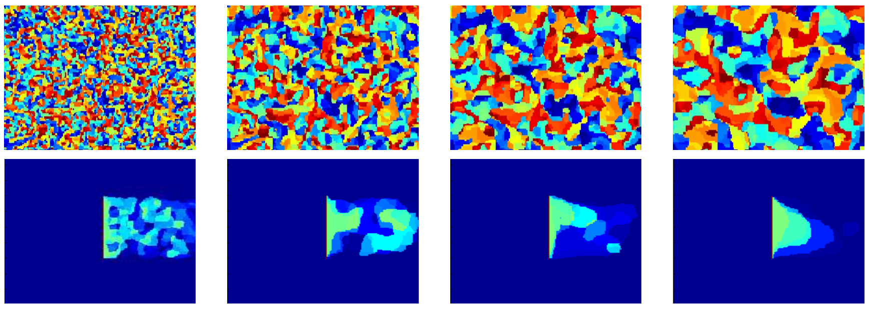

Figure 1: *Bit-Parallel Computation of Ising Marginal* "Bit twiddling" allows us to compute 64 simultaneous realizations of the Ising process on a single array of long integers. In the plots above, color is assigned according to nominal integer value, i.e. each realization contributes according to the significance of its bit position. This is the most direct way of visualizing the data, and the exponential decay of bit significance gives a sense of transparency and depth. Cf. Fig. **??**

1 Numerical Optimizations

1.1 Bit-Parallel Monte Carlo

Recall that the update probability for our Monte Carlo Ising process is

$$P(I_{ij}^{t+1} = 1 | I^t) = \frac{\exp\big(\beta(H_{ij}^t - 2)\big)}{\exp\big(\beta(H_{ij}^t - 2)\big) + \exp\big(-\beta(H_{ij}^t - 2)\big)} \tag{1}$$

where

$$H_{ij}^t := I_{i+1,j}^t + I_{i,j+1}^t + I_{i-1,j}^t + I_{i,j-1}^t$$

This computation, which produces a single random bit, requires a new random floating-point number to be generated each time it is run. Many such bits need to be averaged to compute the marginal obstacle probabilities. If this operation could be approximated using only bit-wise operations, we could run these operations in parallel, one for each bit in a multi-bit datatype.

1.1.1 Overview

We define a random process $\bar{I}_{ij}^t$ to approximate I_{ij}^t. For simplicity, $\bar{I}$ will take values in $\{0, 1\}$, where I took values in $\{-1, 1\}$. First, we generate a random bit $B \sim \text{Bernoulli}(0.5)$. Recursively define

$$B^0 := B, \quad \text{and} \quad B^k := B_1^{k-1} \wedge B_2^{k-1}, \tag{2}$$

where $B_{1,2}^{k-1} \sim B^{k-1}$ are i.i.d. copies (their generation is discussed in the next section). Thus $B^k \sim \text{Bernoulli}(2^{-2^k})$. Define

$$\bar{I}_{ij}^{t+1} := \begin{cases} B(\bar{H}_{ij}^t) & \bar{H}_{ij}^t < 0 \\ \sim B(\bar{H}_{ij}^t) & \bar{H}_{ij}^t > 0 \\ B^0 & \bar{H}_{ij}^t = 0 \end{cases}, \tag{3}$$

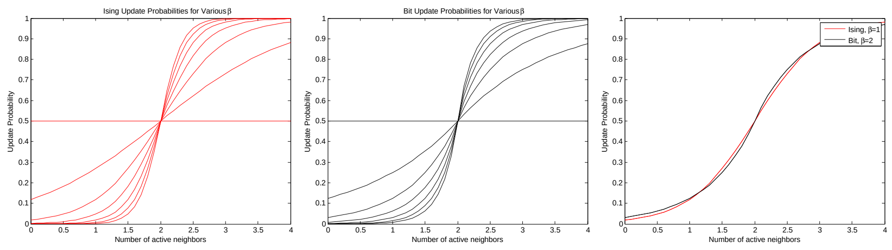

Figure 2: *Comparison of Update Probabilities:* Values of $\bar{\beta}$ are taken from 0 to 6. β is taken such that $\log(2)\bar{\beta} = \beta$. Update probabilities for the Ising Model and the parallel bit estimate are computed on the continuum $[0, 4]$, although only integers $\{0, \ldots 4\}$ are used in practice.

where

$$B(H) := \bigwedge_{k=1}^{K} ((\sim C_k(H)) \vee B^k), \tag{4}$$

$$C_k(H) := k\text{-th bit of } \bar{\beta}(|H| - 1). \tag{5}$$

Here $\bar{\beta}$ must be an integer. Observe that

$$\exp_2\big(-\sum_{k=1}^{K} 2^k\, C_k(H)\big) = 2^{-\beta(|H|-1)},$$

and thus

$$E(\bar{I}_{ij}^{t+1}) = \begin{cases} 2^{-\beta(|H|-1)} & H_{ij}^t < 0 \\ 1 - 2^{-\beta(|H|-1)} & H_{ij}^t > 0 \\ 0.5 & H_{ij}^t = 0 \end{cases} \tag{6}$$

In a more practical sense, as the original ising model

1.1.2 Computing the counts C_k

The counting bits C_k are computed as with a standard binary adder. This requires $O(k)$ bitwise AND and/or XOR operations for binary sum and carry, and about $2k$ boolean variables as registers.

1.1.3 Generating I.I.D. Bits

equivalent to the conjunction of 2^k independent coin-flips B_i^0:

$$B^k \sim B_1^0 \wedge B_2^0 \wedge \cdots B_{2^k}^0.$$

Instead of calling the random number generator 2^k times, we generate a single N-bit ($N \geq 2^k$) integer L (for "long"), and conjoin it with k bit-shifted copies of itself. Recursively define

$$L^0 := L, \qquad L^k := (L^{k-1} \wedge (L^{k-1} \ll 2^{k-1})) \ll 3 \cdot 2^{k-1} - 2. \tag{7}$$

Observe that

$$\begin{aligned}
L^k &= (L^{k-1} \wedge (L^{k-1} \lll 2^{k-1})) \lll 3 \cdot 2^{k-1} - 2 \\
&= (L^{k-1} \lll 2^{k-1}) \wedge (L^{k-1} \lll 2 \cdot 2^{k-1}) \lll 2^k - 2 \\
&= (L^{k-1} \lll 2^{k-2}) \wedge (L^{k-2} \lll 2 \cdot 2^{k-2}) \\
&\qquad \wedge (L^{k-1} \lll 3 \cdot 2^{k-2}) \wedge (L^{k-2} \lll 4 \cdot 2^{k-2}) \lll 2^k - 2 \\
&\ \ \vdots \\
&= \left(\bigwedge\nolimits_{j=0}^{2^k-1} (L^0 \lll j) \right) \lll 2^k - 1.
\end{aligned}$$

Thus, for the i-th bit of L^k:

$$(L^k)_i = \bigwedge\nolimits_{j=2^k-1}^{2^{k+1}-2} L^0_{i-j}$$

where bit indices are taken modulo N. This is a 2^k-fold conjunction of independent random bits, so

$$P((L^k)_i = 1) = 2^{-2^k},$$

Observe, also, that the bits

$$(L^{k_1})_i = \bigwedge\nolimits_{j=2^k-1}^{2^{k+1}-2} L^0_{i-j} \quad \text{and} \quad (L^{k_2})_i = \bigwedge\nolimits_{j=2^k-1}^{2^{k+1}-2} L^0_{i-j}$$

are independent for $k_1 \neq k_2$ (the ranges $[2^{k_1}-1, 2^{k_1+1}-2]$ and $[2^{k_2}-1, 2^{k_2+1}-2]$ do not overlap). Note, however that when indices i_1 and i_2 differ by less than 2^k, the bits $(L^k)_{i_1}$ and $(L^k)_{i_1}$ are highly correlated, especially for large k.

In modern processors, the bit-shift operation runs in constant time, so the random lookup integers $\{L^k, \ldots, L^k\}$ are generated in $O(K)$ time.