RAPID COMMUNICATIONS

PHYSICAL REVIEW E VOLUME 51, NUMBER 2 FEBRUARY 1995

Faster Monte Carlo simulations

James L. Blue and Isabel Beichl
National Institute of Standards and Technology, Gaithersburg, Maryland 20899

Francis Sullivan
Supercomputing Research Center, 17100 Science Drive, Bowie, Maryland 20715

(Received 22 September 1994)

For Monte Carlo simulations of systems of size M, either kinetic simulations or equilibrium simulations that use the method of Bortz, Kalos, and Liebowitz [J. Comput. Phys. **17**, 10 (1975)], the best computer time per event has been $O(M^{1/2})$. We present two methods whose computer time per event is $O(M^{1/K})$ or $O(\log M)$. In practice, for typical simulation sizes, $K = 4$ or $K = 5$ is fastest, requiring even less computer time than the $O(\log M)$ method. For typical simulation sizes, we are able to achieve speedup factors of 5 to 7 over the $O(M^{1/2})$ technique.

PACS number(s): 02.70.Lq, 02.50.Ga, 05.50.+q

Monte Carlo (MC) simulations use a noticeable fraction of the available time on many supercomputers and scientific workstations. This paper presents algorithms to do the same simulations in substantially less computer time. The algorithms are applicable both to equilibrium simulations done according to the formulation of Bortz, Kalos, and Lebowitz (BKL) [1] and kinetic [2] Monte Carlo simulations, but not to equilibrium simulations done according to the original Metropolis formulation [3]. We use the terminology of kinetic Monte Carlo to describe our methods.

Let M be the number of possible MC events (or state transitions), and let r_i be the rate at which the ith event should occur. The total rate is

$$R=\sum_{i=1}^{M} r_i . \tag{1}$$

For an accurate simulation, event i should occur with probability r_i/R. (For equilibrium MC done according to BKL, the calculations are the same, but the $\{r_i\}$ are transition probabilities, not rates.)

The inner loop is as follows.

(i) Choose a random number ρ in the range $[0,R)$.

(ii) Find the corresponding ν such that

$$\sum_{i=1}^{\nu-1} r_i \leq \rho < \sum_{i=1}^{\nu} r_i . \tag{2}$$

(iii) Carry out event ν.

(iv) Update those r_i that have changed as a result of event ν; update R and any data structures being used.

Steps (i) and (iii) take time independent of M, but step (ii) is time-consuming. Depending on the data structure needed for the search scheme, the step (iv) time may or may not grow with M. If a simple linear search is used for step (ii), the search time is $O(M)$ and the step (iv) time is $O(M)$ at the most.

The current state of the art [4–6] is the binning method due to Maksym [7]. To facilitate the search, maintain a data structure of partial sums; there are $\lceil M/g \rceil$ partial sums, each containing the sum of g rates,

$$S_j=\sum_{i=1+(j-1)g}^{jg} r_i ,\ \ j=1,\ldots,\lceil M/g \rceil . \tag{3}$$

The search in step (ii) has two parts: searching for the right bin, which takes time $O(M/g)$, and searching within the bin, which takes time $O(g)$. The updating in step (iv) is independent of M. Minimizing the total time leads to $g=\alpha M^{1/2}$; α depends on the relative time in the two searches. In our simulations, the time used is reasonably insensitive to α in the range 0.5–1.5, and $\alpha = 1$ is used in the results presented later. Clearly, the computer time per simulated event is $O(M^{1/2})$.

We refer to Maksym's method as a two-level search scheme. Better asymptotic behavior can be obtained by constructing K-level schemes with $K > 2$. The lowest level is identical except for a slight change in the notation,

$$S_j^{(2)}=\sum_{i=1+(j-1)g}^{jg} r_i ,\ \ j=1,\ldots,\lceil M/g \rceil \tag{4}$$

and higher levels are defined recursively,

$$S_j^{(k)}=\sum_{i=1+(j-1)g}^{jg} S_i^{(k-1)} ,$$
$$j=1,\ldots,\lceil M/g^{k-1} \rceil ,\quad k=3,\ldots,K. \tag{5}$$

(The g's can depend on k and should depend on K, but for simplicity this is not reflected in the notation.) Minimizing the total time, we find g is $O(M^{1/K})$. The total search time is $O(KM^{1/K})$, and the time for step (iv) is $O(K)$, since K partial sums need to be changed. The extra space required for the data structure is $O(M^{1-1/K}+M^{1-2/K}+\cdots+M^{1/K})$.

For a given M, the best asymptotic time behavior is obtained by using the largest feasible K, namely, the one for which there are only two items in each partial sum,

$M^{1/K}=2$, or $K=\log_2 M$. The data structure is then a binary tree. The search time and the step (iv) time are each $O(\log_2 M)$. The extra space required for the data structure is $O(M)$.

The binary tree may be built as follows (for simplicity of notation, we assume $M=2^K$). Let $T_j^{(1)}=r_j$ and generate the rest of the tree recursively.

$$T_j^{(k)}=T_{2j-1}^{(k-1)}+T_{2j}^{(k-1)}, \quad k=2,\ldots,K. \tag{6}$$

Note that $T_1^{(K+1)}=R$.

Searching the binary tree is done recursively, starting with $T_1^{(K)}$. Start with $j=1$, $k=K$. Recursively, if $\rho \leqslant T_{2j-1}^{(k-1)}$, the desired event is in the left half; set $j=2j-1$. Otherwise, subtract $T_{2j-1}^{(k-1)}$ from ρ and set $j=2j$. If $k=1$, we are done; otherwise set $k=k-1$ and continue.

Updating the binary tree after an event is also done recursively by propagating the effects of a change r_ν from $k=1$ to $k=K$. Both the searching time and the updating time are $O(K)=O(\log_2 M)$.

To compare these methods, we simulated a solid-on-solid model of epitaxial growth [4,5,8] on a simple cubic $N\times N$ lattice. Atoms are deposited at a steady rate and can hop around on the surface, but do not evaporate from the surface. Further details of the model may be found in [5,8]. Of interest here is the computer time for the simulation, expressed in microseconds per simulated deposition or hop event.

For the K-level methods, $g=\lceil M^{1/K}\rceil$ was used. Simulations were done on an IBM RISC 6000 Model 590 workstation with one gigabyte of memory. (Certain commercial equipment may be identified in order to adequately specify or describe the subject matter of this work. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the equipment identified is necessarily the best available for the purpose.) For each data point, five trials with different seeds of the random number generator were done and the median time was used. As shown in Fig. 1, for typical simulation sizes, about $N=512$, the two-level scheme takes more than seven times as long as the five-level. Our runs on other computers (Silicon Graphics Indigo2 and Convex C3820) give qualitatively similar results, with the two-level scheme taking about five times as long as the five-level at $N=512$, but the details of the curves vary considerably, depending on the compiler-computer combination and the presence of cache memory.

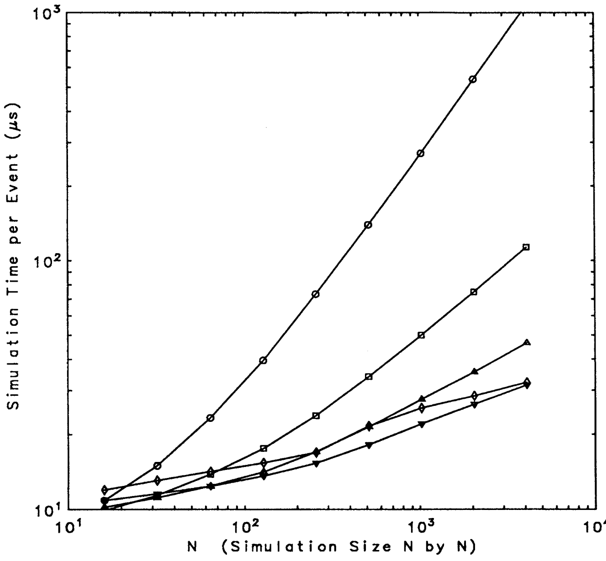

FIG. 1. Log-log plot of time per event for different search schemes. ○, $K=2$; □, $K=3$; △, $K=4$; ▽, $K=5$; ◇, binary tree.

For the shortest times, K should grow as $\log M$, keeping the bin size g constant. Based on our experiments, bin sizes of 5–15 are reasonable. Speedups by a factor of 5 or more for typical problem sizes are obtained over the two-level method.

Note added in proof. The authors recently became aware of [9], which apparently is the first use of a binary tree for searching in Monte Carlo simulations.

[1] A. B. Bortz, M. H. Kalos, and J. L. Lebowitz, J. Comput. Phys. **17**, 10 (1975).

[2] K. Binder, *Monte Carlo Methods in Statistical Physics* (Springer-Verlag, Berlin, 1979), p. 30.

[3] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, J. Chem. Phys. **21**, 1087 (1953).

[4] D. D. Vvedensky, in *Semiconductor Interfaces at the Sub-Nanometer Scale*, edited by H. W. M. Salemink and M. D. Pashley (Kluwer, Amsterdam, 1993), pp. 45–55.

[5] C. Ratsch, A. Zangwill, P. Šmilauer, and D. D. Vvedensky, Phys. Rev. Lett. **72**, 3104 (1994).

[6] A. Zangwill (private communication).

[7] P. A. Maksym, Semicond. Sci. Technol. **3**, 594 (1988).

[8] S. Clarke and D. D. Vvedensky, J. Appl. Phys. **63**, 2272 (1988).

[9] C. K. Wong and M. C. Easton, SIAM (Soc. Ind. Appl. Math.) J. Comput. **9**, 111 (1980).