here. The crucial difference is that, in the sampling of the density of macrostates, one must first choose an order-parameter to group microstates. Information about individual microstates is irrevocably lost in this procedure. In contrast, the scheme described here samples every microstate explicitly and, therefore, is free of preaveraging.

Last, it is tempting to speculate that the approach described here may be useful for the numerical study of many-fermion systems in which an exponential increase of sampling efficiency with system size is called for.

Superdetailed Balance

This article uses the concept of superdetailed balance to demonstrate the validity of the proposed algorithm. In ref. 4, superdetailed balance is discussed in the context of specific algorithms [CBMC, Recoil-Growth, and the Dynamic Pruned Enriched Rosenbluth Method (DPERM)]. Because the method described in this article is more general than any of the above algorithms, I give a presentation of superdetailed balance that does not use the language of the more specific algorithms used in ref. 4.

Consider a system in a specific microstate o. The first stage in a conventional Monte Carlo move is the construction of one or more trial states. In Metropolis Monte Carlo, there is only one such trial state (n), and the probability to generate it is given by the transition matrix $\alpha_{o\rightarrow n}$. If there are more trial states (denoted by the set $[n_1, n_2, \ldots, n_m] \equiv \{n\}$). The number of trial states is finite, and it is a subset of the (possibly infinite) number of states of the system. We have to specify the probability that we generate a specific set of trial states, given that the initial state is o. In the simplest case, this probability depends only on the coordinates of the states o, n_1, n_2, . . ., n_m. However, often there are many different ways to construct the same set of states. In that case, the probability to generate the set of trial states $\{n\}$ also depends on a set of auxiliary variables $\{b\}$. For example, in the CBMC algorithm for chain molecules, a trial chain is grown segment-by-segment. At every step, k trial segments are generated, but only one is selected as the starting point for the next step. In the case of a CBMC move, the coordinates of these additional trial segments are the auxiliary variables.

The conditional probability to generate a given set of trial states, given that the system is originally in state o is then given by the product of $P(\{b\})$, the probability to generate the set of auxiliary variables $\{b\}$, and $K_{\rm gen}(\{n\}; o, \{b\})$, the conditional probability to generate a set of trial states $\{n\}$, given o and $\{b\}$. The final stage of the move is to make a transition from the initial state o to a specific final state f with a probability $P_{\rm trans}(o \rightarrow f)$.

The condition of superdetailed balance requires that detailed balance is satisfied for the forward and backward transitions between o and f for any specific choice for the sets $\{b\}$ and $o+\{n\}$. To write it explicitly, it is useful the denote the set $o+\{n\}$ also by $f+\{n'\}$, where $\{n'\}$ is the set of that does not include f. Note that $\{n\}$ and $\{n'\}$ may be the same. The superdetailed balance condition then reads:

$$P_B(o)P(\{b\})K_{\rm gen}(\{n\}; o, \{b\})\ P_{\rm trans}(o \rightarrow f)$$
$$= P_B(f)P'(\{b\})K_{\rm gen}(\{n'\}; f, \{b\})\ P_{\rm trans}(f \rightarrow o).$$

Superdetailed balance plays a role whenever there are many possible paths from o to f. The condition says that, for every specific choice of the path (i.e., for every specific choice of the variables $\{b\}$ and the states $\{n'\}$), detailed balance is maintained. If we integrate (or sum) the superdetailed balance condition over all paths between o and f, we retrieve the usual detailed balance condition.

For an arbitrary procedure to generate trial moves, the above prescription does not result in a tractable algorithm. However, in all cases of practical interest, we can define (and compute) a "weight" $w_{\rm i}$ for every possible state i, such that $w_{\rm i}$ is only a function of the coordinates of i and of the auxiliary variables $\{b\}$.

$$P_B(o)P(\{b\})K_{\rm gen}(\{n\}; o, \{b\})w_f(f, \{b\})$$
$$= P_B(f)P(\{P\})K_{\rm gen}(\{n'\}; f,\{b\})w_o(o, \{b\})$$

For instance, in CBMC, the w values are the Rosenbluth weights of the initial and final states. From these weights, we can then construct transition probabilities that satisfy superdetailed balance, e.g.:

$$P_{\rm trans}(o \rightarrow f) = \min\{1, w_f/w_o\}.$$

Another possibility would be:

$$P_{\rm trans}(o \rightarrow f) = \frac{w_f}{\sum_i w_i}.$$

This criterion is used to fix the acceptance probability of a Monte Carlo trial move. The explicit expression for the acceptance rule depends on the nature of the algorithm.

In the main text, we use the term "linked microstates" to denote a set of microstates that have been constructed according to the procedure described above. In this set, every individual microstate i has a weight w_i. If we perform Monte Carlo sampling within such a linked set, then every state will be visited with a probability $w_i/\sum_j w_j$. This result was used in an earlier part of the article.

I thank Ivan Coluzza, Simon Tindemans, Bela Mulder, Harm-Geert Muller, and, in particular, Georgios Boulougouris, for critical comments. I also thank Kurt Binder and numerous other colleagues for helpful suggestions. The FOM Institute is part of the research program of the Stichting voor Fundamenteel Onderzoek der Materie, which is supported by the Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO).

1. Metropolis, N., Rosenbluth, A. Rosenbluth, M., Teller, A. & Teller, E. (1953) *J. Chem. Phys.* **21,** 1087–1092.
2. See e.g.: Newman, M. E. J. & Barkema, G. T. (1999) *Monte Carlo Methods in Statistical Physics* (Clarendon, Oxford).
3. Landau, D. P. & Binder, K. (2000) *Guide to Monte Carlo Simulations in Statistical Physics* (Cambridge Univ. Press, Cambridge, U.K.).
4. Frenkel, D. & Smit, B. (2002) *Understanding Molecular Simulations: From Algorithms to Applications* (Academic, San Diego), 2nd Ed.
5. Swendsen, R. H. & Wang, J. S. (1987) *Phys. Rev. Lett.* **58,** 86–88.
6. Liu, J. & Luijten, E. (2004) *Phys. Rev. Lett.* **92,** 035504.
7. Harris, J. & Rice, S. A. (1988) *J. Chem. Phys.* **88,** 1298–1306.
8. Siepmann, J. I. & Frenkel, D. (1992) *Mol. Phys.* **75,** 59–70.
9. Frenkel, D., Mooij, G. C. A. M. & Smit, B. (1992) *J. Phys. Condens. Matter* **4,** 3053–3076.
10. De Pablo, J. J., Laso, M. & Suter U. W. (1992) *J. Chem. Phys.* **96,** 2395–2403.
11. Combe, N. Vlugt, T. J. H., ten Wolde, P. R. & Frenkel, D. (2003) *Mol. Phys.* **101,** 1675–1682.
12. Consta, S., Vlugt, T. J. H., Wichers Hoeth, J., Smit, B. & Frenkel, D. (1999) *Mol. Phys.* **97,** 1243–1254.
13. Consta, S., Wilding, N. B., Frenkel, D. & Alexandrowicz, Z. (1999) *J. Chem. Phys.* **110,** 3220–3228.
14. Baschnagel, J., Wittmer. J. P. & Meyer, H. (2004) in *Computational Soft Matter: From Synthetic Polymers to Proteins*, eds. Attig, N., Binder, K., Grubmüller, H. & Kremer, K. (John von Neumann-Institut für Computing, Jülich, Germany), Vol. 23, pp. 83–140.
15. Manousiouthakis, V. I. & Deem, M. W. (1999) *J. Chem. Phys.* **110,** 2753–2758.
16. Barker, A.A. (1965) *Aust. J. Phys.* **18,** 119–133.
17. Fortuin, C. M. & Kasteleyn, P. W. (1972) *Physica* **57,** 536–564.
18. de Pablo, J. J., Yan, Q. L. & Escobedo, F. A. (1999) *Annu. Rev. Phys. Chem.*, **50,** 377–411.
19. Schulz, B. J., Binder, K. & Müller, M. (2002) *Int. J. Mod. Phys. C* **13,** 477–494.
20. Scott Shell, M., Debenedetti, P. G. & Panagiotopoulos, A. Z. (2003) *J. Chem. Phys.* **119,** 9406–9411.
21. Wang, J. S. & Swendsen, R. H. (2002) *J. Stat. Phys.* **106,** 245–285.