PHYSICAL REVIEW E VOLUME 51, NUMBER 2 FEBRUARY 1995

Parallel Monte Carlo simulations

K. Esselink, L.D.J.C. Loyens, and B. Smit
Shell Research B.V., Koninklijke/Shell-Laboratorium, Amsterdam, P.O. Box 38000, 1030 BN Amsterdam, The Netherlands
(Received 23 February 1994)

The Monte Carlo (MC) method is an important tool in sampling the state space of a chosen statistical ensemble. It allows the study of thermodynamic averages of configurational properties by generating "moves" in a system and accepting or rejecting the thus generated new state depending on the energy of the new system and/or a random choice. These moves are intrinsically sequential and complicate parallel implementation. We propose a method which allows the parallel generation of MC moves, and which is especially useful for simulations with unavoidably low acceptance rates, such as for long chain molecules.

PACS number(s): 02.70.Lq, 83.80.Pc

I. INTRODUCTION

At first sight, the title of this paper "Parallel Monte Carlo simulations" appears to be a contradiction in terms, since Monte Carlo (MC) is known to be an intrinsically sequential process. It turns out that this apparent fundamental impossibility can be circumvented by introducing an algorithm that allows for straightforward parallelization.

The Monte Carlo method was applied for the first time by Metropolis *et al.* for equation of state calculations [1]. The method studies thermodynamic averages of configurational properties by generating new states of a system using certain moves from an old state. If the potential energy U of the new state is lower than that of the old state, the new state is "accepted." If not, the new state is accepted with a probability $\exp(-\Delta U/k_BT)$. Averaging over the properties of the accepted states, and taking the acceptance ratios into account, provides the wanted information.

A very common "move" is the random displacement of a random particle over a certain distance in the x, y and z dimension. If this displacement is small, the new state will hardly differ from the old state, yielding a high probability of acceptance, at the cost of poor sampling of the state space. Large moves suffer from the opposite: good sampling, but low acceptance rates. For chain molecules, the standard MC method becomes rapidly more prohibitive for larger chain length, in particular for the liquid phase. The chance of acceptance for a new state after moving a complete chain is almost zero. This problem can be solved by "growing" the chains judiciously, i.e., in the direction of a favorable energy state. The bias thus introduced can be taken into account properly, which is the essence of the configurational-bias MC (CBMC) method [2–4]. A recent application can be found in the study of critical properties of n alkanes [5], where the combination of CBMC and the Gibbs-ensemble method allows the study of the vapor-liquid curve of alkanes up to C_{48}. However, longer chains do imply lower acceptance rates, with a concomitant need for more computing power.

At this point, application of parallel computers can constitute a solution. They have been applied successfully in many other fields, such as in molecular dynamics (MD) calculations [6,7]. An important ingredient for efficient parallelization is the availability of many portions of independent work. For MD, these portions can be identified in more than one way, the efficiency depending on the system under study [8–10]. Monte Carlo appears to be intrinsically sequential, since the generation of a new move depends on the previously accepted state and cannot be performed independently in a straightforward way. In this work, we present a Monte Carlo algorithm in which new moves are generated throughout the system and then combined. The bias introduced is correctly taken into account by a method corresponding closely to the method of CBMC. We believe that the method allows efficient parallel implementation, and exemplify this with results from a sequential study. First, we explain some of the parallel techniques proposed in the literature.

II. CONVENTIONAL PARALLEL ALGORITHMS

Various parallel Monte Carlo algorithms have been proposed in the literature. One can make the following classification: (1) Parallel moves in independent regions; (2) hybrid Monte Carlo method; (3) task farming; and (4) parallel energy calculation. Parallel moves in independent regions was one of the first references to parallel MC [11]. It provides an example of the use of an ultrashort range potential: on a regular lattice particles ("spins") only interact with their six nearest neighbors. This makes it straightforward to identify well-separated regions: when placing the spins on a checkerboard, all particles on the same color can be updated simultaneously. For continuum systems, a similar approach can be used for models with sufficiently short ranged potentials, such that a change in the location of a particle in one region does not have an effect on the energy of any

1063-651X/95/51(2)/1560(9)/$06.00

particle in another region. A restriction is that the generated moves and the range of interactions should not cover large distances. Therefore, chain molecules and systems with long range forces are excluded from this approach.

Another approach is the use of the hybrid Monte Carlo (HMC) method [12]. In this method velocities are drawn from a Gaussian distribution and subsequently used for solving the equations of motion. In this respect, the method is very similar to molecular dynamics and all parallel algorithms developed for MD can be applied straightforwardly. A disadvantage of HMC, in contrast to MD, is that no information on time dependent properties can be obtained. In addition, the time steps used in both techniques are similar, therefore in itself HMC offers no particular advantage over MD with a Nosé-Hoover thermostat [13,14].

A rather coarse form of parallelism is farming of independent tasks. It is used in [15], where the MC consists of many independent high energy physics computations, and the independence is used for parallelization. In [16], parallelism is introduced by performing calculations with different scales of length and energy parameters, which provides valuable information about cancellation of random sampling errors. The parallelism is thus introduced at the level of complete simulations, not as a means to speed up one single run. Therefore, it does not scale very well. In [17], MC is used to simulate the history of many particles traveling through a certain geometry for performing a radiation transport analysis. In order to sample correctly, many particles need to be traced, but the individual particles are completely independent from each other. Parallelization is straightforward. A similar approach is followed in [18,19]. However, this form of parallelism has no close analog in MC simulations with interacting particles.

In [20], several types of parallelization of the Metropolis method are discussed. In the first type, the energy of a trial conformation is computed in parallel, while a master processor handles the actual moves. This appeared to be rather inefficient due to a low "computation to communication ratio." An improvement was made by the use of a "systolic loop," where each particle is displaced in turn, and all processors continually compute energies between displaced particles and their own. Communications are arranged such, that idle time is minimized. In another approach discussed in [20], the simulation of S trial moves is done by running Q independent batches of S/Q moves, the independent batches simply use different random number seeds (actually, a form of farming). The main disadvantage is the need for (sequential) equilibration. For large systems, equilibration can be a significant part of the simulation. Their conclusion was that the choice of method should depend on the number of processors available.

In summary, we find that the literature does not describe a method in which parallelism is used efficiently to decrease the turnaround time of a single simulation. The sequential nature of generating moves one after another in the Metropolis Monte Carlo algorithm indeed seems difficult to parallelize. In the next section, we will show how this problem can be solved.

III. THE PARALLEL MONTE CARLO TECHNIQUE

In the following we propose an algorithm to perform Monte Carlo moves in parallel, which in Sec. IV will be tested in the simulation of alkanes in zeolites. Note that, although we introduce the method for chain molecules, it is not restricted to this case.

A. Basic principles

Repeatedly, the following steps are executed.

(1) In choosing a new chain, g trial conformations of the chain are generated simultaneously using CBMC. For each of the chain conformations i, the Rosenbluth weight is calculated:

$$W(i) = \exp(-\beta u_1) W^*(i) = \exp(-\beta u_1) \prod_{l=2}^{M} \sum_{j=1}^{k} \exp[-\beta u_l(j)] , \quad (1)$$

where u_l is the energy of atom l of chain i with M atoms, the sum is over the k trial orientations used in the CBMC scheme. Note that the probability that a chain in conformation i is generated is [2]

$$p_c(i) = \frac{\exp[-\beta U(i)]}{W(i)} , \quad (2)$$

where $U(i) = \sum_{l=1}^{M} u_l$ is the total energy of the chain in conformation i.

(2) We define

$$Z = \sum_{i=1}^{g} W(i) , \quad (3)$$

and out of the g conformations, we select one chain n, where each chain i has the probability

$$p_p(i) = \frac{W(i)}{Z} . \quad (4)$$

Furthermore, we define

$$R = Z - W(n) . \quad (5)$$

(3) Out of the old configuration, one chain o is selected at random and calculated is

$$Z' = W(o) + R , \quad (6)$$

where Z' differs from Z in that $W(n)$ is replaced by $W(o)$. The Rosenbluth factor of the old conformation is defined by

$$W(o) = \exp[-\beta u_1(o)] W^*(o) . \quad (7)$$

Note that in the calculation of the Rosenbluth factor of the atoms of the old conformation one of the k trial orientations is the actual position of the atom.

(4) The move of the selected chain o to the new conformation n [which has been selected in step (2)] is accepted with a probability

$$\mathcal{P}_{\rm acc}(o \rightarrow n) = \min(1, Z/Z') . \quad (8)$$

In this algorithm, g trial conformations are generated in parallel. Out of these, the most favorable conformation has the highest probability of being selected. This introduces a bias in the sampling scheme. In the Appendix, it is proven that the bias is removed by the use of acceptance rule (8).

B. Placing the first atom in parallel

Like in the original CBMC scheme, the growing of a chain only proceeds if the first particle is placed successfully. In very low-density systems this will, in general, be no problem, but in many other cases this success rate can be quite low. In a parallel implementation where all processors grow chains simultaneously, this can imply a substantial load imbalance with respect to the distribution of work, since some processors could stop growing chains after unsuccessfully placing the the first atom. It would, therefore, be better to increase the chance of successfully placing this particle *a priori*. Interestingly, this can be done in the same spirit as the algorithm presented above, by placing many "first particles" and choosing the most favorable. We modify the algorithm as follows.

(1) For each chain i, we generate f random positions for placing the first atom, and we calculate for each of these the energy $u_1(m_i)$.

(2) For each chain i, we define

$$w_1(i) = \sum_{m_i=1}^{f} \exp[-\beta u_1(m_i)] , \quad (9)$$

and out of the f "trial firsts," we select one atom h_i, where each atom m_i has the probability

$$p_f(m_i) = \frac{\exp[-\beta u_1(m_i)]}{w_1(i)} . \quad (10)$$

(3) The selected first atom h_i is used to grow the rest of the chain of M atoms using the configurational-bias Monte Carlo algorithm. The probability that a conformation is grown is given by

$$p_c(i) = \frac{\sum_{l=2}^{M} \exp[-\beta u_l(i)]}{\prod_{l=2}^{M} w_l(i)} = \frac{\sum_{l=2}^{M} \exp[-\beta u_l(i)]}{W^*(i)} , \quad (11)$$

where $w_l(i)$ is the Rosenbluth factor for atom l. For each of the chains, we calculate

$$W(i) = w_1(i) W^*(i) . \quad (12)$$

(4) We define

$$Z = \sum_{i=1}^{g} W(i) , \quad (13)$$

and out of the g conformations we select one chain n, where each chain i has the probability

$$p_p(i) = \frac{W(i)}{Z} . \quad (14)$$

Furthermore, we define

$$R = Z - W(n) , \quad (15)$$

and

$$r = w_1(n) - \exp[-\beta u_1(h_n)] . \quad (16)$$

Note that h_n is the first atom of the selected chain n.

(5) Out of the old configuration, one chain o is selected at random and we calculate

$$Z' = W(o) + R , \quad (17)$$

where the Rosenbluth factor of the old conformation is

$$W(o) = w_1(o) W^*(o) = (\exp[-\beta u_1(o)] + r) W^*(o) , \quad (18)$$

in which $u_1(o)$ is the energy of the first atom of the old configuration.

(6) The move is accepted with the probability

$$\mathcal{P}_{\rm acc}(o \rightarrow n) = \min\left(1, \frac{Z}{Z'}\right) . \quad (19)$$

In the Appendix, it is proven that this algorithm samples the correct distribution.

IV. APPLICATION OF THE PARALLEL MONTE CARLO TECHNIQUE

In the previous section, the principles of the parallel Monte Carlo technique have been described. An important question is whether this approach, in a practical parallel application, indeed results in the desired increase of performance. To test this, we have performed some sequential simulations (parallel simulations will be presented in a later paper). As a test system, we have chosen alkanes adsorbed in zeolites. Zeolites are microporous crystalline materials, and their pores are accessible to alkanes. Since zeolites are used as catalytic materials in petrochemical applications, it is of interest to have information on the behavior of alkanes adsorbed in zeolites. Simulations appear to be ideally suited to obtain this information [21]. Smit and Siepmann [22] have used the CBMC technique to study the adsorption of n alkanes in the zeolite silicalite. The simulations reported in [22] consist of the following four steps: random displacement of a molecule, random rotation of a molecule, regrowing of part of the molecule, and insertion of a molecule at a random position. The first three MC moves are introduced to change the (local) conformation of the molecules. If one would restrict the algorithm to only such types of moves, the method would be equivalent to molecular dynamics in efficiency. From a parallel computing point of view, hybrid Monte Carlo [12] pro-

vides a good alternative for the first three steps. The last MC step, the insertion at a random position, results in a very efficient sampling, since large jumps in phase space are made. Below we focus on the parallel computing aspects of this step using the algorithm as developed in the previous section. Details on the zeolite structure and model can be found in [22]. Note that the insertion of a molecule is also one of the time consuming steps in grand-canonical and Gibbs-ensemble simulations [23]. The application of our parallel Monte Carlo technique, is therefore, by no means limited to zeolites.

A. Sequential implementation

We have realized a sequential implementation of the method. The average energies of pentane in silicalite for varying g (number of chains grown in parallel) and f (number of "first" atoms for each chain) are given in Table I. It is important to note that, within the accuracy of the data, the results are indeed *independent* of these parameters. Furthermore, the results show that the percentage of accepted moves increases significantly with increasing g and f. A full analysis of the performance is given in Sec. IV B.

It is instructive to consider a simulation in which the bias is not removed in the acceptance rule. We take the naive approach that the correction is small and that we can select the trial conformation using Eq. (4), but we use instead of Eq. (8) acceptance rule

$$\mathcal{P}_{\rm acc}(o \to n) = \min\left(1, W(n)/W(o)\right). \qquad (20)$$

The results of these simulations are presented in Fig. 1, where the average energies as calculated by the correct scheme are compared with the ones obtained with the incorrect scheme. In contrast to the correct method, the results are not independent of the number of chains grown in parallel and show a systematic drift.

TABLE I. The energies and acceptance probability of pentane in silicalite as a function of g (the number of chains grown in parallel) and f (the number of trials for the first atom). $N_{\rm MC}$ is the total number of Monte Carlo moves (the insertion of a molecule at a random position), $\langle U\rangle_p$ is the average total energy of the pentane molecule, $\langle U\rangle_z$ is the average pentane-zeolite energy, and "Acc." is the percentage of accepted moves. The subscript gives an estimate of the accuracy of the results, so 5.36_3 means 5.36 ± 0.03. τ equals the total execution time (CPU seconds) of the run on an IBM RS6000/560H workstation.

g	f	$N_{\rm MC}$	$\langle U\rangle_p$	$\langle U\rangle_z$	Acc. (%)	τ
800	1	4000	-5919_{18}	-6787_{13}	81.8	12473
400	1	4000	-5925_{20}	-6786_{15}	71.4	6231
160	1	10000	-5914_{16}	-6791_{11}	52.0	6412
80	1	20000	-5941_{17}	-6807_{12}	34.8	6610
40	1	40000	-5954_{17}	-6812_{12}	21.1	7142
20	1	80000	-5906_{16}	-6778_{12}	12.5	8089
10	1	80000	-5957_{20}	-6807_{14}	6.2	4470
5	1	160000	-5910_{19}	-6782_{14}	3.4	4923
2	1	400000	-5918_{19}	-6780_{12}	1.4	5238
1	1	800000	-5936_{20}	-6789_{17}	0.7	5393
5	200	16000	-5942_{17}	-6803_{11}	36.7	6572
5	100	16000	-5940_{13}	-6798_{9}	36.9	4524
5	40	16000	-5918_{15}	-6780_{11}	34.6	3424
5	20	16000	-5931_{19}	-6795_{14}	28.7	2931
5	15	16000	-5923_{19}	-6789_{14}	26.2	2681
5	10	16000	-5921_{22}	-6781_{15}	20.5	2307
5	5	16000	-5957_{29}	-6795_{19}	13.1	1619
5	2	16000	-5945_{39}	-6764_{26}	6.3	867
5	1	16000	-5924_{48}	-6839_{33}	3.1	495

B. Sequential performance

The proposed algorithm has three parameters which influence the performance. The first one (introduced by CBMC) is k, which determines the number of trial orientations probed for each atom (but the first) of the chain. Optimum values for this parameter have been studied in detail by Mooij [24]. We have introduced two new parameters: the number of chains grown in parallel (g) and the number of probed first positions (f). Fixing all variables to one yields the original random insertion method, which for chain molecules is totally infeasible. However, using values that are too high wastes computer time on probes which will not be used anyway. Below, we derive expressions from which the optimum values can be obtained.

In these simulations, the configurations generated (and accepted) are uncorrelated. For this reason, the average time τ_a needed to accept a configuration as a function of f and g for a given system is a good measure for the performance of the algorithm. In Fig. 2, we plotted this mean time between acceptance for methane. In this sequential implementation, we see that for small values of g, τ_a does not increase. This suggests that probing ten methane molecules *in parallel* indeed speeds up computa-

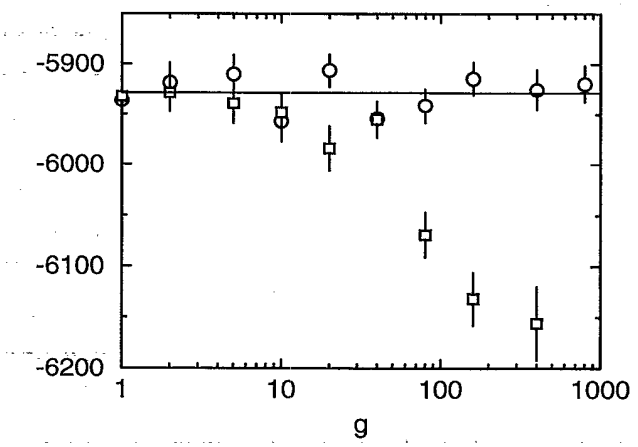

FIG. 1. Comparison of the total energy of pentane in silicalite as calculated from the sampling without the correction for the bias (□) with the correct sampling scheme (○). g is the number of chains grown in parallel and $f = 1$. The horizontal line is the average energy as calculated from the correct results.

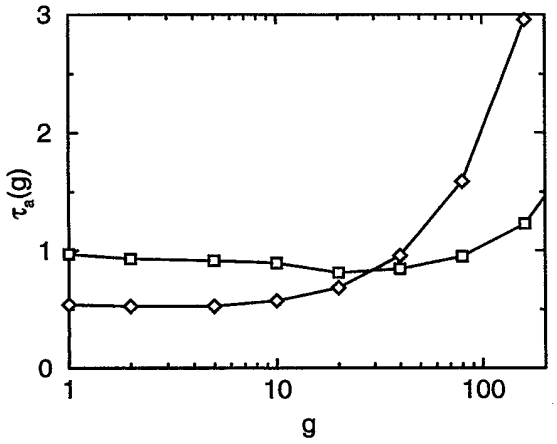

FIG. 2. Average time to acceptance $\tau_a(g)$ for methane ($\diamond$) and pentane ($\square$) in silicalite for varying number of molecules g placed in parallel. The data are taken from Tables I and II.

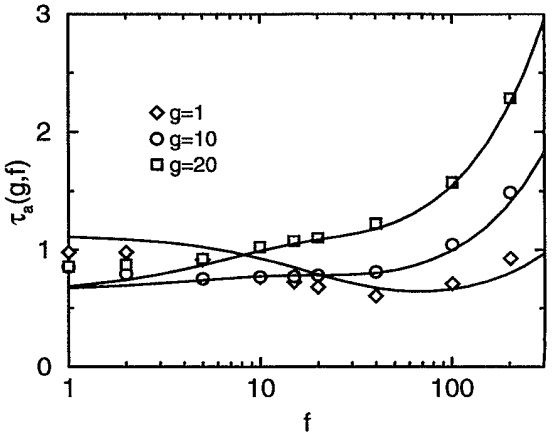

FIG. 3. Average time to acceptance $\tau_a(g,f)$ for pentane in silicalite for a fixed number of chains grown in parallel g as a function of the number of first atoms probed f. Time in seconds for simulations on an IBM RS6000/560H. The symbols are results from simulations and the lines are the predictions of Eq. (21).

tion with a factor of 10. For much larger values of g, the probability that several acceptable configurations are being generated increases, yet only one new configuration can be accepted per step. For example, for $g = 160$, $\tau_a = 2.96$, which is 5.5 times more than $\tau_a = 0.538$ valid for $g = 1$. A parallel implementation can, therefore, at best reach a speedup of 29 ($= 160/5.5$).

For pentane, Fig. 2 shows that $\tau_a(g)$ decreases somewhat with increasing g, so surprisingly this parallel approach is also more efficient for a sequential implementation. The reason for this is that in the parallel scheme, the "optimal" configuration is selected and only for this configuration is the Rosenbluth factor of the old configuration [Eq. (7)] calculated and the acceptance test performed. In the original CBMC version, not only the optimal configuration is tested for acceptance, but also configurations which have a very low probability of acceptance. For each of these configurations the Rosenbluth factor of the old configuration has to be calculated. For this particular system the Rosenbluth factor of the old configuration could be stored and used until a move is accepted. However, normally other MC moves would change the conformation of the molecule or surrounding molecules, necessitating recalculation of the Rosenbluth factor each step.

For pentane, the influence of increasing the number of first atoms placed f on the average time to acceptance τ_a for various values of g is shown in Fig. 3. For large values of g, the probability of generating an acceptable configuration is high, and, therefore, there is no need for increasing f. Nevertheless, for small values of g, increasing f has a significant effect and reduces the CPU time by 30%.

For practical applications, one would like to use the optimum values of g and f, i.e., those which give maximum probability of acceptance for a minimum amount of CPU time. Therefore, we have to find the minimum of

$$\tau_a(g,f) = \tau_m(g,f)/P(g,f)\,, \tag{21}$$

in which $\tau_m(g,f)$ denotes the CPU time needed for one MC move, and $P(g,f)$ denotes the probability of acceptance of a move.

We assume the following form for τ_m,

$$\tau_m(g,f) = c_0 gf + c_1 gF(f) + c_1\,. \tag{22}$$

This form is based on a term for probing gf "first" particles and a term for growing g new chains and one old chain. A chain is only grown if the first atom is successfully placed. This success rate is given by the function $F(f)$,

$$F(f) = 1 - \chi^f\,, \tag{23}$$

where χ is the probability of overlap of a "first" particle with one of the zeolite atoms. For this particular system, we found Eq. (23) to be accurate within one percent with $\chi = 0.87$. Equation (22) with $c_0 = 0.000\,28$ and $c_1 = 0.028$ describes the observed timings with an accuracy of ten percent, for the smallest values of g and f the accuracy is less.

Next, we have to derive an expression of the acceptance probability as a function of g and f. For fixed f, we can use the following approximation. From acceptance rule (19) and equations (15) and (16), it follows that

$$\begin{aligned} P_f(g) &= \left\langle \frac{Z}{Z'} \right\rangle = \left\langle \frac{W(n)+R}{W(o)+R} \right\rangle \\ &\approx \frac{g\,\langle W\rangle}{\langle W(o)\rangle + (g-1)\,\langle W\rangle} \\ &= \frac{gP_f(1)}{1+(g-1)P_f(1)}\,, \end{aligned} \tag{24}$$

where $\langle W\rangle$ denotes the average Rosenbluth factor of a single chain and $\langle W(o)\rangle$ denotes the average Rosenbluth factor of the old configuration. If $g = 1$, we derive the following expression for acceptance:

$$P_{g=1}(f) = \left\langle \frac{W(n)}{W(o)} \right\rangle = \left\langle \frac{w_1(n)W^*(n)}{w_1(o)W^*(o)} \right\rangle \approx \frac{\langle w_1(n)\rangle\langle W^*(n)\rangle}{\langle w_1(o)\rangle\langle W^*(o)\rangle} \approx \frac{f\,\langle \exp[-\beta u_1(m)]\rangle\,\langle W^*(n)\rangle}{(\langle \exp[-\beta u_1(o)]\rangle + (f-1)\,\langle \exp[-\beta u_1(m)]\rangle)\,\langle W^*(o)\rangle}. \quad (25)$$

We can define

$$P_{g=1}(1) \approx \frac{\langle \exp[-\beta u_1(m)]\rangle\,\langle W^*(n)\rangle}{\langle \exp[-\beta u_1(o)]\rangle\,\langle W^*(o)\rangle} \quad (26)$$

and

$$\omega = \frac{\langle \exp[-\beta u_1(m)]\rangle}{\langle \exp[-\beta u_1(o)]\rangle}. \quad (27)$$

Note that ω can be assumed to be equal to the acceptance of methane in the same silicalite with $f = 1, g = 1$, or can be calculated directly from the simulation. In our case, $\omega = 0.044$. Substitution of (26) and (27) in (25) yields

$$P_{g=1}(f) \approx \frac{fP_{g=1}(1)}{1 + (f-1)\omega}. \quad (28)$$

Given g and f, $P_f(1) = P_{g=1}(f)$ is computed using (28) after which $P_f(g)$ yields the required acceptance ratio $P(g, f)$ using (24). We found the accuracy of this prediction of $P(g, f)$ to be within 25%.

The solid lines in Fig. 3 represent the prediction of $\tau_a(g, f)$, which agree well with the times obtained from the simulations. The optimum values for g and f are found by minimizing $\tau_a(g, f)$. It follows that $g = 2$ and $f = 53$ (while k was set to 6), resulting in a gain of 40% as compared to the conventional approach ($g = f = 1$).

C. Estimate of parallel performance

The *possibility* to grow chains in parallel does not necessarily imply that actually doing so results in increased performance. An important aspect is the load balance, which in this case is determined for a great part by the success (overlap-nonoverlap) of placing the first particle. We introduced the parameter f to increase this chance of success, but if every processor grows a small number of chains (s), there is still a chance of substantial load imbalance. Usually, increasing the amount of work per processor helps, but here the effect of this increase on the performance $P(g, f)$ and, therefore, on $\tau_a(g, f)$ (where g equals the number of processors Q times s) has to be taken into account. Using equation (22) of the previous section, we can estimate the speedup as follows. Recall that in the sequential case, the amount of work scales linearly with the expected value of $gF(f)$. On a processor network of Q processors, where every processor tries to grow s chains ($g = sQ$), the total execution time is determined by the processor with the most work, therefore, the term $gF(f)$ has to be replaced by the expected maximum over Q stochastic values $sF(f)$. Since these Q events are uncorrelated, we can write,

$$E \max_Q\{sF(f)\} = s + 1 - \sum_{k=0}^{s} [C(F(f), k, s)]^Q, \quad (29)$$

$$C(f, k, s) = \sum_{i=0}^{k} \binom{s}{i} f^i (1-f)^{s-i}. \quad (30)$$

Equation (29) is proven in the Appendix. This implies that the expected total amount of time $\tau_q(s, f, Q)$ needed for growing s chains per processor on a Q-processor network becomes [see Eq. (22)]

$$\tau_q(s, f, Q) = c_0 sf + c_1(E \max_Q\{sF(f)\}) + c_1, \quad (31)$$

and the resulting prediction for average time to accept a new configuration becomes

$$\tau_r(s, f, Q) = \tau_q(s, f, Q)/P(sQ, f). \quad (32)$$

For a given processor network size Q, we can determine the minimum value of $\tau_r(s, f, Q)$, given by, say, $\tau_r(s_Q, f_Q, Q)$. The expected speedup $S(Q)$ then becomes

$$S(Q) = \tau_r(s_1, f_1, 1)/\tau_r(s_Q, f_Q, Q). \quad (33)$$

For two alkanes, the expected speedup is shown in Fig. 4. We see that for the pentane simulations, the speedup is slightly more than 5 using 20 processors. For larger chains, the expected speedup becomes better, as is demonstrated by the results for dodecane.

TABLE II. The energies and acceptance probability of methane in silicalite as a function of g, $f = 1$. See also the caption of Table I. The runs are performed on a Silicon Graphics Personal Iris workstation.

g	N_{MC}	$\langle U\rangle_p$	$\langle U\rangle_z$	Acc. (%)	τ
400	8000	-1358_3	-1358_3	98.175	55794
160	20000	-1360_2	-1360_2	95.175	56304
80	40000	-1357_1	-1357_1	89.237	56814
40	80000	-1359_1	-1359_1	75.925	58200
20	160000	-1359_1	-1359_1	55.212	60396
10	320000	-1361_1	-1361_1	34.656	63360
5	640000	-1359_1	-1359_1	19.702	66156
2	1600000	-1360_1	-1360_1	8.465	70860
1	3200000	-1360_1	-1360_1	4.363	75138

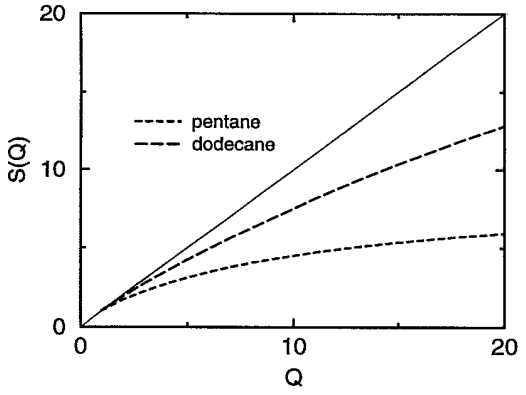

FIG. 4. Speedup $S(Q)$ as a function of network size Q, as predicted by Eq. (33). The solid line represents optimum speedup.

V. CONCLUDING REMARKS

In this work, we have generalized the metropolis Monte Carlo algorithm to generate moves in parallel and select the most favorable with the highest possibility. It is proven that the resulting bias is removed by appropriate acceptance rules. The resulting algorithm allows for straightforward parallelization.

We have demonstrated the feasibility of the method by simulations of alkanes adsorbed in zeolites. These tests show that the method is particularly useful at conditions where conventional moves are unlikely to be accepted. It is important to note that precisely under these conditions, simulations tend to take a large amount of CPU time, and parallel computing may help to reduce this. Surprisingly, the parallel approach also reduces computation time on a sequential computer. Depending on the application, we found a gain of 40%.

This algorithm provides a way to perform any Monte Carlo move in parallel. However, not every type of move is expected to be efficient. For example, random (small) displacement can be performed much more efficiently in parallel with the hybrid Monte Carlo method, which moves all molecules at the same time. In the general case, the most efficient approach is a combination of various Monte Carlo moves, for instance, using the hybrid Monte Carlo method for local changes in conformations and the present method for random insertions.

APPENDIX A: PROOF OF CORRECTNESS OF SAMPLING SCHEMES

In this Appendix, it is proven that the algorithms introduced in this work generate configurations that are distributed according to a Boltzmann distribution, i.e., the probability of finding conformation i is given by

$$\mathcal{N}(i) \propto \exp[-\beta U(i)], \tag{A1}$$

where $U(i)$ is the total energy of configuration i.

1. Algorithm of Sec. III A

The standard technique of proof that a Markov process samples the correct distribution is to show that detailed balance is obeyed, i.e., the flow of configurations going from o to n should equal the reverse,

$$K(o \to n) = K(n \to o). \tag{A2}$$

The flow from o to n is the product of the probability of being in state o, the probability of generating state n, and the probability of acceptance $\mathcal{P}_{\rm acc}$:

$$K(o \to n) = \mathcal{N}(o) p(o \to n) \mathcal{P}_{\rm acc}(o \to n). \tag{A3}$$

Note that a configuration n or o can be generated in an infinite number of ways, see for example Fig. 5. Let us denote a set of g trial conformations by

$$\{\mathfrak{b}\}_g = (\mathfrak{b}_1, \mathfrak{b}_2, \ldots, \mathfrak{b}_g). \tag{A4}$$

The set of all sets $\{\mathfrak{b}\}_g$ which include conformation n is denoted by

$$\{\tilde{b}_n\} \equiv \{\{\mathfrak{b}\}_g | \mathfrak{b}_n \in \{\mathfrak{b}\}_g\}. \tag{A5}$$

Every element $\{\mathfrak{b}\}_g$ of $\{\tilde{b}_n\}$ can be written as $(\mathfrak{b}_n, b^*)$, which defines b^*. For the detailed balance condition, we have to sum over all sets in $\{\tilde{b}_n\}$ that generate conformation n,

$$K(o \to n) = \mathcal{N}(o) \sum_{i \in \{\tilde{b}_n\}} p(o \to n|i) \mathcal{P}_{\rm acc}(o \to n|i). \tag{A6}$$

Note that the probability of generating this conformation and its acceptance depend on the particular set of trial conformations.

The probability of generating state n is the product of generating a chain in this state (2) and the probability

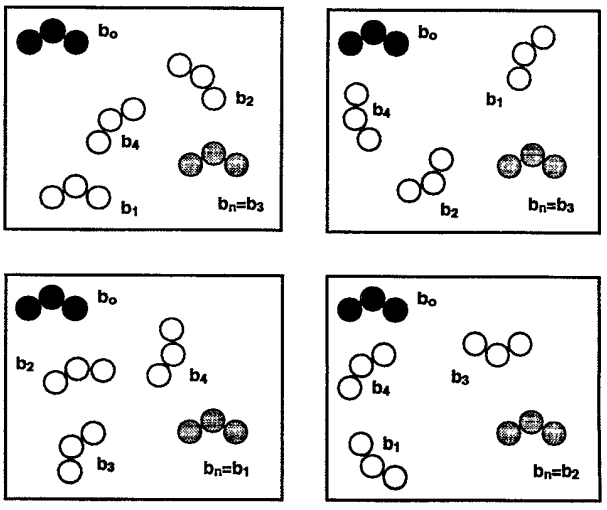

FIG. 5. Four different sets of trial conformations $\{\mathfrak{b}\}_k$ with $k = 4$ that each can lead to a move from configuration $\mathfrak{b}_o$ to $\mathfrak{b}_n$.

that this conformation is selected (4),

$$p(o \to n|(\mathfrak{b}_n, b^*)) = p_c(\mathfrak{b}_n)p_p((\mathfrak{b}_n, b^*)) = \frac{\exp[-\beta U(\mathfrak{b}_n)]}{W(\mathfrak{b}_n)} \frac{W(\mathfrak{b}_n)}{Z((\mathfrak{b}_n, b^*))} = \frac{\exp[-\beta U(\mathfrak{b}_n)]}{Z((\mathfrak{b}_n, b^*))} = \frac{\exp[-\beta U(\mathfrak{b}_n)]}{W(\mathfrak{b}_n) + R(b^*)} . \tag{A7}$$

In the last equality, we have used the fact the R does not depend on the selected orientation. Similarly, for the reverse move, we can write

$$p(n \to o|(\mathfrak{b}_o, b^*)) = \frac{\exp[-\beta U(\mathfrak{b}_o)]}{Z'((\mathfrak{b}_o, b^*))} = \frac{\exp[-\beta U(\mathfrak{b}_o)]}{W(\mathfrak{b}_o) + R(b^*)} . \tag{A8}$$

Substitutions of equations (A1), (A6), (A7), and (A8) in the equation for detailed balance (A2) gives

$$\sum_{i \in \{\tilde{\mathfrak{b}}_n\}} \frac{\mathcal{P}_{\rm acc}(o \to n|i)}{Z(i)} = \sum_{j \in \{\tilde{\mathfrak{b}}_o\}} \frac{\mathcal{P}_{\rm acc}(n \to o|j)}{Z'(j)} ,$$

$$\sum_{i \in \{\tilde{\mathfrak{b}}_n\}} \frac{\mathcal{P}_{\rm acc}(o \to n|i)}{W(\mathfrak{b}_n) + R(b_i^*)} = \sum_{j \in \{\tilde{\mathfrak{b}}_o\}} \frac{\mathcal{P}_{\rm acc}(n \to o|j)}{W(\mathfrak{b}_o) + R(b_j^*)} . \tag{A9}$$

Note that for each element of the sum on the left-hand side, there is a corresponding element on the right-hand side with the same b^*. The above equality is certainly obeyed if each of these two corresponding elements are equal, viz.

$$\frac{\mathcal{P}_{\rm acc}(o \to n|b^*)}{W(\mathfrak{b}_n) + R(b^*)} = \frac{\mathcal{P}_{\rm acc}(n \to o|b^*)}{W(\mathfrak{b}_o) + R(b^*)} . \tag{A10}$$

This gives, as a condition for the acceptance rule,

$$\frac{\mathcal{P}_{\rm acc}(o \to n|b^*)}{\mathcal{P}_{\rm acc}(n \to o|b^*)} = \frac{Z(b^*)}{Z'(b^*)} . \tag{A11}$$

In the equations above, we have imposed a much stronger condition "super-detailed balance." This has as an important practical advantage that, per definition, for Z' [Eq. (6)] one has to use the same set of additional trial orientations b^* as is used for Z. One, therefore, has to calculate only the Rosenbluth factor $W(o)$ for the old conformation. It is straightforward to show that the acceptance rule used in our algorithm [Eq. (8)] obeys this condition. This proves that indeed the correct distribution is sampled by our algorithm.

2. The algorithm of Sec. III B

In this version of the algorithm, the first segment is selected for each of the chains out of a set of trial positions. The probability that a particular conformation n is generated is the product of the probabilities of the first segment to be selected [Eq. (10)], of the remainder of the chain to be generated [Eq. (11)], and of this configuration to be selected out of the g chains [Eq. (14)]. This probability is given by

$$p(o \to n) = p_f p_c p_p = \frac{\exp[-\beta u_1(n)]}{w_1(n)} \frac{\sum_{l=2}^{M} \exp[-\beta u_l(n)]}{W^*(n)} \frac{W(n)}{Z} = \frac{\exp[-\beta U(n)]}{Z} . \tag{A12}$$

For the reverse move, we have

$$p(n \to o) = \frac{\exp[-\beta U(o)]}{Z'} . \tag{A13}$$

Imposing super-detailed balance gives, as a condition for the acceptance rule,

$$\frac{\mathcal{P}_{\rm acc}(o \to n)}{\mathcal{P}_{\rm acc}(n \to o)} = \frac{Z}{Z'} . \tag{A14}$$

Since Eq. (19) obeys this equation, we have proven that indeed distribution (A1) is sampled correctly.

APPENDIX B: PROOF OF EQ. (29)

Given are Q stochastic variables $a_1, \ldots a_Q$, identical and independent distributed, and each having values between 0 and s. The variable a_i specifies the amount of work of processor i. Since we assume that the amount of work is linearly related to the completion time, a_i is also a measure for the execution time of processor i. The parallel execution time can be determined by considering the processor with the most work. Hence, we have to determine the expected maximum of the Q stochastic variables a_i.

$$E \max_i\{a_i\} = \sum_{k=0}^{s} k \mathrm{Prob}(\max_i\{a_i\} = k) .$$

It is straightforward to show that the right-hand side of the above equation equals

$$s - \sum_{k=0}^{s-1} \mathrm{Prob}(\max_i\{a_i\} \le k) .$$

Since the Q stochastic variables are all independent, we have

$$E \max_i\{a_i\} = s - \sum_{k=0}^{s-1} \mathrm{Prob}(a \le k)^Q = s + 1 - \sum_{k=0}^{s} \mathrm{Prob}(a \le k)^Q ,$$

where the stochastic variable a denotes the amount of work of a processor.

Note that this equation is valid for any probability distribution Prob. If we apply the above result to our problem, i.e., the growing of chains in parallel, then the amount of work of a processor equals the number of chains that have successfully been grown. The proba-

bility to grow one chain successfully is $F(f)$ [Eq. (23)]. For this problem, the probability to grow k chains successfully given s attempts is binomially distributed, since each attempt is independent of other attempts. Consequently,

$$\text{Prob}(a \leq k) = C(F(f), k, s) ,$$

$$C(f, k, s) = \sum_{i=0}^{k} \binom{s}{i} f^i (1-f)^{s-i} .$$

[1] N. Metropolis *et al.*, J. Chem. Phys. **21**, 1087 (1953).
[2] D. Frenkel, G. Mooij, and B. Smit, J. Phys. Condens. Matter **4**, 3053 (1992).
[3] G. Mooij, D. Frenkel, and B. Smit, J. Phys. Condens. Matter **4**, L255 (1992).
[4] J. I. Siepmann and D. Frenkel, Mol. Phys. **75**, 59 (1992).
[5] J. I. Siepmann, S. Karaborni, and B. Smit, Nature **365**, 330 (1993).
[6] S. Karaborni, N. van Os, K. Esselink, and P. Hilbers, Langmuir **9**, 1175 (1993).
[7] B. Smit *et al.*, Nature **348**, 624 (1990).
[8] K. Esselink, B. Smit, and P. Hilbers, J. Comput. Phys. **106**, 101 (1993).
[9] K. Esselink and P. Hilbers, J. Comput. Phys. **106**, 108 (1993).
[10] S. Plimpton and G. Heffelfinger, in *Scalable High Performance Computing Conference SHPCC '92*, IEEE Computer Society (IEEE Computer Society Press, Los Alamitos, CA, 1992), pp. 246–251.
[11] G. Pawley, K. Bowler, R. Kenway, and D. Wallace, Comput. Phys. Commun. **37**, 251 (1985).
[12] B. Mehlig, D. Heermann, and B. Forrest, Phys. Rev. **B 45**, 679 (1992).
[13] S. Nosé, J. Chem. Phys. **81**, 511 (1984).
[14] W. G. Hoover, Phys. Rev. **A 31**, 1695 (1985).
[15] A. Moatti, J. Goldberg, and G. Memmi, Comput. Phys. Commun. **45**, 355 (1987).
[16] C. Traynor and J. Anderson, Chem. Phys. Lett. **147**, 389 (1988).
[17] C. Zhao and J. Wood, Ann. Nucl. Energy **16**, 649 (1989).
[18] J. Wood, H. Al-Bahadili, and S. Khaddaj, Ann. Nucl. Energy **18**, 155 (1991).
[19] G. L. Singleton, C.-H. Wu, and J.-H. Tsai, Comput. Phys. Commun. **66**, 181 (1991).
[20] D. M. Jones and J. M. Goodfellow, J. Comput. Chem. **14**, 127 (1993).
[21] J. M. Thomas, Sci. Am. **266**, 82 (1992).
[22] B. Smit and J. I. Siepmann, J. Phys. Chem. **98**, 8442 (1994).
[23] M. Allen and D. Tildesley, *Computer Simulation of Liquids* (Oxford University Press, Oxford, 1987).
[24] G. Mooij, Ph.D. thesis, University of Utrecht, 1993 (unpublished).