N87-26538

A Fast MPP Algorithm for Ising Spin Exchange Simulations

Francis Sullivan and Raymond D. Mountain
National Bureau of Standards
Gaithersburg, Maryland 20899

September 19, 1986

Abstract

Our main purpose is to describe a very efficient MPP algorithm for performing one important class of Ising spin simulations. Results and physical significance of MPP calculations using the method described here will be discussed elsewhere. However, we will make a few comments on the problem under study and report briefly on results so far. Ted Einstein has provided us with much guidance in interpreting our initial results and in suggesting calculations to perform.

1 Introduction

Ising spin simulations occur in many areas of physics and have attracted the attention of researchers since the earliest days of electronic computation (Ref. 4). They provide a very good example of how, from the point of view of algorithm design, two apparently disparate problem types can be attacked by almost the same techniques. Some classes of Ising spin calculations can require speeds far in excess of anything currently available.

The basic model is easily described. The spin variable $\sigma(i)$ is specified at the nodes of a uniform grid in two (or three) dimensions. At each grid site the spin can take on only the values $+1$ and -1. Spins are related to one another via the energy expression given by the Hamiltonian

$$H = -\sum_{\{i,j\}} \sigma(i)\sigma(j)$$

where $\{i,j\}$ ranges over all nearest neighbor pairs of sites. In an $\ell \times \ell$ square lattice, at site i we have the local energy expression

$$\begin{aligned} E(i) &= -\sigma(i) \times (\sigma(i+1) + \sigma(i-1) \\ &\quad + \sigma(i+\ell) + \sigma(i-\ell)). \end{aligned}$$

In most cases periodic boundary conditions are imposed, so that $i+1$ and $i+\ell$ are to be determined mod ℓ.

One wishes to compute various averages with respect to the probability $P(C)$ for a configuration of spins, C, to occur. The "classical" algorithm for using Monte Carlo methods to sample configuration space is due to Metropolis, Rosenbluth, Rosenbluth, Teller and Teller (Ref. 4), (called the $M(RT)^2$ algorithm for short). It consists of a series of moves through configuration space, making use of the fact that for each C, $P(C) \propto \exp(-JE(C)/kT)$. Here $E(C)$ is the energy associated with configuration C, J is the coupling constant for the problem under study, T is the temperature and k is Boltzmann's constant. A site i is chosen at random and the change in energy $\Delta E(i)$ which would result in reversing the spin at that site is determined. Since only the site i and its four nearest neighbors are involved, it is easy to see that the change in energy is

$$\begin{aligned} \Delta E(i) &= (E'(i) - E(i)) \\ &\quad +(E'(i+1) - E(i+1)) \\ &\quad +(E'(i-1) - E(i-1)) \\ &\quad +(E'(i+\ell) - E(i+\ell)) \\ &\quad +(E'(i-\ell) - E(i-\ell)) \\ &= -4E(i) \end{aligned}$$

If $\Delta E(i) \leq 0$, then the move is "accepted" and the sign of $\sigma(i)$ is reversed. In case $\Delta E(i) > 0$, the move is accepted with probability $\exp(-J\Delta E/kT)$.

It can be shown that the $M(RT)^2$ algorithm defines a Markov process which samples the "correct" (Boltzmann) distribution of configuration of spins. However, for a large system several hundreds of thousands, or even millions of updates of each site must be performed in order to approach a single equilibrium configuration, and often averages over many such configurations are required. Most of the work is in generating the random numbers, since as many as 10^{12}

moves of sites may be required in a single simulation. We will delay discussion of this critical aspect for the moment, and concentrate on how to modify $M(RT)^2$ in order to get an algorithm well suited to the MPP. Later we explain why this algorithm is also well suited to vector machines, and we then describe one method for vectorizing the generation and testing of random numbers.

2 Algorithm

The MPP consists of a square array of 128x128 single-bit processors, having an SIMD architecture. Processors are interconnected via nearest neighbor bonds and periodic boundary conditions is one possible connection pattern for border processors. Because of its SIMD architecture, the MPP is extremely well adapted to image processing applications and, in fact, one point we hope to make is that, so far as algorithms are concerned, Ising spin simulations are a type of image processing problem.

One is tempted to think of associating processors with spin sites in a one-one manner. However, this does not make very good use of the machine, since in the $M(RT)^2$ algorithm, only one site is examined for each move. Instead of mapping sites to processors in the obvious way, we instead notice that the expressions for the energy associated with a site are similar in form to the central difference approximation used to solve the Poisson equation,

$$-\nabla^2 u = \rho.$$

In the Poisson case, a grid site and its four nearest neighbors are related through the finite difference expression

$$\begin{aligned} -u(i-\ell) - u(i-1) & \\ +4u(i) & \\ -u(i+1) - u(i+\ell) &= (\Delta x)(\Delta y) * \rho(i). \end{aligned}$$

A common strategy used in implementing iterative methods for solving the Poisson equation is to use the "red-black ordering" depicted below:

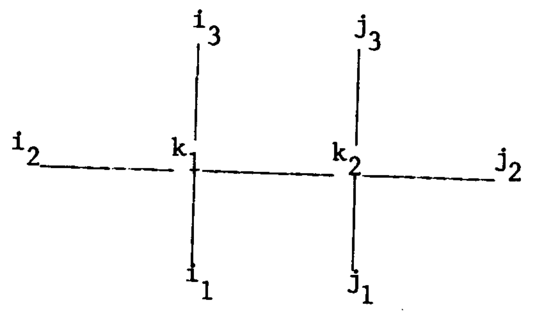
$$\begin{array}{cccccc} R & B & R & B & R & B \\ B & R & B & R & B & R \\ R & B & R & B & R & B \\ B & R & B & R & B & R \end{array}$$

Since no pair of red sites are nearest neighbors, all red sites can be updated "simultaneously". These values can then be used to update the black sites, and so-forth, alternating on each iteration between red and black sites.

The same idea can be applied to modify $M(RT)^2$. Spin flips can be attempted at all red sites or at all black sites. This is a different Markov process than $M(RT)^2$ but the same distribution is sampled. Of course it is the dynamical aspects which are of interest now, rather than merely the converged solution. This use of colors is part of the so-called multi-spin algorithm (Ref. 5). In order to implement multi-spin on the MPP, all sites of a single color can be associated with a single MPP plane of 128x128 processors.

The problem to which we have applied multi-spin is slightly more complicated than that of attempting to flip single sites (Glauber dynamics). Instead we want to try to exchange the spins of a pair of neighboring sites. These spin exchanges or Kawasaki dynamics calculations can be used to study phenomena such as growth of magnetic domains (Ref. 2).

Because spin exchanges are to be attempted, eight different sites are involved in each move. For example to interchange sites k_1 and k_2 we have all of the depicted bonds to consider. The change in energy is



the sum of changes over all eight sites and is given by the expression

$$\begin{aligned} \Delta E &= -2(\sigma(k_2) - \sigma(k_1)) \\ &\quad * \{\sigma(i_1) + \sigma(i_2) + \sigma(i_3) \\ &\quad -\sigma(j_1) - \sigma(j_2) - \sigma(j_3)\}. \end{aligned}$$

In order to apply the multi-spin idea enough "colors" must be assigned so that no two sites of the same "color" are involved in the same move. This can be accomplished by partitioning sites into sixteen groups as is shown below:

```
15  16  13  14  15  16
11  12   9  10  11  12
 7   8   5   6   7   8
 3   4   1   2   3   4
        13  14  15  16
```

Each group is now associated with a single 128x128 MPP bit plane, so that a 512x512 simulation can be handled in a straight-forward way.

Notice now that the part of the expression for ΔE which is enclosed in brackets can take on only seven distinct values, namely $\{-6, -4, -2, 0, 2, 4, 6\}$. The MPP array consists of single-bit processors and Boolean operations can be done simultaneously in a single operation on all 2^{14} processors. The computation of ΔE can be reduced to Boolean operations by first associating the spin values $\sigma = \{+1, -1\}$ with Boolean values $= \{1, 0\}$ (which in effect makes a transcription to a lattice gas model), and next noticing that the value of the bracket part of ΔE corresponds to the sum of the 1 bits of the Boolean expression. For example:
Spin version:

$$\{1\ 1\ -1\ 1\ 1\ 1\}$$

Value = 4.
Boolean version:

$$\{1\ 1\ 0\ 1\ 1\ 1\}$$

Sum of bits = 5

The complete correspondence is as follows:

Spin Sum	Bit Sum	Representation
6	6	110
4	5	101
2	4	100
0	3	011
-2	2	010
-4	1	001
-6	0	000

The bit representation for the sum of bits can itself be determined by defining summation of Boolean expressions by Boolean operations. Assume that BITS(I), I = 1,2,...,6 is an array of bit planes each consisting of all 128x128 sites of a given color. The bit representation of the sum consists of three more bit planes B1, B2, and B3, initially all zero. Addition is performed bit by bit with the proper rules for carries. The correct representation for the final values is obtained by executing the following loop.

```
      DO  10  I=1,6
        B1 = B1 .OR. (BITS(I)
               .AND. B3 .AND. B2)
        B2 = (.NOT. (BITS(I)
             .AND. B3) .XOR. (.NOT. B2))
        B3 = (.NOT. BITS (I))
             .XOR. (.NOT. B3)
 10     CONTINUE
```

The sign of the sum of neighboring spins is determined by the value of the high order bit B1. This must be combined with the value of $-2(\sigma(k_2)-\sigma(k_1))$ to determine when a random number needs to be compared with $\exp(-J\Delta E/kT)$. Of course, the exponential also takes only finitely many values, so that it is easy to parallelize the comparison step. The generation of random numbers can be done simultaneously if a method which allows more than 2^{14} seeds is used. We have adapted a program due to Marsaglia and Kahaner (Ref. 3), but other methods are possible. In any case, 2^{14} pairs of sites are handled in a single step. The total number different types of nearest neighbor pairs ($< 1,2 >, < 6,7 >, < 9,5 >$, etc.) is 32 , and one should not cycle through these types in a fixed pattern because this marching introduces false dynamics into the simulation.

3 Vector Machines

Essentially the same algorithm can be used on a vector machine such as the CYBER 205. The representations for ΔE can be computed very efficiently using bit vector operations. However, vector instructions are not really the same as parallel instructions, and so the 2^{14} random numbers which are required for each step take a lot of time to generate on a vector machine. The way to alleviate this is to reduce even the comparison with random numbers to vector operations on bit arrays. Assume, for convenience, that the value of $-2(\sigma(k_2) - \sigma(k_1))$ is -4, and let $\alpha = \exp(-8J/kT)$. As in (Ref. 1), we create bit arrays D1 D0 of length 2^{14} where D1D0 = 11 with probability α^3; D1D0 = 10 with probability $\alpha^2 - \alpha^3$, D1D0 = 01 with probability $\alpha - \alpha^2$ and D1D0 = 00 with probability $1 - \alpha$. This is easily done by generating 2^{14} random numbers and noting where they fall in the intervals I3 = $[0, \alpha^3)$, I2 = $[\alpha^3, \alpha^2)$, I1 = $[\alpha^2, \alpha)$ and I0 = $[\alpha, 1]$. The test can now be performed by computing the bits of (B1 B2 B3 + D1 D0) $\oplus$ 001. Here the operation $\oplus$ means addition modified so that the high-order bit is not "turned off" by a carry operation (e.g. 111 $\oplus$ 001 = 100). After this operation has been performed, exchanges can be made at all sites with high-order bit equal to 1.

A little reflection reveals that the above method performs exchanges with nearly the correct frequency, because the bit vectors D1D0 approximate the correct probability distribution. After D1D0 have been used for a step, the vectors must be permuted in some random fashion, and to insure that the false periodicities are not introduced, the D1D0 vector should occasionally be re-loaded by generating 2^{14} new random numbers. This, of course, is *much* more efficient than generating a full set of random numbers for each step.

4 Preliminary Results

The first program to implement the algorithm on the MPP was written in Parallel Pascal by Julie O'Connell. However, signigicant portions of it were later recoded in PEARL by Jim Abeles. The result is a code which performs better than 200 million tests of spin sites per second. Since our goal is to study the long time growth of domains, very long simulation times and averages over many different configurations are required. The present code makes this a practical possibility.

As has been mentioned, in the case of spin exchanges great care must be taken to avoid introducing any false periodicities into the results. This need for care is especially acute in our study because there is considerable controversy about the long time behavior of such models (Ref. 6). Preliminary runs on the MPP indicate that our algorithm is correct. We are now conducting more extensive tests of the reliability of our technique for using the random number generator.

References

1. Bhanot, G., Duke, D. and Salvador,R. 1986. A fast Algorithm for the CYBER 205 to Simulate the 3-D Ising Model.*J. Statistical Physics.* to appear.

2. Huse, D. A. 1986. Late Stages of Spinodal Decomposition: Corrections to Lifshitz-Slyozov Scaling and Monte Carlo Simulations. *preprint.*

3. Marsaglia, G. and Kahaner, D. Generation of Uniform and Normal Random Variables. *preprint.*

4. Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. and Teller, E. 1954. *J. Chem. Phys.* **21**: 1087.

5. Williams, G. O. and Kalos, M. H. 1984. A New Multispin Coding Algorithm for Monte Carlo Simulation of the Ising Model *J. Statistical Physics* **37**: 283-299.

6. Mazenko, G. F., Valls, O. T. and Zhang, F. C. 1985. *Phys. Rev.* **B31**: 4453.