UNIVERSIDADE DE SAO PAULO
INSTITUTO DE FISICA E QUIMICA DE SAO CARLOS
DEPARTAMENTO DE FISICA E CIENCIA DOS MATERIAIS

**"Relevo Energético de Vidros de Spin numa
Rede Quadrada"**

Eduardo Soares Rodrigues

**Dissertação apresentada ao IFQSC/USP
para obtenção do título de Mestre em
Ciencias "Fisica Aplicada"**

Orientador: Prof.Dr.Paulo Murilo Castro de Oliveira

**Sao Carlos - Sao Paulo
1993**

**IFSC UNIVERSIDADE
DE SÃO PAULO**
Instituto de Física e Química de São Carlos

Fone (0162) 72-6222
Fax (0162) 72-2218

MEMBROS DA COMISSÃO JULGADORA DA DISSERTAÇÃO DE MESTRADO DE ~~EDUARDO SOARES RODRIGUES~~ APRESENTADA Carlos Botelho, 1465
AO INSTITUTO DE FÍSICA E QUÍMICA DE SÃO CARLOS, DA UNIVERSIDADE DE SÃO PAULO, EM 20/09/1993 Caixa Postal 369
CEP 13560.970 - São Carlos - SP
Brasil

COMISSÃO JULGADORA:

Prof. Dr. Paulo Murilo Castro de Oliveira

Prof. Dr. Roberto Nicolau Opody

Prof. Dr. Francisco Castilho Alcaraz

Dedicatória

Dedico este trabalho aos meu pais: que sempre estiveram ao meu lado em todas as circunstâncias; ora por criticar, ora por aconselhar.

Dedico também aos filhos da terra, que por toda a eternidade dos séculos lutaram para que um dia toda criança e velho tivessem direito à alimentação , moradia, saúde e educação .

Agradecimentos

Inúmeras são as pessoas que deveria eu agradecer. Algumas delas talvez eu não citasse aqui: quer por esquecimento, quer por ideologia. Na tentativa de não ser injusto, não citarei nomes; exceto os dois orientadores que tive, pois naturalmente sem eles esse trabalho não seria possível. Agradeço a compreensão, a amizade, a paciência e a confiança que minha ex-orientadora Cristina Terrile e que meu orientador Paulo Murilo depositaram em mim.

Agradeço desse modo a todos aqueles que direta ou indiretamente viveram comigo e me auxiliaram querendo ou não; mas eu agradeço principalmente àqueles que discordaram de mim e que tentaram me prejudicar consciente ou inconscientemente porque eles me tornaram naquilo que sou.

Resumo

Neste trabalho simulamos computacionalmente o modelo $\pm J$ numa rede quadrada para o cálculo dos mínimos e máximos locais de energia. Para tal, fizemos o uso de variáveis **Booleanas** para armazenar as configurações variáveis de spins e as configurações fixas de ligações ferro e antiferromagnéticas entre sítios vizinhos. Definindo uma dinâmica conveniente dos spins na rede, estudamos o gráfico da Energia $\times \vec{S}$, onde $\vec{S}$ representa um elemento do espaço de estados de dimensão $N = L^2$. Utilizamos redes de dimensões $L = 8, 16, 32$ e 64 .

Abstract

In this work we have simulated numerically the $\pm J$ Ising model in a square lattice to obtain the minimum and maximum local energies. We used **Boolean** variables to store the variable configuration of the spins and the fixed configuration of ferro and antiferromagnetic bonds linking nearest neighbors. We have defined a convenient dynamic of the spins on the lattice and we analyzed the behavior of this dynamic by plotting $\text{Energy} \times \vec{S}$, where $\vec{S}$ represents an element of the associated space of states with dimension $N = L^2$. We take lattice with linear sizes $L = 8, 16, 32$ and 64 .

Índice

1	Introdução	7
1.1	Simulação Computacional	7
1.2	Simulação em Vidros de Spin	9
1.3	A Hamiltoniana de Ising	11
2	Evolução Dinâmica no Espaço de Estados	14
2.1	Introdução	14
2.2	Procura de Mínimos e Máximos da Energia	16
3	Estratégia Computacional	24
3.1	Introdução	24
3.2	Redes Quadradas	26
3.2.1	Rede 8×8	31
3.2.2	Rede 16×16	31

3.2.3	Rede 32×32	32
3.2.4	Rede 64×64	32
4	Resultados	37
4.1	Introdução	37
4.2	Resultados	39
4.2.1	Concentração de ligações iguais	39
4.2.2	Concentração de ligações diferentes	41
4.3	Colapso de Dados	42
5	Conclusões	53
A Tempo de Cálculo nas diferentes redes e um Programa em linguagem C para uma rede 32×32		55

Lista de Figuras

2.2.1	Número potencial de inversões numa rede 8×8	20
2.2.2	Número potencial de inversões numa rede 16×16	20
2.2.3	Número potencial de inversões numa rede 32×32	21
2.2.4	Número potencial de inversões numa rede 64×64	21
4.3.1	Histograma de uma rede 16×16 , com uma configuração e um caminho.	45
4.3.2	Histograma de uma rede 32×32 , com uma configuração e um caminho.	45
4.3.3	Histograma de uma rede 16×16 , com 4 configurações pares e 4 ímpares com 48 caminhos.	46
4.3.4	Histograma de uma rede 32×32 , com 4 configurações pares e 4 ímpares com 48 caminhos.	46
4.3.5	Histograma de uma rede 16×16 , com 45 configurações pares e 45 ímpares com 1800 caminhos.	47
4.3.6	Histograma de uma rede 16×16 , com 1512 configurações pares e 1448 ímpares com 3000 caminhos.	47

4.3.7	Concentração de 70% de ligações ferromagnéticas.	48
4.3.8	Concentração de 75% de ligações ferromagnéticas.	48
4.3.9	Concentração de 80% de ligações ferromagnéticas.	49
4.3.10	Concentração de 85% de ligações ferromagnéticas.	49
4.3.11	Concentração de 90% de ligações ferromagnéticas.	50
4.3.12	Concentração de 95% de ligações ferromagnéticas.	50
4.3.13	Histogramas de mínimos e máximos das redes.	51
4.3.14	Histograma do colapso das redes.	52

Lista de Tabelas

1.1.1 Operações básicas com variáveis Booleanas	8
2.2.1 Valores de $ S $ médios	23
3.2.1 Rede 32×32 , palavra com 16 bits	26
3.2.2 Definição do estado da vizinhança na rede 32×32	30
3.2.3 Rede 8×8 , palavras A e B com 32 bits.	33
3.2.4 Definição do estado da vizinhança na rede 8×8	33
3.2.5 Representação binária das palavras usadas como máscaras, na rede 8×8	34
3.2.6 Rede 16×16 , palavras com 16 bits.	34
3.2.7 Definição do estado da vizinhança na rede 16×16	35
3.2.8 Rede 64×64 , palavras com 32 bits.	35
3.2.9 Definição do estado da vizinhança na rede 64×64	36

4.2.1 Tabela de inversão de spins.	40
4.3.1 Valores médios e larguras dos histogramas das energias.	42

Capítulo 1

Introdução

1.1 Simulação Computacional

Uma forma atual de se abordar problemas físicos tem sido a simulação computacional. Com essa técnica podemos analisar de uma forma intermediária a relação entre modelos teóricos e resultados experimentais. Em particular, uma grande parte dos modelos estatísticos para o estudo de fenômenos críticos e transições de fase, têm sido tratados por meio de simulações computacionais.

Na última década, a melhoria constante da tecnologia de hardware e de algoritmos cada vez mais otimizados, tem permitido simulações cada vez mais velozes e maiores nos diferentes campos em simulação. O processamento paralelo [1, 2], em que diferentes partes do sistema são calculadas simultaneamente, tem sido a melhor estratégia para uma boa performance em simulação computacional. Nesse trabalho, trataremos uma classe de modelos estatísticos que nos permite usar processamento paralelo em sistemas de hardware convencional para simulações computacionais. onde para tal usamos computadores pessoais do tipo PC (386 e 486).

Nessa classe de sistemas, trabalharemos com uma rede discreta de pontos, onde

cada ponto poderá assumir valores lógicos, falso ou verdadeiro, que são associados respectivamente a 0 ou 1.

O processamento paralelo é possível devido às operações **Booleanas** disponíveis em qualquer tipo de computador devido ao uso interno de variáveis binárias. Essas operações **Booleanas** são definidas conforme a tabela abaixo.

AND	OR	XOR	NOT
$0 \wedge 0 = 0$	$0 \vee 0 = 0$	$0 \oplus 0 = 0$	$\neg 0 = 1$
$1 \wedge 0 = 0$	$1 \vee 0 = 1$	$1 \oplus 0 = 1$	$\neg 1 = 0$
$0 \wedge 1 = 0$	$0 \vee 1 = 1$	$0 \oplus 1 = 1$	
$1 \wedge 1 = 1$	$1 \vee 1 = 1$	$1 \oplus 1 = 0$	

Tabela 1.1.1: Operações básicas com variáveis **Booleanas**.

Computadores pessoais tipo PC armazenam e calculam dados em palavras. Uma palavra é um conjunto de 16 bits de memória que podem assumir valores 0 ou 1 cada. Em computadores mais avançados uma palavra é um conjunto de 32 ou 64 bits de memória. As quatro operações **Booleanas** definidas na tabela (1.1.1), efetuam-se ao mesmo tempo em todos os bits, ou seja, na palavra toda (processo paralelo). Como consequência do processo paralelo, temos que as operações lógicas são muito mais rápidas que as operações algébricas.

1.2 Simulação em Vidros de Spin

Analiticamente temos poucas informações sobre o modelo $\pm J$ de Vidro de Spin. Com isso, tem sido necessário o uso de análises numéricas para a obtenção de alguns resultados. Nesse caso o método utilizado tem sido o Método de Monte Carlo com redes finitas e com condições de contorno periódicas.

Para o caso simétrico, concentração 1/2 de ligações ferro ou antiferromagnéticas, cálculos com **matrizes de transferências** [3] nos dão estimativas para a energia $E(0)$ e para a entropia $S(0)$ do estado fundamental como:

$$\frac{E(0)}{NJ} \approx -1,4024 \pm 0,0012$$

$$\frac{S(0)}{Nk_B} \approx 0,0701 \pm 0,0005$$

onde N é o número de sítios na rede.

Vemos por aqui que como a entropia é uma medida logarítmica do número de estados acessíveis do sistema, a degenerescência do estado fundamental aumenta indefinidamente no limite termodinâmico, violando desta forma a 3ª lei da termodinâmica.

Este valor da entropia está razoavelmente de acordo com a estimativa de Vannimenus e Toulouse [4] que usaram uma geometria de plaquetas (quadrados) com o auxílio da seguinte equação :

$$\phi_p = \prod_p \text{sgn}(J_{ij})$$

onde $J_{ij} > 0$ para as ligações ferromagnéticas e $J_{ij} < 0$ para as ligações antiferromagnéticas em cada plaqueta. Quando $\phi_p = -1$ a plaqueta é dita frustrada, e ao menos uma ligação restará não satisfeita. No entanto, este valor numérico é distintamente menor que o resultado da simulação de **Monte Carlo** feito por Kirkpatrick [5]:

$$\frac{S(0)}{Nk_B} \approx 0,10$$

O estudo mais recente aponta para $S(0)$ numa rede quadrada com o modelo $\pm J$ o seguinte valor [6]:

$$\frac{S(0)}{Nk_B} \approx 0,08$$

1.3 A Hamiltoniana de Ising

Consideremos agora o modelo proposto por Lenz para seu estudante Ising [7], que é uma descrição simples de ferromagnetos.

Considere que cada átomo numa rede possa assumir dois estados magnéticos distintos, spin **para cima** ou spin **para baixo**, interagindo somente com os vizinhos mais próximos. Se dois spins vizinhos estão no mesmo estado (ambos **para cima** ou ambos **para baixo**) a energia magnética armazenada nesse par assume um dado valor E_1 , e se os spins estão em estados opostos (um **para cima** e outro **para baixo**) a energia magnética assume um outro valor E_2 . A energia total do sistema é a soma desses valores para todos os pares da rede. Escolhendo o nível zero de energia de tal modo que $E_2 = -E_1 = J$, onde $J > 0$, podemos escrever a seguinte Hamiltoniana de Ising [7]:

$$\mathcal{H} = -J \sum_{\langle ij \rangle} S_i S_j \quad (1.3.1)$$

onde S_i representa o estado magnético do átomo i (± 1) e $\langle ij \rangle$ representa um par de átomos vizinhos. A termodinâmica desse modelo foi resolvida exatamente por Ising [7] em uma dimensão e não apresenta ordem de longo alcance (isto é, não tem magnetização espontânea) para temperaturas não nulas. Em duas dimensões, a primeira solução exata foi obtida por Onsager [8] apresentando uma ordem de longo alcance abaixo de uma temperatura crítica bem definida. Este trabalho é considerado um dos grandes marcos da mecânica estatística, sendo um dos poucos problemas resolvidos exatamente para modelos estatísticos; ele também é usado como um teste para essas teorias. Em três dimensões não se obteve ainda a solução analítica do modelo de Ising.

Para trabalharmos com a equação (1.3.1) com o auxílio de variáveis **Booleanas**, podemos reescrevê-la do seguinte modo:

$$\mathcal{H} = 2J \sum_{\langle ij \rangle} \sigma_i \odot \sigma_j \quad (1.3.2)$$

onde σ_i é uma variável **Booleana** que representa o estado magnético do spin i , substituindo S_i . Temos agora para $\sigma_i = 0$, spin **para cima**; para $\sigma_i = 1$, spin **para baixo**. A equação (1.3.2) foi obtida simplesmente pela mudança do nível zero de energia para $E_1 = 0$ e $E_2 = 2J$. Usando $2J = 1$ como unidade de energia, a equação (1.3.2) é mais conveniente que a equação (1.3.1) para cálculos computacionais, devido à operação lógica *XOR* ($\odot$) que pode ser feita simultaneamente para todos os pares de bits das palavras envolvidas. Devemos então armazenar um spin por bit em cada palavra.

Passemos, agora, a considerar o modelo $\pm J$, cuja Hamiltoniana é

$$\mathcal{H} = - \sum_{\langle ij \rangle} J_{ij} S_i S_j \quad (1.3.3)$$

ao invés da eq. (1.3.1). Neste caso a constante de acoplamento J_{ij} , correspondente aos spins vizinhos i e j , pode assumir um dos dois valores $\pm J$. É um exemplo de vidro de spin, em que ligações ferro ($+J$) e antiferromagnéticas ($-J$) são distribuídas aleatoriamente na rede. Como as ligações em cada par de átomos vizinhos $\langle ij \rangle$ podem estar satisfeitas ou não, elas também podem ser armazenadas como variáveis **Booleanas**, uma vez que podemos associar a elas 0 ou 1 respectivamente.

Para o caso ferromagnético ($+J$), caso onde os spins tendem a permanecer paralelos, temos: ligação satisfeita (spins vizinhos paralelos) e ligação não satisfeita (spins vizinhos anti-paralelos). Para o caso antiferromagnético ($-J$), no qual os

spins tendem a permanecer anti-paralelos, temos: ligação satisfeita (spins vizinhos anti-paralelos) e não satisfeita (spins vizinhos paralelos).

Temos assim uma nova forma de reescrever a equação (1.3.2) para o caso chamado de **vidro de spin $\pm J$** :

$$\mathcal{H} = 2J \sum_{\langle ij \rangle} \Lambda_{\langle ij \rangle} \odot \sigma_i \odot \sigma_j \quad (1.3.4)$$

onde a soma se dá sobre todos os pares de vizinhos $\langle ij \rangle$. Os $\Lambda_{\langle ij \rangle}$ s e os σ_i s representam variáveis **Booleanas** cujos valores podem ser 0 ou 1, como já foi dito, obedecendo ao operador lógico *XOR* ($\odot$) dado na tabela (1.1.1). Os valores $\Lambda_{\langle ij \rangle} = 0, 1$ são distribuídos aleatoriamente com uma probabilidade (p) de ligações ferromagnéticas e $(1 - p)$ de ligações antiferromagnéticas respectivamente.

A distribuição total de energia pode também ser armazenada em palavras, onde cada bit corresponde a uma ligação $\langle ij \rangle$. A energia total vem dada pelo número de bits-1 nessas palavras ou o número total de ligações não satisfeitas, de acordo com a equação (1.3.4).

Com esse método queremos então calcular os mínimos e máximos locais de energia em redes quadradas finitas de diferentes tamanhos. Em seguida queremos fazer uma extrapolação para o limite termodinâmico, através do escalonamento de tamanhos finitos (finite size scaling).

Capítulo 2

Evolução Dinâmica no Espaço de Estados

2.1 Introdução

Consideremos uma rede quadrada de $N = L \times L$ sítios, ou seja, cada sítio com quatro vizinhos. Cada sítio da rede pode ter dois estados magnéticos distintos, spin **para cima** ou **para baixo**, e interage somente com os vizinhos mais próximos. A energia total desse sistema vem dada pela Hamiltonia de Ising ((1.3.3) ou (1.3.4)) para o modelo $\pm J$.

Sorteamos uma distribuição inicial dos spins na rede que denotaremos estado inicial e sorteamos também uma distribuição de ligações ferromagnéticas ($+J$) e antiferromagnéticas ($-J$) com concentrações (p) e $(1 - p)$ respectivamente, que denotaremos configuração , e que será mantida fixa durante a evolução dinâmica dos spins. Em seguida consideremos o gráfico da **Energia $\times$ Estado**, para uma configuração obedecendo à Hamiltoniana de vidro de spin de Ising, chamado **Landscape** ou **gráfico do relevo energético**. Imaginando, para efeitos de raciocínio, que o espaço de estados pudesse ser embebido num plano horizontal

e que o eixo das energias fosse vertical, este gráfico teria a aparência de um relevo montanhoso com picos e vales. Na verdade a situação é mais complexa, porque a dimensão do espaço de estados é igual ao número N de spins na rede, não podendo, portanto, ser embebido num plano. A dimensão deste espaço é N porque dois estados distintos de spins são considerados vizinhos quando diferirem entre si apenas pela orientação de um único spin. Desta forma, havendo N possíveis maneiras de modificar um dado estado transformando-o num vizinho, há também N distintas direções neste espaço, que tem portanto uma dimensão N .

Variando os estados, sempre de um para algum de seus N vizinhos no espaço de estados, definimos duas regras dinâmicas distintas:

I) Aumentamos sempre a energia até atingir um valor máximo, e armazenamos esse valor.

II) Diminuimos sempre a energia até atingir um valor mínimo, e armazenamos esse valor.

Repete-se os passos I e II iterativamente.

Feito isso, montamos dois histogramas, um para energias mínimas e outro para energias máximas que nos dá o seguinte comportamento. Ao aumentarmos o tamanho da rede, as curvas de mínimos e máximos de energia diminuem sua largura e o valor médio tende para um determinado limite. O objetivo final é o de fazer uma extrapolação através do escalonamento de tamanhos finitos para determinar o valor médio das energias mínimas e máximas no limite termodinâmico $L \rightarrow \infty$.

2.2 Procura de Mínimos e Máximos da Energia

Um dos principais objetivos da mecânica estatística é o de obter médias térmicas das quantidades físicas de interesse. Para obter essas médias, temos que considerar todos os estados possíveis do sistema.

O mesmo ocorre para o cálculo das energias de uma determinada rede (temos um valor de energia para cada estado). Consideraremos aqui sistemas clássicos, ou seja, sistemas com valores de energia bem definidos para cada estado.

Para o modelo de Ising definido numa rede com N sítios, temos 2^N estados distintos. Temos então que calcular 2^N valores de energias descritos como E_c , onde $c = 1, 2, 3, \dots, 2^N$. A energia E_c de cada estado c pode ser obtida da equação (1.3.3), onde a soma se expande para todos os pares vizinhos da rede.

Considerando condições de contorno periódicas temos, para uma rede quadrada com N sítios, $2N$ ligações e 2^N estados distintos. Obtemos assim $2N \times 2^N$ parcelas a somar. Para redes com L grande torna-se impossível (devido ao grande tempo de cálculo) calcular todas as energias permitidas da rede.

Para soluções exatas com o modelo de Ising numa rede quadrada $L \times L$, nos deparamos então com um sério problema computacional, uma vez que por exemplo uma rede quadrada com $L = 6$, tem $2^{36} = 7 \times 10^{10}$ estados. Num computador tipo PC, programando em linguagem C, levaríamos cerca de um ano para calcular todos estes estados. Além disto, para o nosso problema, teríamos que repetir este cálculo para cada uma das 2^{2N} possíveis configurações.

Por outro lado, bons resultados termodinâmicos necessitam de redes com dimensões de no mínimo 20×20 . Verificar as energias de todos estes estados (2^{400}) para depois destacar os mínimos e os máximos levaria um tempo muito grande.

Torna-se então impossível realizar tais cálculos. Para contornar esta dificuldade, então, adotaremos a regra dinâmica já mencionada de minimizações e maximizações sucessivas da energia. Desta forma, ao invés de varrer todo o espaço de estados, apenas alguns estados selecionados serão visitados.

Sorteamos inicialmente um estado aleatório, a partir do qual estudaremos sua evolução no tempo; chamaremos particularmente essa dinâmica de caminho, lembrando que cada estado difere do seguinte apenas pela inversão de um único spin, e portanto se trata de uma evolução dinâmica passo a passo. Fixando um estado inicial, temos então definida uma energia para a rede. A inversão de um spin da rede nos leva a analisar como se comporta a energia local em torno daquele determinado sítio, uma vez que todos os outros spins continuam inalterados. Um determinado sítio i possui quatro ligações com seus vizinhos mais próximos. A inversão de seu spin modifica apenas a energia local ε_i , que pode assumir cinco valores possíveis 0, 1, 2, 3 ou 4, de acordo com o número de ligações não satisfeitas entre o sítio e seus vizinhos, conforme a equação (1.3.4) tendo $2J$ como unidade de energia.

Temos ainda outro fator relevante a adotar: procuramos uma energia cada vez menor ou então cada vez maior com o objetivo de obter um mínimo ou um máximo, respectivamente. Consideremos primeiro o processo de minimização. Uma possível regra dinâmica simplista seria inverter o spin i caso sua energia local ε_i fosse 3 ou 4, ou não invertê-lo caso sua energia local ε_i fosse 0, 1 ou 2. Varrendo a rede sequencial ou aleatoriamente, até que não houvesse mais nenhum spin com energia local $\varepsilon_i = 3$ ou 4, teríamos chegado a um mínimo local **não estrito**, no sentido que a energia deste estado é menor do que a de qualquer um de seus N vizinhos, ou no máximo **igual** a de alguns dentre estes vizinhos. Não se trata necessariamente, portanto, de um mínimo local **estrito**, cuja energia seja menor (**não igual**) do que a de qualquer vizinho. Este fato está relacionado com a existência de spins com energia local $\varepsilon_i = 2$, cuja inversão não modificaria a energia. Retomando a idéia do relevo

montanhoso, teríamos vales **não pontuais** no formato de “rios”. Seguindo estes “rios” e suas bifurcações, pode-se eventualmente chegar a energias ainda menores do que o mínimo local **não estrito** obtido pela regra dinâmica simplista descrita acima.

Num sistema real, estes mínimos locais **não estritos** não seriam sequer metaestáveis. Não há barreiras energéticas separando-os de eventuais estados de menor energia ao longo dos citados “rios”. A condição de metaestabilidade de um estado, ou de um conjunto conexo de estados vizinhos com a mesma energia, é que qualquer outro estado vizinho deste conjunto tenha energia estritamente maior. Com o objetivo de buscar os valores das energias destes estados verdadeiramente metaestáveis, a regra dinâmica que adotaremos será mais elaborada que aquela descrita acima. Passaremos a descrevê-la. A idéia se inspira na estratégia já adotada anteriormente [9] no tratamento de um problema análogo. Considera-se um campo magnético externo uniforme e infinitesimal¹, apontando inicialmente no sentido dos spins **para cima**. Por ser infinitesimal, este campo só influencia os spins com energia local $\varepsilon_i = 2$. Portanto nossa regra dinâmica de minimização é: inverter o spin selecionado, caso sua energia local seja $\varepsilon_i = 3$ ou 4, ou se o spin estiver atualmente **para baixo** e sua energia local for $\varepsilon_i = 2$. A rede toda é varrida sequencial ou aleatoriamente, repetidas vezes, até que não haja mais spins a serem invertidos. Não é ainda este estado final de spins, no entanto, o mínimo local procurado. A evolução dinâmica prossegue, desta vez com o campo infinitesimal apontado no sentido dos spins **para baixo**. Após uma nova convergência, inverte-se novamente o sentido deste campo infinitesimal, e assim sucessivamente até que o valor da energia não possa mais ser diminuído por este processo. Este último é o valor da energia mínima adotada em nossas estatísticas.

Note que o critério de convergência não pode ser o da configuração de spins

¹Para minimizar, os spins e o campo infinitesimal devem estar paralelos e para maximizar, os spins e o campo infinitesimal devem estar opostos.

se estabilizar, como é dentro de cada inversão de campo infinitesimal, porque os estados metaestáveis procurados não são necessariamente pontos isolados do espaço de estados, mas podem ser conjuntos conexos de estados vizinhos com a mesma energia.

Para ilustrar a eficiência desta estratégia de inversões sucessivas de um campo infinitesimal, as figuras (2.2.1), (2.2.2), (2.2.3) e (2.2.4) mostram o gráfico $\langle G \rangle \times t$, onde o tempo t mede o número de varreduras completas da rede (cada spin testado e atualizado uma vez). A grandeza G mede o número de spins na rede a serem invertidos na próxima varredura, e a média $\langle G \rangle$ corresponde a várias amostras e várias trajetórias **mínimo - máximo** para cada uma, o que explica sua forma arredondada. No tempo $t = 0$ a rede se encontra num máximo de energia. Os mínimos da curva correspondem às inversões sucessivas do campo infinitesimal.

Figura 2.2.1: Número potencial de inversões numa rede 8×8 .

Figura 2.2.2: Número potencial de inversões numa rede 16×16 .

Figura 2.2.3: Número potencial de inversões numa rede 32×32 .

Figura 2.2.4: Número potencial de inversões numa rede 64×64 .

Após obtermos um mínimo da energia passamos a procurar um máximo, adotando exatamente a mesma regra dinâmica, invertendo desta vez os spins com energias locais $\varepsilon_i = 0$ ou 1 ao invés daqueles com $\varepsilon_i = 3$ ou 4 . A caminhada pelo espaço de estados prossegue, desta forma, com sucessivas visitas a mínimos e máximos de energia.

Para evitar que o sistema possa ficar preso em um único máximo e um único mínimo que se sucedam alternadamente, também incluímos um ingrediente de aleatoriedade em nossa regra dinâmica. Ao testar um spin e depois de decidir invertê-lo pela regra até então determinística apresentada acima, realiza-se esta inversão ou não com probabilidade de 50%. Os gráficos anteriores levam em conta este ingrediente. O resultado desta aleatoriedade pode ser apreciado na tabela (2.2.1) que mostra os valores médios da semelhança $|S|$. Esta grandeza S mede a fração de spins que apresentam a mesma orientação em duas configurações (mínimos ou máximos) consecutivas, menos a fração correspondente aos spins opostos quando se comparam as mesmas duas configurações. Desta forma, $S = 1$ corresponde a configurações idênticas, enquanto $S = -1$ corresponde a configurações simétricas (spins **para cima** numa apontam **para baixo** na outra). Devido à invariância da Hamiltonia por inversão global dos spins, configurações simétricas correspondem ao mesmo estado físico. Assim, a grandeza que mede a semelhança entre dois estados é o valor absoluto $|S|$ que vale 1 para estados iguais e será da ordem de $1/\sqrt{N}$ para estados descorrelacionados.

Esse modelo possui algumas semelhanças com a teoria da evolução das espécies elaborada há cerca de um século atrás por C. Darwin [10]. Essa teoria considera a evolução natural das espécies com pequenas mutações aleatórias que modificam ligeiramente os indivíduos, comparados com os pais. Hoje sabe-se que tais mutações localizam-se nas ligações do ácido desoxiribonucleico (**DNA**), estrutura helicoidal que forma os genes. Essas mutações são submetidas a uma seleção não

aleatória, uma vez que os indivíduos modificados devem se adaptar às condições do meio em que se encontram. As mutações aleatórias localizadas caracterizam a evolução passo a passo citada, já que poucas ligações diferenciam um **DNA** do seu antecessor. Essas mutações podem se localizar em qualquer posição ao longo de todo o **DNA**, o que gera as múltiplas possibilidades de evolução nessa estrutura. As duas regras dinâmicas apresentam basicamente as seguintes 3 características: aleatoriedade das mutações, seleção não aleatória em cada passo e exploração de possibilidades múltiplas numa estrutura de bifurcações sucessivas.

Rede	$\langle S \rangle$
8×8	0,31
16×16	0,20
32×32	0,17
64×64	0,17

Tabela 2.2.1: Valores de $|S|$ médios

Capítulo 3

Estratégia Computacional

3.1 Introdução

Utilizaremos a técnica computacional do processamento paralelo, como já descrita anteriormente, para acelerar o cálculo computacional das energias nas redes quadradas. Para isso, cada spin **para cima** ou **para baixo** de um estado será armazenado com o valor 0 ou 1, respectivamente. Necessitamos também a energia local de interação de um spin da rede com seus quatro vizinhos, que é medida como o número de ligações não satisfeitas em torno deste sítio. Esse número pode assumir os valores 0, 1, 2, 3 ou 4 em unidade de $2J$ de acordo com a equação (1.3.4). Temos ainda que armazenar as ligações ferro ou antiferromagnéticas distribuídas ao longo da rede. Utilizaremos bits-0 para ligações ferro e bits-1 para ligações antiferromagnéticas.

Dividiremos a rede quadrada em duas sub-redes como um tabuleiro de xadrez, de modo que os spins de uma dada sub-rede não interajam com os spins da outra [11]. Podemos atualizar uma sub-rede inteira simultaneamente independente da outra. Utilizando essas sub-redes com o armazenamento de spins em palavras e

através de uma operação *XOR* definida na tabela (1.1.1) com uma outra palavra contendo bits-1 nas posições onde os spins devem ser invertidos e bits-0 onde os spins não devem ser invertidos, atualizamos de uma só vez todos os spins de uma palavra [12].

3.2 Redes Quadradas

Definida essa técnica para atualização de spins, dividiremos agora uma rede quadrada em quatro sub-redes $A[i]$, $B[i]$, $C[i]$, e $D[i]$. Por simplicidade consideremos uma rede quadrada de 32×32 - onde é válida a extensão para $32m \times 32m$, onde m é um inteiro positivo [13]. A , B , C , e D são palavras com 16 bits. O estado dos spins 0 ou 1 serão armazenados bit-a-bit nessas palavras. A primeira linha da rede é armazenada nas palavras $D[1]$ e $C[1]$. Usando a notação bW para representar o b -ésimo bit da palavra W ($b = 0, 1, 2, \dots, 15$), temos a primeira linha da rede como:

$${}^0D[1] \quad {}^0C[1] \quad {}^1D[1] \quad {}^1C[1] \quad \dots \quad {}^{15}D[1] \quad {}^{15}C[1]$$

Para a segunda linha temos analogamente $A[1]$ e $B[1]$ e na terceira linha $D[2]$ e $C[2]$ e assim sucessivamente. A rede vem então dada pela tabela (3.2.1).

${}^0D[1]$	${}^0C[1]$	${}^1D[1]$	${}^1C[1]$	$\dots$	${}^{15}D[1]$	${}^{15}C[1]$
${}^0A[1]$	${}^0B[1]$	${}^1A[1]$	${}^1B[1]$	$\dots$	${}^{15}A[1]$	${}^{15}B[1]$
${}^0D[2]$	${}^0C[2]$	${}^1D[2]$	${}^1C[2]$	$\dots$	${}^{15}D[2]$	${}^{15}C[2]$
${}^0A[2]$	${}^0B[2]$	${}^1A[2]$	${}^1B[2]$	$\dots$	${}^{15}A[2]$	${}^{15}B[2]$
${}^0D[3]$	${}^0C[3]$	${}^1D[3]$	${}^1C[3]$	$\dots$	${}^{15}D[3]$	${}^{15}C[3]$
${}^0A[3]$	${}^0B[3]$	${}^1A[3]$	${}^1B[3]$	$\dots$	${}^{15}A[3]$	${}^{15}B[3]$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
${}^0D[16]$	${}^0C[16]$	${}^1D[16]$	${}^1C[16]$	$\dots$	${}^{15}D[16]$	${}^{15}C[16]$
${}^0A[16]$	${}^0B[16]$	${}^1A[16]$	${}^1B[16]$	$\dots$	${}^{15}A[16]$	${}^{15}B[16]$

Tabela 3.2.1: Rede 32×32 , palavra com 16 bits

Ao mesmo tempo que armazenamos um estado dos spins da rede, necessitamos também armazenar a configuração das ligações. Cada sítio possui uma ligação vertical e uma horizontal ligando-o aos seus vizinhos de cima e da direita respectivamente. Por exemplo o sítio ${}^bA[i]$ possui uma ligação horizontal ${}^bAH[i]$ com o vizinho da direita ${}^bB[i]$ e uma outra ligação vertical ${}^bAV[i]$ com o vizinho superior ${}^bD[i]$ conforme a tabela (3.2.1). Sendo assim, as informações totais de uma rede vem dadas pelos seguintes vetores, todos com 16 elementos de 16 bits cada: $A, B, C, D, AH, AV, BH, BV, CH, CV, DH$ e DV . As palavras A, B, C e D contêm as informações sobre o estado de spins que são as variáveis a serem consideradas, enquanto que as palavras $AH, AV, BH, BV, CH, CV, DH$ e DV armazenam a configuração constante de ligações ferro e antiferromagnéticas.

As palavras I_1, I_2, I_3 e I_4 definidas na tabela (3.2.2) armazenam as informações relevantes das ligações ferro e antiferromagnéticas de cada spin armazenado numa palavra S com seus vizinhos e são construídas de acordo com as condições de vizinhança para atualizarmos cada sub-rede. As operações LR e RR correspondem às rotações à esquerda e à direita, respectivamente, transladando todos os bits de uma unidade, adotando condições de contorno periódicas.

Precisamos agora determinar os spins com energia 0, ou seja, aqueles que possuem quatro ligações satisfeitas com os seus quatro vizinhos. A operação bit-a-bit que determina esses spins vem dada por

$$E_0 = (S \wedge I_1 \wedge I_2 \wedge I_3 \wedge I_4) \vee (\neg S \wedge \neg I_1 \wedge \neg I_2 \wedge \neg I_3 \wedge \neg I_4) \quad (3.2.1)$$

que apresenta bits-1 nas posições correspondentes a energias locais iguais a zero e 0 nas outras posições. Analogamente calculamos agora os spins com energia igual a 1, ou seja, aqueles que possuem apenas uma das quatro ligações com seus vizinhos não satisfeita, cujas posições são dadas pela palavra E_1 :

$$E_1 = \neg \left(S \otimes \left((I_1 \wedge I_2) \vee (I_3 \wedge I_4) \right) \right) \wedge (I_1 \otimes I_2 \otimes I_3 \otimes I_4) \quad (3.2.2)$$

que tem bits-1 nas posições correspondentes a energia local 1 e 0 nas outras posições .

Analogamente, temos os valores E_2 , E_3 e E_4 dados a seguir

$$E_2 = \left((I_1 \otimes I_2) \wedge (I_3 \otimes I_4) \right) \vee \left((I_1 \otimes I_3) \wedge (I_2 \otimes I_4) \right) \quad (3.2.3)$$

$$E_3 = (I_1 \otimes I_2 \otimes I_3 \otimes I_4) \wedge \left(S \otimes \left((I_1 \wedge I_2) \vee (I_3 \wedge I_4) \right) \right) \quad (3.2.4)$$

$$E_4 = (I_1 \wedge I_2 \wedge I_3 \wedge I_4 \wedge \neg S) \vee (\neg I_1 \wedge \neg I_2 \wedge \neg I_3 \wedge \neg I_4 \wedge S) \quad (3.2.5)$$

onde essas palavras possuem bit-1 nas posições correspondentes a energias locais iguais a 2, 3 e 4 respectivamente e 0 nas outras posições .

Para minimizar a energia temos que inverter os spins com energia E_3 e E_4 , além daqueles com energia E_2 que estejam opostos ao campo infinitesimal. Temos então que construir uma palavra com bits-1 nas posições correspondentes dos spins a serem invertidos e bits-0 nas demais posições . Tal palavra vem dada por

$$t = E_4 \vee E_3 \vee (E_2 \wedge S) \quad (3.2.6)$$

Precisamos construir ainda uma palavra de 16 bits com probabilidade de 50% de bits-1. Geramos então um número aleatório R com 32 bits, mas só usamos os 8 últimos cujos períodos de repetição são suficientemente grandes para poderem ser considerados aleatórios [12]. Realizamos sobre esse número uma operação de translação de 24 bits para a direita e em seguida 8 bits para a esquerda. Ao contrário das rotações LR e RR já citadas, nas translações não há condições de contorno periódicas: os bits que “transbordam” da palavra são perdidos enquanto bits-0 são incluídos no outro extremo. A palavra fica então expressa do seguinte modo:

bits-0	bits-0	bits-0 e bits-1	bits-0
--------	--------	-----------------	--------

Geramos em seguida outro número aleatório R' e realizamos sobre ele uma operação de translação de 24 para a direita, logo o $n^{\text{º}}$ R' fica da seguinte forma:

bits-0	bits-0	bits-0	bits-0 e bits-1
--------	--------	--------	-----------------

Feito isso realizamos uma operação OR definida na tabela (1.1.1) com os $n^{\text{º}}$ R e R' e assim construímos um número l da seguinte forma:

bits-0	bits-0	bits-0 e bits-1	bits-0 e bits-1
--------	--------	-----------------	-----------------

Em seguida realizamos a operação :

$$t = t \wedge l \tag{3.2.7}$$

para inverter apenas 50% dos spins selecionados inicialmente, evitando assim que o sistema fique preso num ciclo de um mínimo e um máximo que se alternam.

Temos analogamente o mesmo caso para a maximização , exceto o fato de considerarmos os spins com energia E_0 e E_1 ao invés de E_3 e E_4 e os spins com energia E_2 que estejam paralelos ao campo infinitesimal, na construção da palavra

$$t = E_0 \vee E_1 \vee (E_2 \wedge S) \tag{3.2.8}$$

em substituição a equação (3.2.6).

Finalmente, os 16 spins armazenados numa mesma palavra $W[i]$, são atualizados simultaneamente através da operação

$$W[i] = W[i] \odot t \tag{3.2.9}$$

onde $W[i] = A[i], B[i], C[i]$ e $D[i]$.

$$\begin{aligned}
S &= A[i] \\
I_1 &= B[i] \odot AH[i], I_2 = D[i] \odot AV[i] \\
I_3 &= D[1] \odot DV[1] \text{ se } i = 16, \text{ caso contrário } I_3 = D[i+1] \odot DV[i+1] \\
I_4 &= LR(B[i] \odot BH[i])
\end{aligned}$$

Definição do estado da vizinhança da sub-rede A , na rede 32×32

$$\begin{aligned}
S &= B[i] \\
I_1 &= A[i] \odot AH[i], I_2 = C[i] \odot BV[i] \\
I_3 &= C[1] \odot CV[1] \text{ se } i = 16, \text{ caso contrário } I_3 = C[i+1] \odot CV[i+1] \\
I_4 &= RR(A[i]) \odot BH[i]
\end{aligned}$$

Definição do estado da vizinhança da sub-rede B , na rede 32×32

$$\begin{aligned}
S &= C[i] \\
I_1 &= D[i] \odot DH[i], I_2 = B[i] \odot BV[i] \\
I_3 &= B[16] \odot CV[1] \text{ se } i = 1, \text{ caso contrário } I_3 = B[i-1] \odot CV[i] \\
I_4 &= RR(D[i]) \odot CH[i]
\end{aligned}$$

Definição do estado da vizinhança da sub-rede C , na rede 32×32

$$\begin{aligned}
S &= D[i] \\
I_1 &= C[i] \odot DH[i], I_2 = A[i] \odot AV[i] \\
I_3 &= A[16] \odot DV[1] \text{ se } i = 1, \text{ caso contrário } I_3 = A[i-1] \odot DV[i] \\
I_4 &= LR(C[i] \odot CH[i])
\end{aligned}$$

Definição do estado da vizinhança da sub-rede D , na rede 32×32

Tabela 3.2.2: Definição do estado da vizinhança na rede 32×32

3.2.1 Rede 8×8

Para uma rede 8×8 , faremos um mapeamento diferente ao invés daquele usado na rede 32×32 conforme a tabela (3.2.1). Usaremos duas sub-redes A e B ao invés das quatro anteriores e palavras com 32 bits ao invés de 16, conforme a tabela (3.2.3).

Naturalmente, com o mapeamento usado, teremos também novas formas de atualização das sub-redes conforme dadas na tabela (3.2.4), onde os valores PAR , $IMPAR$, M , N , P e Q vem dados na tabela (3.2.5). As operações LR_n e RR_n ainda são rotações para a esquerda ou direita, respectivamente, desta vez deslocando n posições ao invés de apenas uma. Ao contrário da rede 32×32 , os bits bAV e bBV para b par agora correspondem às ligações verticais **abaixo** dos sítios bA e bB , respectivamente.

3.2.2 Rede 16×16

A rede 16×16 vem também mapeada de uma outra maneira. Dividimos a rede também em duas sub-redes A e B com palavras com 16 bits, o que também modifica suas equações para a atualização. O mapeamento da rede vem dado na tabela (3.2.6) e suas regras de atualização na tabela (3.2.7).

Aqui também, bAV e bBV representam as ligações verticais **abaixo** dos sítios bA e bB , respectivamente, para b par.

3.2.3 Rede 32×32

Para a rede de 32×32 o mapeamento (tabela (3.2.1)) e as regras de atualização (tabela (3.2.2)) já foram definidas na introdução do presente capítulo.

3.2.4 Rede 64×64

Para a rede de 64×64 , que é um múltiplo da rede 32×32 conforme tabela (3.2.8), temos um mapeamento análogo, exceto que as palavras têm 32 bits.

Para a atualização temos apenas que modificar o valor limite da condição de contorno da rede dados na tabela (3.2.9).

No apêndice A temos uma aproximação do tempo de cálculo de cada tamanho de rede e um exemplo dos programas utilizados em linguagem C, para a rede 32×32 .

0A	1B	2A	3B	4A	5B	6A	7B
0B	1A	2B	3A	4B	5A	6B	7A
8A	9B	${}^{10}A$	${}^{11}B$	${}^{12}A$	${}^{13}B$	${}^{14}A$	${}^{15}B$
8B	9A	${}^{10}B$	${}^{11}A$	${}^{12}B$	${}^{13}A$	${}^{14}B$	${}^{15}A$
${}^{16}A$	${}^{17}B$	${}^{18}A$	${}^{19}B$	${}^{20}A$	${}^{21}B$	${}^{22}A$	${}^{23}B$
${}^{16}B$	${}^{17}A$	${}^{18}B$	${}^{19}A$	${}^{20}B$	${}^{21}A$	${}^{22}B$	${}^{23}A$
${}^{24}A$	${}^{25}B$	${}^{26}A$	${}^{27}B$	${}^{28}A$	${}^{29}B$	${}^{30}A$	${}^{31}B$
${}^{24}B$	${}^{25}A$	${}^{26}B$	${}^{27}A$	${}^{28}B$	${}^{29}A$	${}^{30}B$	${}^{31}A$

Tabela 3.2.3: Rede 8×8 , palavras A e B com 32 bits.

$$\begin{aligned}
I_1 &= AH \otimes \left((RR_1(B) \wedge P) \vee (LR_7(B) \wedge Q) \right) \\
I_2 &= AV \odot B \\
I_3 &= BV \odot B; I_3 = (LR_8(I_3) \wedge PAR) \vee (RR_8(I_3) \wedge IMPAR) \\
I_4 &= BH \odot B; I_3 = (LR_1(I_4) \wedge M) \vee (RR_7(I_3) \wedge N)
\end{aligned}$$

Definição do estado da vizinhança da sub-rede A , na rede 8×8 .

$$\begin{aligned}
I_1 &= AH \otimes A; I_1 = (LR_1(I_1) \wedge M) \vee (RR_7(I_1) \wedge N) \\
I_2 &= AV \odot A \\
I_3 &= \left((BV \odot RR_8(A)) \wedge PAR \right) \vee \left((BV \odot LR_8(A)) \wedge IMPAR \right) \\
I_4 &= \left((BH \odot RR_1(A)) \wedge P \right) \vee \left((BH \odot LR_7(A)) \wedge Q \right)
\end{aligned}$$

Definição do estado da vizinhança da sub-rede B , na rede 8×8 .

Tabela 3.2.4: Definição do estado da vizinhança na rede 8×8 .

variável	número	representação binária
<i>PAR</i>	1.431.655.765	01010101010101010101010101010101
<i>IMPAR</i>	2.863.311.530	10101010101010101010101010101010
<i>M</i>	4.278.124.286	11111110111111110111111101111110
<i>N</i>	16.843.009	00000001000000010000000100000001
<i>P</i>	2.139.062.143	01111111011111110111111101111111
<i>Q</i>	2.155.905.152	10000000100000001000000010000000

Tabela 3.2.5: Representação binária das palavras usadas como máscaras, na rede 8 × 8.

⁰ <i>A</i> [1]	¹ <i>B</i> [8]	² <i>A</i> [1]	³ <i>B</i> [8]	...	¹⁴ <i>A</i> [1]	¹⁵ <i>B</i> [8]
⁰ <i>B</i> [1]	¹ <i>A</i> [8]	² <i>B</i> [1]	³ <i>A</i> [8]	...	¹⁴ <i>B</i> [1]	¹⁵ <i>A</i> [8]
⁰ <i>A</i> [2]	¹ <i>B</i> [7]	² <i>A</i> [2]	³ <i>B</i> [7]	...	¹⁴ <i>A</i> [2]	¹⁵ <i>B</i> [7]
⁰ <i>B</i> [2]	¹ <i>A</i> [7]	² <i>B</i> [2]	³ <i>A</i> [7]	...	¹⁴ <i>B</i> [2]	¹⁵ <i>A</i> [7]
⋮	⋮	⋮	⋮	⋱	⋮	⋮
⁰ <i>A</i> [8]	¹ <i>B</i> [1]	² <i>A</i> [8]	³ <i>B</i> [1]	...	¹⁴ <i>A</i> [8]	¹⁵ <i>B</i> [1]
⁰ <i>B</i> [8]	¹ <i>A</i> [1]	² <i>B</i> [8]	³ <i>A</i> [1]	...	¹⁴ <i>B</i> [8]	¹⁵ <i>A</i> [1]

Tabela 3.2.6: Rede 16 × 16, palavras com 16 bits.

$$\begin{aligned}
S &= A[i] \\
I_1 &= AH[i] \otimes RR(B[9-i]) \\
I_2 &= AV[i] \otimes B[i] \\
I_3 &= BV[8] \otimes B[8] \text{ se } i = 1, \text{ caso contrário } I_3 = BV[i-1] \otimes B[i-1] \\
I_4 &= LR(BH[9-i] \odot B[9-i])
\end{aligned}$$

Definição do estado da vizinhança da sub-rede A , na rede 16×16 .

$$\begin{aligned}
S &= B[i] \\
I_1 &= LR(AH[9-i] \odot A[9-i]) \\
I_2 &= AV[i] \odot A[i] \\
I_3 &= BV[8] \odot A[1] \text{ se } i = 8, \text{ caso contrário } I_3 = BV[i] \odot A[i+1] \\
I_4 &= BH[i] \odot RR(A[9-i])
\end{aligned}$$

Definição do estado da vizinhança da sub-rede B , na rede 16×16 .

Tabela 3.2.7: Definição do estado da vizinhança na rede 16×16 .

${}^0D[1]$	${}^0C[1]$	${}^1D[1]$	${}^1C[1]$	...	${}^{31}D[1]$	${}^{31}C[1]$
${}^0A[1]$	${}^0B[1]$	${}^1A[1]$	${}^1B[1]$	...	${}^{31}A[1]$	${}^{31}B[1]$
${}^0D[2]$	${}^0C[2]$	${}^1D[2]$	${}^1C[2]$	...	${}^{31}D[2]$	${}^{31}C[2]$
${}^0A[2]$	${}^0B[2]$	${}^1A[2]$	${}^1B[2]$	...	${}^{31}A[2]$	${}^{31}B[2]$
${}^0D[3]$	${}^0C[3]$	${}^1D[3]$	${}^1C[3]$	...	${}^{31}D[3]$	${}^{31}C[3]$
${}^0A[3]$	${}^0B[3]$	${}^1A[3]$	${}^1B[3]$	...	${}^{31}A[3]$	${}^{31}B[3]$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
${}^0D[32]$	${}^0C[32]$	${}^1D[32]$	${}^1C[32]$	...	${}^{31}D[32]$	${}^{31}C[32]$
${}^0A[32]$	${}^0B[32]$	${}^1A[32]$	${}^1B[32]$	...	${}^{31}A[32]$	${}^{31}B[32]$

Tabela 3.2.8: Rede 64×64 , palavras com 32 bits.

$$\begin{aligned}
S &= A[i] \\
I_1 &= B[i] \otimes AH[i], I_2 = D[i] \otimes AV[i] \\
I_3 &= D[1] \otimes DV[1] \text{ se } i = 32, \text{ caso contrário } I_3 = D[i+1] \otimes DV[i+1] \\
I_4 &= LR(B[i] \otimes BH[i])
\end{aligned}$$

Definição do estado da vizinhança da sub-rede A , na rede 64×64 .

$$\begin{aligned}
S &= B[i] \\
I_1 &= A[i] \otimes AH[i], I_2 = C[i] \otimes BV[i] \\
I_3 &= C[1] \otimes CV[1] \text{ se } i = 32, \text{ caso contrário } I_3 = C[i+1] \otimes CV[i+1] \\
I_4 &= RR(A[i]) \otimes BH[i]
\end{aligned}$$

Definição do estado da vizinhança da sub-rede B , na rede 64×64 .

$$\begin{aligned}
S &= C[i] \\
I_1 &= D[i] \otimes DH[i], I_2 = B[i] \otimes BV[i] \\
I_3 &= B[32] \otimes CV[1] \text{ se } i = 1, \text{ caso contrário } I_3 = B[i-1] \otimes CV[i] \\
I_4 &= RR(D[i]) \otimes CH[i]
\end{aligned}$$

Definição do estado da vizinhança da sub-rede C , na rede 64×64 .

$$\begin{aligned}
S &= D[i] \\
I_1 &= C[i] \otimes DH[i], I_2 = A[i] \otimes AV[i] \\
I_3 &= A[32] \otimes DV[1] \text{ se } i = 1, \text{ caso contrário } I_3 = A[i-1] \otimes DV[i] \\
I_4 &= LR(C[i] \otimes CH[i])
\end{aligned}$$

Definição do estado da vizinhança da sub-rede D , na rede 64×64 .

Tabela 3.2.9: Definição do estado da vizinhança na rede 64×64 .

Capítulo 4

Resultados

4.1 Introdução

Consideramos nesse trabalho o modelo de vidro de spin $\pm J$ numa rede quadrada $L \times L$ ($N = L^2$). A densidade de energia é:

$$\mathcal{E} = \frac{1}{2N} \sum_{\langle ij \rangle} \Lambda_{\langle ij \rangle} \odot \sigma_i \odot \sigma_j \quad (4.1.1)$$

onde a soma se expande sobre $2N$ ligações $\langle ij \rangle$ que unem os vizinhos i e j com condições de contorno periódicas. O Λ_s e os σ_s representam variáveis **Booleanas** com valores 0 e 1 para falso e verdadeiro respectivamente, obedecendo as operações bit-a-bit da tabela (1.1.1). Os Λ_{ij} (ligações da rede) são aleatoriamente distribuídos e fixos como uma configuração, a princípio com uma probabilidade de 50%. Ao mesmo tempo, uma dada distribuição dos σ_i é chamada de estado. Para cada configuração, um estado inicial aleatório evolui dinamicamente obedecendo as regras enunciadas na seção 2.2 com um determinado caminho. Essa dinâmica nos leva alternadamente a mínimos e máximos de energia aleatoriamente, no espaço de estados. Um mínimo (máximo) local de energia corresponde ao estado cuja energia $\mathcal{E}$ não pode mais diminuir (aumentar) pela inversão de um único spin, isto é,

realizando uma simples mutação [14]. Medimos os valores dos mínimos e máximos de $\mathcal{E}$ atingidos ao longo dos caminhos, bem como a semelhança $|S|$ entre dois mínimos ou dois máximos sucessivos, conforme tabela (2.2.1).

4.2 Resultados

4.2.1 Concentração de ligações iguais

Dois histogramas típicos de $\mathcal{E}$ para uma rede 16×16 e outra 32×32 são apresentados nas figuras (4.3.1) e (4.3.2). Cada uma delas representa somente um caminho através de 15 mil mínimos e 15 mil máximos.

O interesse nesses histogramas está em suas larguras e na localização de seus valores médios $\langle E_{max} \rangle$ e $\langle E_{min} \rangle$ para diferentes tamanhos de redes.

Histogramas semelhantes para 8 configurações com 48 caminhos são apresentados nas figuras (4.3.3) e (4.3.4) com 720 mil mínimos e 720 mil máximos para cada tamanho.

Surge agora um efeito interessante: cada histograma parece composto de dois outros com valores médios diferentes. A razão física disto é a existência de duas classes distintas de configurações, pares e ímpares. Isto devido ao seguinte fato: ao sortearmos uma determinada configuração de ligações e um determinado estado de spins para a rede, a energia será dada pela soma das ligações não satisfeitas. Como a inversão dos spins dá uma variação par para o número de ligações não satisfeitas conforme a tabela (4.2.1), a energia calculada será um valor sempre par ou sempre ímpar para uma mesma configuração. Sendo assim, todos os 2^N estados de uma dada configuração têm a mesma paridade.

Na figura (4.3.3) temos uma quantidade maior de ligações não satisfeitas nas configurações ímpares, enquanto que na figura (4.3.4) temos o comportamento

oposto. Esses comportamentos indicam que o histograma de uma única configuração depende dessa particular configuração simulada. Então, a média das energias e suas larguras devem ser tomadas de muitas configurações aleatórias. Para testar a dependencia desses resultados, simulamos uma rede 16×16 com 45 configurações pares e 45 ímpares correspondentes à 1,8 milhões de mínimos e 1,8 milhões de máximos distintos, com 20 caminhos por configuração , conforme a figura (4.3.5). Os picos duplos persistem. Já na figura (4.3.6) aparece somente um pico simples para energias mínimas, e outro para energias máximas. Neste caso adotamos 1,2 milhões de mínimos e 1,2 milhões de máximos distintos, com 3000 configurações e apenas um caminho cada. Concluimos então que um único caminho por configuração é suficiente, no entanto são necessárias muitas configurações para obtermos boas médias estatísticas. Utilizaremos a comparação entre as configurações pares e ímpares para controlar a qualidade estatística de nossos resultados.

	inversão do spin central				
n^2 de lig. não satisfeitas antes da inversão	0	1	2	3	4
n^2 de lig. não satisfeitas após a inversão	4	3	2	1	0
variação da energia	4	2	0	-2	-4

Tabela 4.2.1: Tabela de inversão de spins.

4.2.2 Concentração de ligações diferentes

Uma outra simulação foi realizada com concentrações diferentes de ligações ferro e antiferromagnéticas. Tais concentrações foram introduzidas na equação (1.3.4) com os valores de $\Lambda_{<ij>}$.

Temos assim as figuras (4.3.7), (4.3.8), (4.3.9), (4.3.10), (4.3.11) e (4.3.12), que correspondem à **Histogramas** com concentrações de 70%, 75%, 80%, 85%, 90% e 95%, respectivamente, de ligações ferromagnéticas numa rede 16×16 com 1 milhão de mínimos e 1 milhão de máximos. Pela análise dos histogramas dos mínimos e máximos de energia (que se comportam de uma maneira diferente) podemos notar a influência nos aspectos dos histogramas da transição para-ferromagnética que ocorre em torno de 83% à temperatura nula [16, 17, 18, 19, 20].

4.3 Colapso de Dados

Utilizando os métodos de simulação citados no capítulo anterior, encontramos os valores médios e as larguras dos picos de energia para as redes de dimensão 8×8 com 4000000 mínimos e máximos sendo 1964400 pares e 2035600 ímpares, 16×16 com 2400000 mínimos e máximos sendo 1219200 pares e 1180800 ímpares, 32×32 com 1600000 mínimos e máximos sendo 805200 pares e 794800 ímpares e 64×64 com 1059200 mínimos e máximos sendo 521200 pares e 538000 ímpares dados na tabela (4.3.1).

		ambas	par	ímpar
8×8	$\langle \mathcal{E}_{min} \rangle$	0.17888	0.17887	0.17888
	$\langle \mathcal{E}_{max} \rangle$	0.82119	0.82110	0.82114
	$\langle \Delta \mathcal{E}_{min} \rangle$	0.02032	0.02044	0.02038
	$\langle \Delta \mathcal{E}_{max} \rangle$	0.02033	0.02044	0.02038
16×16	$\langle \mathcal{E}_{min} \rangle$	0.17435	0.17445	0.17440
	$\langle \mathcal{E}_{max} \rangle$	0.82563	0.82556	0.82560
	$\langle \Delta \mathcal{E}_{min} \rangle$	0.00991	0.00992	0.00992
	$\langle \Delta \mathcal{E}_{max} \rangle$	0.00993	0.00991	0.00992
32×32	$\langle \mathcal{E}_{min} \rangle$	0.16994	0.17013	0.17004
	$\langle \mathcal{E}_{max} \rangle$	0.83007	0.82989	0.82998
	$\langle \Delta \mathcal{E}_{min} \rangle$	0.00489	0.00490	0.00489
	$\langle \Delta \mathcal{E}_{max} \rangle$	0.00490	0.00489	0.00489
64×64	$\langle \mathcal{E}_{min} \rangle$	0.16643	0.16637	0.16640
	$\langle \mathcal{E}_{max} \rangle$	0.83358	0.83363	0.83360
	$\langle \Delta \mathcal{E}_{min} \rangle$	0.00252	0.00248	0.00250
	$\langle \Delta \mathcal{E}_{max} \rangle$	0.00252	0.00248	0.00250

Tabela 4.3.1: Valores médios e larguras dos histogramas das energias.

Temos na figura (4.3.13) o histograma dessas redes. Levantamos assim as seguintes questões :

- 1) Qual a localização do pico desses histogramas no limite termodinâmico ($L \rightarrow \infty$)?
- 2) Como se comporta a largura destes picos no mesmo limite?

Para responder a estas questões, fizemos o ajuste dessas curvas para a forma de uma função homogênea generalizada [15]

$$\mathcal{H}(\ell^a(\mathcal{E} - \mathcal{E}_\infty), \ell^{-1}L) = \mathcal{H}((\mathcal{E} - \mathcal{E}_\infty), L) \quad (4.3.1)$$

onde $\mathcal{E}_\infty$ e a são parâmetros a determinar que respondem às duas questões levantadas, respectivamente. O critério é o de colapsar os gráficos da curva $\mathcal{H}$, fazendo uma escala no eixo horizontal. Isto pode ser compreendido escolhendo-se o valor $\ell = L$ para o parâmetro livre da equação (4.3.1). Desta forma, a função $\mathcal{H}$ se expressa como uma função universal, ou seja a mesma para diferentes tamanhos, da combinação $L^a(\mathcal{E} - \mathcal{E}_\infty)$. Esta lei de escala é suposta válida para tamanhos L suficientemente grandes.

Na tabela (4.3.1) nota-se que as larguras dos histogramas diminuem segundo uma lei simples de potência $\Delta\mathcal{E} \sim L^{-1}$, que corresponde ao expoente $a = 1$ na equação (4.3.1). Adotando este valor, encontramos colapso das curvas relativas a $L = 16, 32$ e 64 para

	mínimo	máximo
$\mathcal{E}_\infty$	0,1633	0,8367

conforme a figura (4.3.14). A curva obtida para a rede menor, $L = 8$, não colapsa exatamente nas demais, mostrando que a sequência $L = 16, 32$ e 64 corresponde ao início da região assintótica onde a equação (4.3.1) é suposta válida. Desta forma, não devemos confiar no último dígito dos valores $\mathcal{E}_\infty$ obtidos.

Não podemos afirmar que os mínimos de energia encontrados correspondam a estados fundamentais. De acordo com a evolução dinâmica definida, podemos

garantir apenas que eles sejam estados metaestáveis. Podemos então definir duas diferentes possibilidades.

Primeira, os estados fundamentais têm uma densidade de energia menor que $\mathcal{E}_\infty$. Neste caso eles não estariam sendo detectados em nossas simulações, ou seja, formariam um conjunto de medida nula comparado aos estados metaestáveis.

Segunda, a densidade de energia do estado fundamental coincide com nosso $\mathcal{E}_\infty$. Nesse caso, nossos estados metaestáveis são os verdadeiros estados fundamentais. Resultados de simulações tradicionais de Monte Carlo [21] estimam $\mathcal{E}_\infty \cong 0.15$ enquanto argumentos topológicos teóricos [4] dão um limite inferior de $\mathcal{E}_\infty = 0.125$. Uma revisão recente pode ser apreciada na referência [22].

Figura 4.3.1: Histograma de uma rede 16 × 16, com uma configuração e um caminho.

Figura 4.3.2: Histograma de uma rede 32 × 32, com uma configuração e um caminho.

Figura 4.3.3: Histograma de uma rede 16×16 , com 4 configurações pares e 4 ímpares com 48 caminhos.

Figura 4.3.4: Histograma de uma rede 32×32 , com 4 configurações pares e 4 ímpares com 48 caminhos.

Figura 4.3.5: Histograma de uma rede 16×16 , com 45 configurações pares e 45 ímpares com 1800 caminhos.

Figura 4.3.6: Histograma de uma rede 16×16 , com 1512 configurações pares e 1448 ímpares com 3000 caminhos.

Figura 4.3.7: Concentração de 70% de ligações ferromagnéticas.

Figura 4.3.8: Concentração de 75% de ligações ferromagnéticas.

Figura 4.3.9: Concentração de 80% de ligações ferromagnéticas.

Figura 4.3.10: Concentração de 85% de ligações ferromagnéticas.

Figura 4.3.11: Concentração de 90% de ligações ferromagnéticas.

Figura 4.3.12: Concentração de 95% de ligações ferromagnéticas.

Figura 4.3.13: Histogramas de mínimos e máximos das redes.

Figura 4.3.14: Histograma do colapso das redes.

Capítulo 5

Conclusões

Consideramos o modelo de vidro de spin de Ising $\pm J$ numa rede quadrada com condições de contorno periódicas interagindo apenas com os vizinhos mais próximos, com concentrações de 50% de ligações ferro e 50% antiferromagnéticas. Inspirados na teoria da evolução de Darwin, determinamos uma regra dinâmica que inclui, aleatoriedade, seleção e múltiplas bifurcações sucessivas. Seguindo tal dinâmica, determinamos uma forma de caminhos alternados sucessivamente entre mínimos e máximos de energia para uma análise estatística. Traçamos em seguida seus histogramas e pelo método de escalonamento por tamanhos finitos determinamos o mínimo e o máximo de energia para o limite termodinâmico. Esses histogramas tendem para um determinado valor e suas larguras tendem a zero.

Fizemos também simulações com porcentagens diferentes de ligações ferro e antiferromagnéticas para um tamanho de rede. Vimos uma mudança no comportamento dos histogramas de mínimos e máximos, compatível com a mudança de fase para-ferromagnética.

Como continuação deste trabalho, pretendemos realizar as mesmas simulações em 3 dimensões , além de completar o estudo dos casos com concentrações diferentes de 50% de ligações ferro e antiferromagnéticas na rede.

Apêndice A

Tempo de Cálculo nas diferentes redes e um Programa em linguagem C para uma rede 32×32

Tempo de cálculo para as diferentes dimensões de redes calculados com a seguinte expressão :

$$\text{tempo de cada passo} = \text{tempo total (min)} / n^2 \text{ de passos}$$

8×8	16×16	32×32	64×64
$1,97 \times 10^{-4}$	$9,55 \times 10^{-4}$	$5,56 \times 10^{-3}$	$6,29 \times 10^{-2}$

Em seguida calculamos a seguinte expressão para sabermos o número de atualizações de spins por minuto nas diferentes dimensões de rede:

tempo de cada passo / L^2

8×8	16×16	32×32	64×64
$3,08 \times 10^{-6}$	$3,73 \times 10^{-6}$	$5,43 \times 10^{-6}$	$1,54 \times 10^{-5}$

Exemplo de um programa para uma rede 32×32

```

#include <stdio.h>
#include <ctype.h>
#include <time.h>
#include <math.h>
#include <dos.h>

/*  SPIN-GLASS ENERGY LANDSCAPE
    32 x 32  square lattice          */

void novaConf(), novoWalk(), maximiza(), minimiza(), inverte(),
    media(), hist();
unsigned energia();

const unsigned bit15 = 1<<15;
unsigned long r, rsemente, niter, iter, it, Nwalk, Pconf, Iconf, Nmed, TmaxM, TminM,
    eHmax[2049], eHmin[2049];
unsigned volta, i1, i2, i3, i4, e0, e1, e2, e3, e4, t, l, i, j, s, v, Ea, Eint, Sint, Tmax, Tmin,
    Pt, Emaxsup, Emaxinf, Eminsup, Emininf, bit[16],
    ah[17], bh[17], ch[17], dh[17], av[17], bv[17], cv[17], dv[17],
    aMax[17], bMax[17], cMax[17], dMax[17],
    aMin[17], bMin[17], cMin[17], dMin[17],
    a[17], b[17], c[17], d[17];
int convergiu, mudou;
long int tinic, tempo = 0;
double Emax, Smax, Emin, Smin, SmaxM, SminM, X, Hatual[2049];
FILE *arquivo, *saida, *comandos;
char nomeArq[80];
struct date dia;
struct time hora;

```

Esta rotina sorteia uma configuração de 16×16 para as ligações ferro e antiferromagnéticas.

```
void novaConf() {
    t = 0;
    for (i=1; i<17; i++) {
        r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); l |= r>>24;
        ah[i] = 1; while (l != 0) { l &= l-1; t++; }
        r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); l |= r>>24;
        bh[i] = 1; while (l != 0) { l &= l-1; t++; }
        r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); l |= r>>24;
        ch[i] = 1; while (l != 0) { l &= l-1; t++; }
        r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); l |= r>>24;
        dh[i] = 1; while (l != 0) { l &= l-1; t++; }
        r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); l |= r>>24;
        av[i] = 1; while (l != 0) { l &= l-1; t++; }
        r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); l |= r>>24;
        bv[i] = 1; while (l != 0) { l &= l-1; t++; }
        r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); l |= r>>24;
        cv[i] = 1; while (l != 0) { l &= l-1; t++; }
        r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); l |= r>>24;
        dv[i] = 1; while (l != 0) { l &= l-1; t++; }
    }
}
```

Contagem das configurações pares e ímpares.

```
t &= 1;
if(t) Iconf++; else Pconf++;
novoWalk();
}
```


Escolha de um estado inicial a, b, c e d.

```
void novoWalk() {
  for (i=1; i<17; i++) {
    r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16);
    a[i] = 1|(r>>24);
    r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16);
    b[i] = 1|(r>>24);
    r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16);
    c[i] = 1|(r>>24);
    r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16);
    d[i] = 1|(r>>24);
  }
  maximiza();
  minimiza();
  Nwalk++;
}
```

Cálculo das energias E_0 , E_1 , E_2 e dos spins a serem invertidos para maximizar a energia.

```
void maximiza() {
  Tmax = 0;
  Eint = energia();
  do {
    Ea = Eint;
    do {
      Pt = convergiu = mudou = 0;
      /* atualiza subrede D */
      s = d[1]; i1 = c[1]^dh[1]; i2 = a[1]^av[1]; i3 = a[16]^dv[1];
      i4 = c[1]^ch[1]; i4 = i4 & bit15 ? (i4 << 1) | 1 : i4 << 1;
      e0 = (i1&i2&i3&i4&s)|(~i1&~i2&~i3&~i4&~s);
```

```

e1 = (i1^i2^i3^i4)&^(s^((i1&i2)|(i3&i4)));
e2 = ((i1^i2)&(i3^i4))|((i1^i3)&(i2^i4));
t = e0|e1|(e2&s); convergiu |= t;
l = t; while (l != 0) { l &= l-1; Pt++; }
r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); t &= l|(r>>24);
d[1] ^= t; mudou |= t;
for (i = 2; i <= 16; i++) {
    s = d[i]; i1 = c[i]^dh[i]; i2 = a[i]^av[i]; i3 = a[i-1]^dv[i];
    i4 = c[i]^ch[i]; i4 = i4 & bit15 ? (i4 << 1) | 1 : i4 << 1;
    e0 = (i1&i2&i3&i4&s)|(~i1&~i2&~i3&~i4&~s);
    e1 = (i1^i2^i3^i4)&^(s^((i1&i2)|(i3&i4)));
    e2 = ((i1^i2)&(i3^i4))|((i1^i3)&(i2^i4));
    t = e0|e1|(e2&s); convergiu |= t;
    l = t; while (l != 0) { l &= l-1; Pt++; }
    r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); t &= l|(r>>24);
    d[i] ^= t; mudou |= t;
}

/* atualiza subrede B */
s = b[16]; i1 = a[16]^ah[16]; i2 = c[16]^bv[16]; i3 = c[1]^cv[1];
i4 = a[16] & 1 ? (a[16] >> 1) | bit15 : a[16] >> 1; i4 ^= bh[16];
e0 = (i1&i2&i3&i4&s)|(~i1&~i2&~i3&~i4&~s);
e1 = (i1^i2^i3^i4)&^(s^((i1&i2)|(i3&i4)));
e2 = ((i1^i2)&(i3^i4))|((i1^i3)&(i2^i4));
t = e0|e1|(e2&s); convergiu |= t;
l = t; while (l != 0) { l &= l-1; Pt++; }
r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); t &= l|(r>>24);
b[16] ^= t; mudou |= t;
for (i = 1; i < 16; i++) {
    s = b[i]; i1 = a[i]^ah[i]; i2 = c[i]^bv[i]; j = i+1; i3 = c[j]^cv[j];
    i4 = a[i] & 1 ? (a[i] >> 1) | bit15 : a[i] >> 1; i4 ^= bh[i];
    e0 = (i1&i2&i3&i4&s)|(~i1&~i2&~i3&~i4&~s);
    e1 = (i1^i2^i3^i4)&^(s^((i1&i2)|(i3&i4)));
    e2 = ((i1^i2)&(i3^i4))|((i1^i3)&(i2^i4));
    t = e0|e1|(e2&s); convergiu |= t;
    l = t; while (l != 0) { l &= l-1; Pt++; }
    r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); t &= l|(r>>24);
    b[i] ^= t; mudou |= t;
}

```

```

/* atualiza subrede A */
s = a[16]; i1 = b[16]^ah[16]; i2 = d[16]^av[16]; i3 = d[1]^dv[1];
i4 = b[16]^bh[16]; i4 = i4 & bit15 ? (i4 << 1) | 1 : i4 << 1;
e0 = (i1&i2&i3&i4&s)|(~i1&~i2&~i3&~i4&~s);
e1 = (i1^i2^i3^i4)&~(s^((i1&i2)|(i3&i4)));
e2 = ((i1^i2)&(i3^i4))|((i1^i3)&(i2^i4));
t = e0|e1|(e2&s); convergiu |= t;
l = t; while (l != 0) { l &= l-1; Pt++; }
r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); t &= 1|(r>>24);
a[16] ^= t; mudou |= t;
for (i = 1; i < 16; i++) {
    s = a[i]; i1 = b[i]^ah[i]; i2 = d[i]^av[i]; j = i+1; i3 = d[j]^dv[j];
    i4 = b[i]^bh[i]; i4 = i4 & bit15 ? (i4 << 1) | 1 : i4 << 1;
    e0 = (i1&i2&i3&i4&s)|(~i1&~i2&~i3&~i4&~s);
    e1 = (i1^i2^i3^i4)&~(s^((i1&i2)|(i3&i4)));
    e2 = ((i1^i2)&(i3^i4))|((i1^i3)&(i2^i4));
    t = e0|e1|(e2&s); convergiu |= t;
    l = t; while (l != 0) { l &= l-1; Pt++; }
    r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); t &= 1|(r>>24);
    a[i] ^= t; mudou |= t;
}

```

```

/* atualiza subrede C */
s = c[1]; i1 = d[1]^dh[1]; i2 = b[1]^bv[1]; i3 = b[16]^cv[1];
i4 = d[1] & 1 ? (d[1] >> 1) | bit15 : d[1] >> 1; i4 ^= ch[1];
e0 = (i1&i2&i3&i4&s)|(~i1&~i2&~i3&~i4&~s);
e1 = (i1^i2^i3^i4)&~(s^((i1&i2)|(i3&i4)));
e2 = ((i1^i2)&(i3^i4))|((i1^i3)&(i2^i4));
t = e0|e1|(e2&s); convergiu |= t;
l = t; while (l != 0) { l &= l-1; Pt++; }
r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); t &= 1|(r>>24);
c[1] ^= t; mudou |= t;
for (i = 2; i <= 16; i++) {
    s = c[i]; i1 = d[i]^dh[i]; i2 = b[i]^bv[i]; i3 = b[i-1]^cv[i];
    i4 = d[i] & 1 ? (d[i] >> 1) | bit15 : d[i] >> 1; i4 ^= ch[i];
    e0 = (i1&i2&i3&i4&s)|(~i1&~i2&~i3&~i4&~s);
    e1 = (i1^i2^i3^i4)&~(s^((i1&i2)|(i3&i4)));
    e2 = ((i1^i2)&(i3^i4))|((i1^i3)&(i2^i4));
    t = e0|e1|(e2&s); convergiu |= t;
}

```

```

    l = t; while (l != 0) { l &= l-1; Pt++; }
    r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); t &= l|(r>>24);
    c[i] ^= t; mudou |= t;
}
if(mudou) {Tmax++; Hatual[Tmax] += Pt;}
} while(convergiu);
Eint = energia();
inverte();
} while(Ea!=Eint);
Sint = 0;
for(i=1; i<=16; i++) {
    l = a[i]^aMax[i]; while (l != 0) { l &= l-1; Sint++; } aMax[i] = a[i];
    l = b[i]^bMax[i]; while (l != 0) { l &= l-1; Sint++; } bMax[i] = b[i];
    l = c[i]^cMax[i]; while (l != 0) { l &= l-1; Sint++; } cMax[i] = c[i];
    l = d[i]^dMax[i]; while (l != 0) { l &= l-1; Sint++; } dMax[i] = d[i];
}
Smax = 1.0 - Sint/512.0;
Emax = Eint/2048.0;
if(Eint > Emaxsup) Emaxsup = Eint;
if(Eint < Emaxinf) Emaxinf = Eint;
}

```

Cálculo das energias E_2 , E_3 , E_4 e dos spins a serem invertidos para minimizar a energia.

```

void minimiza() {
    Tmin = 0;
    Eint = energia();
    do {
        Ea = Eint;
        do {
            Pt = convergiu = mudou = 0;
            /* atualiza subrede D */
            s = d[1]; i1 = c[1]^dh[1]; i2 = a[1]^av[1]; i3 = a[16]^dv[1];

```

```

i4 = c[1]^ch[1]; i4 = i4 & bit15 ? (i4 << 1) | 1 : i4 << 1;
e4 = (i1&i2&i3&i4&~s)|(~i1&~i2&~i3&~i4&s);
e3 = (i1^i2^i3^i4)&(s^((i1&i2)|(i3&i4)));
e2 = ((i1^i2)&(i3^i4))|((i1^i3)&(i2^i4));
t = e4|e3|(e2&s); convergiu |= t;
l = t; while (l != 0) { l &= l-1; Pt++; }
r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); t &= 1|(r>>24);
d[1] ^= t; mudou |= t;
for (i = 2; i <= 16; i++) {
    s = d[i]; i1 = c[i]^dh[i]; i2 = a[i]^av[i]; i3 = a[i-1]^dv[i];
    i4 = c[i]^ch[i]; i4 = i4 & bit15 ? (i4 << 1) | 1 : i4 << 1;
    e4 = (i1&i2&i3&i4&~s)|(~i1&~i2&~i3&~i4&s);
    e3 = (i1^i2^i3^i4)&(s^((i1&i2)|(i3&i4)));
    e2 = ((i1^i2)&(i3^i4))|((i1^i3)&(i2^i4));
    t = e4|e3|(e2&s); convergiu |= t;
    l = t; while (l != 0) { l &= l-1; Pt++; }
    r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); t &= 1|(r>>24);
    d[i] ^= t; mudou |= t;
}

/* atualiza subrede B */
s = b[16]; i1 = a[16]^ah[16]; i2 = c[16]^bv[16]; i3 = c[1]^cv[1];
i4 = a[16] & 1 ? (a[16] >> 1) | bit15 : a[16] >> 1; i4 ^= bh[16];
e4 = (i1&i2&i3&i4&~s)|(~i1&~i2&~i3&~i4&s);
e3 = (i1^i2^i3^i4)&(s^((i1&i2)|(i3&i4)));
e2 = ((i1^i2)&(i3^i4))|((i1^i3)&(i2^i4));
t = e4|e3|(e2&s); convergiu |= t;
l = t; while (l != 0) { l &= l-1; Pt++; }
r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); t &= 1|(r>>24);
b[16] ^= t; mudou |= t;
for (i = 1; i < 16; i++) {
    s = b[i]; i1 = a[i]^ah[i]; i2 = c[i]^bv[i]; j = i+1; i3 = c[j]^cv[j];
    i4 = a[i] & 1 ? (a[i] >> 1) | bit15 : a[i] >> 1; i4 ^= bh[i];
    e4 = (i1&i2&i3&i4&~s)|(~i1&~i2&~i3&~i4&s);
    e3 = (i1^i2^i3^i4)&(s^((i1&i2)|(i3&i4)));
    e2 = ((i1^i2)&(i3^i4))|((i1^i3)&(i2^i4));
    t = e4|e3|(e2&s); convergiu |= t;
    l = t; while (l != 0) { l &= l-1; Pt++; }
    r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); t &= 1|(r>>24);

```

```

    b[i] ^= t; mudou |= t;
}

    /* atualiza subrede A */
s = a[16]; i1 = b[16]^ah[16]; i2 = d[16]^av[16]; i3 = d[1]^dv[1];
i4 = b[16]^bh[16]; i4 = i4 & bit15 ? (i4 << 1) | 1 : i4 << 1;
e4 = (i1&i2&i3&i4&~s)|(~i1&~i2&~i3&~i4&s);
e3 = (i1^i2^i3^i4)&(s^((i1&i2)|(i3&i4)));
e2 = ((i1^i2)&(i3^i4))|((i1^i3)&(i2^i4));
t = e4|e3|(e2&s); convergiu |= t;
l = t; while (l != 0) { l &= l-1; Pt++; }
r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); t &= l|(r>>24);
a[16] ^= t; mudou |= t;
for (i = 1; i < 16; i++) {
    s = a[i]; i1 = b[i]^ah[i]; i2 = d[i]^av[i]; j = i+1; i3 = d[j]^dv[j];
    i4 = b[i]^bh[i]; i4 = i4 & bit15 ? (i4 << 1) | 1 : i4 << 1;
    e4 = (i1&i2&i3&i4&~s)|(~i1&~i2&~i3&~i4&s);
    e3 = (i1^i2^i3^i4)&(s^((i1&i2)|(i3&i4)));
    e2 = ((i1^i2)&(i3^i4))|((i1^i3)&(i2^i4));
    t = e4|e3|(e2&s); convergiu |= t;
    l = t; while (l != 0) { l &= l-1; Pt++; }
    r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); t &= l|(r>>24);
    a[i] ^= t; mudou |= t;
}

    /* atualiza subrede C */
s = c[1]; i1 = d[1]^dh[1]; i2 = b[1]^bv[1]; i3 = b[16]^cv[1];
i4 = d[1] & 1 ? (d[1] >> 1) | bit15 : d[1] >> 1; i4 ^= ch[1];
e4 = (i1&i2&i3&i4&~s)|(~i1&~i2&~i3&~i4&s);
e3 = (i1^i2^i3^i4)&(s^((i1&i2)|(i3&i4)));
e2 = ((i1^i2)&(i3^i4))|((i1^i3)&(i2^i4));
t = e4|e3|(e2&s); convergiu |= t;
l = t; while (l != 0) { l &= l-1; Pt++; }
r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); t &= l|(r>>24);
c[1] ^= t; mudou |= t;
for (i = 2; i <= 16; i++) {
    s = c[i]; i1 = d[i]^dh[i]; i2 = b[i]^bv[i]; i3 = b[i-1]^cv[i];
    i4 = d[i] & 1 ? (d[i] >> 1) | bit15 : d[i] >> 1; i4 ^= ch[i];
    e4 = (i1&i2&i3&i4&~s)|(~i1&~i2&~i3&~i4&s);
    e3 = (i1^i2^i3^i4)&(s^((i1&i2)|(i3&i4)));

```

```

        e2 = ((i1^i2)&(i3^i4))|((i1^i3)&(i2^i4));
        t = e4|e3|(e2&s); convergiu |= t;
        l = t; while (l != 0) { l &= l-1; Pt++; }
        r+=(r<<1)+(r<<16); l=(r>>24)<<8; r+=(r<<1)+(r<<16); t &= l|(r>>24);
        c[i] ^= t; mudou |= t;
    }
    if(mudou) {Tmin++; Hatual[Tmin] += Pt;}
} while(convergiu);
Eint = energia();
inverte();
} while(Ea!=Eint);
Sint = 0;
for(i=1; i<=16; i++) {
    l = a[i]^aMin[i]; while (l != 0) { l &= l-1; Sint++; } aMin[i] = a[i];
    l = b[i]^bMin[i]; while (l != 0) { l &= l-1; Sint++; } bMin[i] = b[i];
    l = c[i]^cMin[i]; while (l != 0) { l &= l-1; Sint++; } cMin[i] = c[i];
    l = d[i]^dMin[i]; while (l != 0) { l &= l-1; Sint++; } dMin[i] = d[i];
}
Smin = 1.0 - Sint/512.0;
Emin = Eint/2048.0;
if(Eint > Eminsup) Eminsup = Eint;
if(Eint < Emininf) Emininf = Eint;
}

void inverte() {
    for(i=1;i<17;i++) {
        a[i] = ~a[i];
        b[i] = ~b[i];
        c[i] = ~c[i];
        d[i] = ~d[i];
    }
}

```

Preparação do arquivo de saída.

```

void media() {
    fprintf(saida, "\n\n %lu configuracoes pares, %lu impares e %lu walks",
            Pconf, Iconf, Nwalk);
    fprintf(saida, "\n\n Histograma dos %lu maximos locais percorridos\n", Nmed);
    fprintf(saida, "\n      i      Emax      H");
    for(i=Emaxinf; i<=Emaxsup; i++)
        fprintf(saida, "\n %5u %8.5lf %12lu", i, i/2048.0, eHmax[i]);
    fprintf(saida, "\n\n Histograma dos %lu minimos locais percorridos\n", Nmed);
    fprintf(saida, "\n      i      Emin      H");
    for(i=Emininf; i<=Eminsup; i++)
        fprintf(saida, "\n %5u %8.5lf %12lu", i, i/2048.0, eHmin[i]);
    fprintf(saida, "\n\n Tempos medios:");
    X = TmaxM; X /= Nmed;
    fprintf(saida, "\n      do minimo ao maximo seguinte = %8.5lf", X);
    X = TminM; X /= Nmed;
    fprintf(saida, "\n      do maximo ao minimo seguinte = %8.5lf", X);
    fprintf(saida, "\n\n Recobrimentos (em modulo) medios:");
    fprintf(saida, "\n      entre maximos consecutivos = %8.5lf", SmaxM/Nmed);
    fprintf(saida, "\n      entre minimos consecutivos = %8.5lf\n\n", SminM/Nmed);
    for(i=0; i<2049; i++) if((Hatual[i] /= (2*Nmed)) != 0.0) j = i;
    fprintf(saida, "\n\n Numero medio de spins passíveis de inversao a cada
            passo");
    fprintf(saida, "\n      passo      N");
    for(i=1; i<=j; i++)
        fprintf(saida, "\n %7u      %8.5lf", i, Hatual[i]);
    fprintf(saida, "\n\n");
}

void hist() {
    fwrite(eHmax, 4, 2049, arquivo);
    fwrite(eHmin, 4, 2049, arquivo);
    fclose(arquivo);
}

```


Cálculo dos valores mínimos e máximos de energia.

```

unsigned energia() {
    unsigned E,E1,E2;
    E = 0;

                                /* subrede D */
    s = d[1]; i1 = c[1]^dh[1]; i2 = a[1]^av[1]; i3 = a[16]^dv[1];
    i4 = c[1]^ch[1]; i4 = i4 & bit15 ? (i4 << 1) | 1 : i4 << 1;
    l = s^i1; while (l != 0) { l &= l-1; E++; }
    l = s^i2; while (l != 0) { l &= l-1; E++; }
    l = s^i3; while (l != 0) { l &= l-1; E++; }
    l = s^i4; while (l != 0) { l &= l-1; E++; }
    for (i = 2; i <= 16; i++) {
        s = d[i]; i1 = c[i]^dh[i]; i2 = a[i]^av[i]; i3 = a[i-1]^dv[i];
        i4 = c[i]^ch[i]; i4 = i4 & bit15 ? (i4 << 1) | 1 : i4 << 1;
        l = s^i1; while (l != 0) { l &= l-1; E++; }
        l = s^i2; while (l != 0) { l &= l-1; E++; }
        l = s^i3; while (l != 0) { l &= l-1; E++; }
        l = s^i4; while (l != 0) { l &= l-1; E++; }
    }

                                /* subrede B */
    s = b[16]; i1 = a[16]^ah[16]; i2 = c[16]^bv[16]; i3 = c[1]^cv[1];
    i4 = a[16] & 1 ? (a[16] >> 1) | bit15 : a[16] >> 1; i4 ^= bh[16];
    l = s^i1; while (l != 0) { l &= l-1; E++; }
    l = s^i2; while (l != 0) { l &= l-1; E++; }
    l = s^i3; while (l != 0) { l &= l-1; E++; }
    l = s^i4; while (l != 0) { l &= l-1; E++; }
    for (i = 1; i < 16; i++) {
        s = b[i]; i1 = a[i]^ah[i]; i2 = c[i]^bv[i]; j = i+1; i3 = c[j]^cv[j];
        i4 = a[i] & 1 ? (a[i] >> 1) | bit15 : a[i] >> 1; i4 ^= bh[i];
        l = s^i1; while (l != 0) { l &= l-1; E++; }
        l = s^i2; while (l != 0) { l &= l-1; E++; }
        l = s^i3; while (l != 0) { l &= l-1; E++; }
        l = s^i4; while (l != 0) { l &= l-1; E++; }
    }
    E1 = E; E = 0;
}

```

```

/* subrede A */
s = a[16]; i1 = b[16]^ah[16]; i2 = d[16]^av[16]; i3 = d[1]^dv[1];
i4 = b[16]^bh[16]; i4 = i4 & bit15 ? (i4 << 1) | 1 : i4 << 1;
l = s^i1; while (l != 0) { l &= l-1; E++; }
l = s^i2; while (l != 0) { l &= l-1; E++; }
l = s^i3; while (l != 0) { l &= l-1; E++; }
l = s^i4; while (l != 0) { l &= l-1; E++; }
for (i = 1; i < 16; i++) {
    s = a[i]; i1 = b[i]^ah[i]; i2 = d[i]^av[i]; j = i+1; i3 = d[j]^dv[j];
    i4 = b[i]^bh[i]; i4 = i4 & bit15 ? (i4 << 1) | 1 : i4 << 1;
    l = s^i1; while (l != 0) { l &= l-1; E++; }
    l = s^i2; while (l != 0) { l &= l-1; E++; }
    l = s^i3; while (l != 0) { l &= l-1; E++; }
    l = s^i4; while (l != 0) { l &= l-1; E++; }
}

/* subrede C */
s = c[1]; i1 = d[1]^dh[1]; i2 = b[1]^bv[1]; i3 = b[16]^cv[1];
i4 = d[1] & 1 ? (d[1] >> 1) | bit15 : d[1] >> 1; i4 ^= ch[1];
l = s^i1; while (l != 0) { l &= l-1; E++; }
l = s^i2; while (l != 0) { l &= l-1; E++; }
l = s^i3; while (l != 0) { l &= l-1; E++; }
l = s^i4; while (l != 0) { l &= l-1; E++; }
for (i = 2; i <= 16; i++) {
    s = c[i]; i1 = d[i]^dh[i]; i2 = b[i]^bv[i]; i3 = b[i-1]^cv[i];
    i4 = d[i] & 1 ? (d[i] >> 1) | bit15 : d[i] >> 1; i4 ^= ch[i];
    l = s^i1; while (l != 0) { l &= l-1; E++; }
    l = s^i2; while (l != 0) { l &= l-1; E++; }
    l = s^i3; while (l != 0) { l &= l-1; E++; }
    l = s^i4; while (l != 0) { l &= l-1; E++; }
}
E2 = E;
if(E1!=E2) printf("\n\n***** erro na energia *****\n\n");
return(E);
}

```

Programa principal.

```
main() {
    char *texto,resposta,formato[80],leitura[80];
    unsigned tam;
```

Entrada de dados.

```
    /* pede arquivo de comandos (ou stdin) */
    do {
        printf(" D o nome do arquivo de comandos [teclado] ");
        gets(nomeArq);
        if (*nomeArq == '\0') comandos = stdin;
        else {
            comandos = fopen(nomeArq,"r");
            if (comandos == NULL) printf("\n arquivo no encontrado. \a");
        }
    } while (comandos == NULL);
    volta = 1;
    while(1) {
        if(volta) {
            tempo = 0; niter = 0;
            if(comandos == stdin)
                printf("\n D uma semente inteira < entre 1 e 4294967295 > ");
            fscanf(comandos,"%lu",&rsemente); while(getc(comandos) != '\n');
            rsemente |= 1;
            r = rsemente;
            for(i=0;i<2049;i++) {
                eHmax[i] = 0L;
                eHmin[i] = 0L;
            }
            Nmed = 0L;
```

```

Emaxsup = Eminsup = 0;
Emaxinf = Emininf = 2048;
TmaxM = TminM = 0L;
SmaxM = SminM = 0.0;
for(i=0;i<16;i++) bit[i] = 1<<i;
for(i=0;i<2049;i++) Hatual[i] = 0.0;
Nwalk = Pconf = Iconf = 0L;
novaConf();

```

Saída de dados.

```

/* prepara arquivo de saida */
do {
    if(comandos == stdin)
        printf(" D o nome do arquivo de saida [SG16-LS.PRN] ");
        strcpy(nomeArq,"sg16-ls.prn");
    fgets(leitura,79,comandos); sscanf(leitura,"%s",nomeArq);
    if((saida = fopen(nomeArq,"a")) == NULL) {
        comandos = stdin;
        texto = "\n Erro na abertura do arquivo !"
            " Tome as devidas providncias e ...\n\a";
        printf(texto);
    }
} while(saida == NULL);
/* prepara arquivo do histograma */
do {
    if(comandos == stdin)
        printf(" e o do arquivo do histograma [SG16-LS.HST] ");
        strcpy(nomeArq,"sg16-ls.hst");
    fgets(leitura,79,comandos); sscanf(leitura,"%s",nomeArq);
    if((arquivo = fopen(nomeArq,"wb")) == NULL) {
        comandos = stdin;
        texto = "\n Erro na abertura do arquivo do histograma !"
            " Tome as devidas providncias e ...\n\a";
        printf(texto);
    }
}

```

Comandos de tela.

```

} while(arquivo == NULL);
getdate(&dia); gettime(&hora);
strcpy(formato, "\n\n SPIN-GLASS LANDSCAPE   %d/%d/%d   %02d:%02d");
strcat(formato, "\n   (gera e muda configuracoes)\n\n");
fprintf(saida, formato, dia.da_day, dia.da_mon, dia.da_year, hora.ti_hour,
        hora.ti_min);
strcpy(formato, " Rede 32 x 32   semente inicial %lu\n");
fprintf(saida, formato, rsemente);
}
volta = 0;
strcpy(formato, "\n %8lu passos");
strcat(formato, "   tempo = %2ld:%2ld:%2ld\n");
printf(formato, niter, tempo / 3600, (tempo / 60) % 60, tempo % 60);
printf("   Emax = %8.5lf   Tmax = %8u   Smax = %8.5lf\n", Emax, Tmax, Smax);
printf("   Emin = %8.5lf   Tmin = %8u   Smin = %8.5lf\n", Emin, Tmin, Smin);
do {
    if(comandos == stdin) printf(" Quantos passos ? ");
    fgets(leitura, 79, comandos);
    tam = strlen(leitura);
    if((tam == 0) || (tam > 5) || (strcspn(leitura, "1234567890")))
        it = 0;
    else
        sscanf(leitura, "%lu", &it);
    if(it == 0) {
        if(comandos == stdin)
            printf(" Sai Volta Zera novoWalk novaConf :");
        resposta = getc(comandos); while(getc(comandos) != '\n');
        resposta = tolower(resposta);
        switch (resposta) {
            case 'v': media();
                        getdate(&dia); gettime(&hora);
                        strcpy(formato, "\n           %d/%d/%d           %02d:%02d");

```

```

        fprintf(saida,formato,dia.da_day,dia.da_mon,dia.da_year,
                hora.ti_hour,hora.ti_min);
        fclose(saida); volta = 1; it = 1; hist(); break;
case 's': media();
            getdate(&dia); gettime(&hora);
            strcpy(formato,"\n      %d/%d/%d      %02d:%02d");
            fprintf(saida,formato,dia.da_day,dia.da_mon,dia.da_year,
                    hora.ti_hour,hora.ti_min);
            fclose(saida); hist(); exit(); break;

case 'w': novoWalk(); break;
case 'c': novaConf(); break;
case 'z': media();
            for(i=0;i<2049;i++) {
                eHmax[i] = 0L;
                eHmin[i] = 0L;
            }
            Nmed = 0L;
            Emaxsup = Eminsup = 0;
            Emaxinf = Emininf = 2048;
            TmaxM = TminM = 0L;
            SmaxM = SminM = 0.0;
            break;
    default : resposta = '\r';
}
}
} while (it == 0);
if(volta == 0) {
    iter = niter+1;
    niter = niter + it;
    printf("      processamento at o passo %6lu",niter);
    time(&tinic);
    for (it = iter; it <= niter; it++) {

```

Cálculo das contagens de energias mínima e máxima e da semelhança $|S|$.

```
    maximiza();
    eHmax[Eint]++;
    minimiza();
    eHmin[Eint]++;
    Nmed++;
    TmaxM += Tmax;
    TminM += Tmin;
    SmaxM += fabs(Smax);
    SminM += fabs(Smin);
}
time(&tempo); tempo -= tinic;
}
}
}
```

Referências Bibliográficas

- [1] Zorn R., Hermann H. J. and Rebbi C., *Comp. Phys. Commun.* **23**, 337 (1981).
- [2] Jacobs L. and Rebbi C., *J. Comput. Phys.* **41**, 203 (1981).
- [3] Cheung H. F., and W. L. McMillan, *J. Phys.* **C16**, 7027 (1983).
- [4] Vannimenus J. and Toulouse G., *J. Phys.* **C10**, L537 (1977).
- [5] Kirkpatrick S., *Phys. Rev.* **B16**, 4630 (1977).
- [6] Kolan A. J. and R. G. Palmer, *J. Appl. Phys.* **53**, 2198 (1982).
- [7] Ising E.; *Z. Physik* **31**, 253 (1925).
- [8] Onsager L.; *Phys. Rev.* **65**, 117 (1944).
- [9] de Oliveira P. M. C., Moss de Oliveira S. M. and de Queiroz S. L. A.; *Physica* **A175**, (3), 345 (1991).
- [10] Darwin C., *The origin of Species*, Reprinted Penguin (London 1859).
- [11] Oed W., *Appl. Informatics* **7**, 358 (1982).
- [12] de Oliveira P. M. C., *Computing Boolean Statistical Models*, World Scientific (Singapore 1991).
- [13] Herrmann H. J., *J. Stat. Phys.* **45**, 145 (1986).
- [14] Kauffman S.A., in *The Economy as an Evolving Complex System*, eds. Anderson P.W., Arrow K.J. and Pines D., Santa Fe Institute Studies in the Sciences of Complexity, Addison Wesley (New York 1988).

- [15] Stanley H. E., Chang T. S., Harbus F. and Liu L. L., in *Proceedings of the International Summer School Enrico Fermi*, Course LIX, ed. Müller K. A. and Rigamonti A., Varenna, Italy (North Holland, Amsterdam 1976).
- [16] Maynard R. and Rammal, *J. Physique Lettres* **43**, L347 (1982).
- [17] Nishimori H., *Prog. Theor. Phys.* **66**, 1169 (1981).
- [18] Kitatani H. and Oguchi T., *J. Phys. Soc. Japan* **59**, 3823 (1990).
- [19] Kitatani H. and Oguchi T., *J. Phys. Soc. Japan* **61**, 1598 (1992).
- [20] Kitatani H., *J. Phys. Soc. Japan* **61** (1992).
- [21] Binder K. and Young A. P., *Reviews of Modern Physics* **58**, 801 (1986).
- [22] Vogel E. E., Cartes J., Contreras S. and Villegas J., *Phys. Rev. B*, submitted (1993).