Paralelização do Algoritmo de Geração de Redes Aleatórias Contínuas *

Gustavo Romano, Nicolas Maillard
Instituto de Informática
Universidade Federal do Rio Grande do Sul
Caixa Postal 15.064 – CEP 91501-970
Porto Alegre – RS – Brasil
{gromano,nicolas}@inf.ufrgs.br

Ricardo Vargas Dorneles
Departamento de Informática
Universidade de Caxias do Sul (UCS)
R. F. Getúlio Vargas, 1130 – CEP 95070-560
Caxias do Sul – RS – Brasil
rvdornel@ucs.br

Abstract

This paper presents the parallelization of an algorithm to generate Continuous Random Networks (CRNs). Such algorithm was proposed by a researcher of the Institute of Physics of UFRGS and it will be used to look for new materials harder than diamond. The results here presented show that the parallel version of the algorithm reduces significantly the execution time, in this manner, enlarging the possibility of finding such materials.

1. Introdução

Recentemente, uma das áreas da ciência dos materiais em que se tem tido um grande interesse é a nanotecnologia. A possibilidade de se obter compostos de propriedades diversas apenas por diferentes configurações estruturais, abre inúmeras possibilidades de pesquisas. Nessa perspectiva, o elemento que tem se destacado é o carbono. Diamante, grafite e nanotubos são alguns exemplos de materiais formados unicamente por esse elemento e que possuem propriedades físicas distintas determinadas pela forma com que os átomos arranjam-se.

Das diferentes fases que o carbono pode assumir, a amorfa tem-se mostrado de alta utilidade prática. Filmes finos com *Diamond-like carbon* são um bom exemplo disso. Eles exibem alto grau de dureza, sendo empregados para o recobrimento de peças ópticas.

Um modelo relativamente realístico para descrever uma grande gama desses materiais não-cristalinos é o das redes aleatórias contínuas (CRN), proposto por [6], originalmente para explicar a organização molecular no vidro. Trata-se da idealização de que o material não possui *dangling-bonds*, isto é, que cada átomo realiza o número de ligações que sua hibridização permite. Dessa forma, tem-se um material sem nenhum tipo de defeito químico. A estrutura possuirá uma ordem de curta distância, mas não apresentará periodicidade cristalina.

Embora as CRN tenham sido utilizadas com sucesso para a descrição de semicondutores amorfos desde meados da década de 1930 [1], os algoritmos que são usados para gerar essas redes normalmente funcionam apenas para aquelas formadas somente por átomos com hibridização sp^3 (como é o caso do algoritmo WWW [5]).

Na busca de novos materiais, pesquisadores do Instituto de Física da Universidade Federal do Rio Grande do Sul (UFRGS) propuseram um novo algoritmo para a geração de redes aleatórias contínuas formadas de átomos de carbono com hibridização mista sp^3/sp^2. Estas redes podem propor novos materiais cuja dureza e módulo volumétrico sejam significativamente superiores aos do diamante [4].

O algoritmo proposto gerou bons resultados, porém o tempo de cada execução foi muito grande [2]. Visando contornar esse problema, esse trabalho se propõe a apresentar uma proposta de paralelização para o mesmo.

Esse trabalho está dividido da seguinte forma: na seção 2 é apresentado o algoritmo seqüencial proposto para geração de CRN; na seção 3 é apresentada a paralelização feita nesse trabalho; por fim, na seção 4 são apresentadas algumas conclusões e os trabalhos futuros.

2. Algoritmo Proposto

O algoritmo seqüencial proposto em [2] para gerar CRN formadas por átomos com hibridização mista (sp^3/sp^2) consiste em posicionar de forma aleatória no espaço certa quantidade de átomos. Após isso, através do método *Simulated Annealing* (SA) altera-se aleatoriamente as posições dos mesmos até convergir em um sistema estável que atenda as restrições impostas. O sistema baseado no algoritmo foi desenvolvido em linguagem C, seu pseudocódigo está representado na Figura 1 e será explicado em detalhes nos próximos parágrafos.

* Trabalho financiado pelo CNPq

```
 1: le_parametros();{lê parâmetros de entrada do sistema}
 2: posiciona_atomos();{posiciona os átomos aleatoriamente}
 3: do_cells();{coloca cada átomo em sua célula do sistema}
 4: hibri_all();{hibridiza todos os átomos}
 5: tote();{calcula a energia total do sistema}
 6: temperatura = temp_ini;{inicializa a temperatura}
 7: for iter = 0 to MAX_ITER − 1 do
 8:     if iter%RESIZE_UNI_EACH == 0 then
 9:         muda_tamanho_universo();{muda tam. do universo}
10:     end if
11:     at = atomos[rand(NATM)];{pega um átomo aleat.}
12:     E_local_old = local_energy(at);{calc. energia local}
13:     Hibri_old = hibri_cost(at);{calc. custo de hibrid.}
14:     desloca_atomo(at);{desloca átomo direções x, y e z}
15:     E_local_new = local_energy(at);{calc. energia local}
16:     Hibri_new = hibri_cost(at);{calc. custo de hibrid.}
17:     dE = E_local_new - E_local_old;{calc. var. de energia}
18:     dH = Hibri_new - Hibri_old;
19:     dfc = (1.0 - COST_HIBRI)*(dE) + COST_HIBRI*(dH);
        {calc. var. da função custo}
20:     if aceita(dfc, temperatura) then
21:         atualiza_posicao();{atualiza a posição do átomo}
22:     else
23:         desfaz_movimento();{volta posição anterior}
24:     end if
25:     temperatura = temperatura * λ {diminui a temp.}
26: end for
```

Figura 1. Alg. seqüenc. para geração de CRN

O algoritmo começa pela leitura do arquivo que contém os parâmetros da simulação (linha 1). Nele estão contidas informações tais como: número de átomos, átomos com hibridização sp^2 e sp^3 desejados, densidade do sistema, número de passos do SA e temperatura inicial e final. Em seguida, os átomos informados nos parâmetros são posicionados de forma aleatória no espaço (linha 2) e alocados nas suas células correspondentes (linha 3).

A seguir, hibridiza-se todos os átomos (linha 4), isto é, cria-se ligações entre os átomo que estejam próximos entre si, e calcula-se a energia total do sistema (linha 5).

Após posicionados e calculados os valores da distribuição inicial, uma temperatura inicial é definida (linha 6) e a seguir um conjunto de iterações (linhas 7 a 26) de SA são executadas.

A cada iteração um átomo é selecionado aleatoriamente (linha 11), e a energia local (linha 12) de onde ele está inserido é calculada (entende-se como energia local a energia da célula onde o átomo está inserido e das células vizinhas a ela). Além da energia local, o custo de hibridização também é calculado (linha 13). Para esse cálculo, são verificados os valores de hibridização de todos os átomos do sistema e comparados com os informados no arquivo de parâmetros. Caso os valores sejam diferentes, um custo é aplicado a essa diferença.

O passo seguinte consiste em fazer uma pequena perturbação no sistema através da alteração aleatória da posição do átomo selecionado (linha 24). Após isso, a nova energia local (linha 15) e o novo custo de hibridização (linha 16) são calculados, bem como a variação desses valores (linhas 17 e 18). A partir da variação dos valores, a variação da função custo é calculada (linha 19) e através do critério de Metropolis [1] (linha 20) verifica-se se esse movimento é aceito ou não. Se for aceito, os novos valores são atualizados, se não, volta-se para os valores anteriores. Por fim, a temperatura do sistema é atualizada (linha 25) e finaliza-se o laço.

Cabe notar que na linha 8 existe um teste que verifica se a iteração atual corresponde a uma iteração de mudança do tamanho do universo. Caso seja, essa operação, que consiste em alterar o tamanho do espaço onde os átomos estão inseridos, é executada. Uma vez feito isso, a célula em que cada átomo está inserido, bem como as ligações entre eles podem ser alteradas. Essas alterações gerarão uma nova função custo, que será avaliada pelo critério de Metropolis. Caso seja aceita, a nova configuração fará parte do sistema, caso contrário, volta-se à configuração anterior.

Simulações feitas com o programa geram boas redes, porém, devido às características inerentes do SA (necessidade de um grande número de passos até a convergência), essas simulações foram muito demoradas, chegando à casa de semanas para redes de 128 átomos. Para que existam maiores chances de encontrar um material que atenda as propriedades desejadas, são necessárias novas simulações com um universo superior a 1024 átomos, o que levaria meses de simulação.

3. Paralelização

Para contornar o problema de tempo apresentado na seção anterior, foi proposta a paralelização do algoritmo de geração das CRN. A primeira abordagem adotada foi a de divisão de domínio, isto é, cada processo ficou responsável pelo processamento de certa região do espaço. Devido ao fato de que para execução de cálculos locais são necessárias informações de átomos localizados até duas células vizinhas, optou-se por criar uma "região neutra" nas fronteiras de cada processo. Com isso, evita-se que átomos dessas regiões sejam modificados e o resultado interfira nos dados de outros processos.

A cada conjunto de passos os processos trocam informações sobre a posição de seus átomos e, para evitar que as regiões neutras nunca sejam exploradas, a região de responsabilidade de cada processo também é alterada. A Figura 2 ilustra a divisão do domínio (cada cor representa a área de atuação de cada processador), as regiões

1 O critério de Metropolis diz que: se a variação da função custo for negativa, isso é, melhore o resultado, essa será aceita. Caso contrário, ela será aceita com uma probabilidade $e^{\frac{-\Delta C}{T}}$, onde ΔC é a variação da função custo e T é a temperatura atual do sistema.

neutras (área cinza) e o funcionamento do sistema de deslocamento das células.

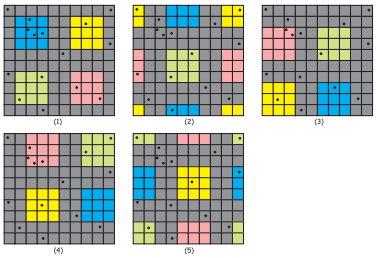

Figura 2. Funcionamento do deslocamento

Além de evitar que átomos posicionados nas regiões neutras não sejam explorados, o sistema de deslocamento permite que o número de mensagens trocadas entre os processos seja muito menor. Isso se deve ao fato de que as mensagens precisam ser enviadas apenas para uma direção, visto que os dados da fronteira da outra direção já pertenciam ao mesmo processo. Cabe lembrar também que, esse sistema de deslocamento torna-se eficiente para esse problema, devido ao fato do acesso aos dados ser cíclico, isso é, os átomos posicionados nas últimas colunas fazem ligações com os átomos localizados nas primeiras colunas.

Na figura 3 estão ilustradas as comunicações necessárias para troca de informações num sistema convencional e num sistema com deslocamento. Nota-se que, para casos em que existe divisão em apenas um eixo (a), o número de mensagens cai de 2 para 1; no caso de divisões em dois eixos (b), de 8 para 3; e no caso de divisões nos três eixos (c), de 26 para apenas 7.

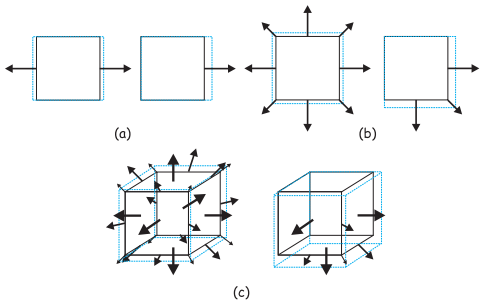

Figura 3. Mensagens trocadas no sistema convencional e naquele com deslocamento

A abordagem de paralelização adotada não se enquadra diretamente em nenhuma das presentes na literatura, mas, pode ser considerada uma adaptação do método de Múltiplas Cadeias de Markov Síncronas Interativas [3]. A diferença entre elas está relacionada à área de atuação, sendo que, na apresentada na literatura todos os processos podem modificar todas as variáveis e, na adotada, a atuação de cada processo é limitada a certas variáveis.

```
1:  if rank==0 then
2:      le_parametros();{lê parâmetros de entrada do sistema}
3:      posiciona_atomos();{posiciona os átomos}
4:  end if
5:  temperatura = temp_ini;{inicializa a temperatura}
6:  Max_iters = (Max_Steps/size/Pas_Independ)
7:  for iter = 0 to Max_iters do
8:      if iter%RESIZE_UNI_EACH == 0 then
9:          recolhe_dados() {procs. enviam dados proc. 0}
10:         muda_tamanho_universo();{muda tam. do universo}
11:         distribui_dados();{Distrib. pos. átms p/ procs.}
12:         do_cells();{coloca cada átomo em sua célula}
13:         hibri_all();{refaz a hibridização dos átomos}
14:     else
15:         troca_dados();{troca pos. vizinhos.}
16:         hibri_novos();{hibri átomos receb}
17:         desloca_area();{desloc. área atu.}
18:     end if
19:     tote();{calcula a energia total do sistema}
20:     for passo = 0 to Passos_Independ do
21:         at = atomos[rand(NATM)];{pega um átomo aleat.}
22:         E_local = local_energy(at);{calc. a energia local}
23:         Hibri = hibri_cost(at);{calc. o custo de hibrid.}
24:         desloca_atomo(at);{desloca átomo direções x, y e z}
25:         E_local_new = local_energy(at);{calc. energia local}
26:         Hibri_new = hibri_cost(at);{calc. custo de hibrid.}
27:         dE = E_new - E_old;
28:         dH = Hibri_new - Hibri_old;{calc. var. custo hibrid.}
29:         dfc = (1.0 - COST_HIBRI)*(dE) + COST_HIBRI*(dH);
30:         if aceita(dfc) then
31:             atualiza_posicao();{atualiza posição do átomo}
32:         else
33:             desfaz_movimento();{volta átomo pos. anterior}
34:         end if
35:         temperatura = temperatura * λ {diminui a temp.}
36:     end for
37: end for
```

Figura 4. Alg. paralelo para geração de CRN

Na Figura 4 está representado um pseudocódigo do algoritmo paralelo. Nele vê-se que a leitura dos parâmetros (linha 2) e o posicionamento dos átomos (linha 3) são feitos por um processo principal. Após isso, define-se a temperatura inicial do sistema (linha 5) e calcula-se o número de iterações executada por cada processo (linha 6).

O passo seguinte consiste em um laço que controla as iterações do sistema (linha 7 a 37). A cada iteração do laço verifica-se se a iteração atual é uma iteração de mudança do tamanho do universo (linha 8). Caso seja, os dados são recolhidos para o processo principal (linha 9), esse executa o procedimento de mudança (linha 10) e envia os dados de volta aos demais processos (linha 11). Depois de recebidos os dados, os processos colocam cada átomo em uma célula (linha 12) e fazem a hibridização dos mesmos (linha 13). Caso não seja uma iteração de mudança do tamanho do universo, os processos trocam informações apenas

Proc	Seq	2		4		8		27	
Átomos	**Tempo**	**Tempo**	**Efic**	**Tempo**	**Efic**	**Tempo**	**Efic**	**Tempo**	**Efic**
8	37	19	97,37%	14	66,07%	10	46,25%	-	-
32	94	48	97,92%	31	75,81%	22	53,41%	-	-
128	145	78	92,95%	45	80,59%	36	50,35%	-	-
512	195	95	102,63%	54	90,28%	40	60,94%	15	48,15%
2048	530	211	125,59%	115	115,22%	81	81,79%	29	67,69%

Tabela 1. Tempo de execução e eficiência das simulações (tempos segundos)

com os processos vizinhos (linha 15). Em seguida, realiza-se a hibridização dos novos átomos recebidos (linha 16), desloca-se a área de atuação (linha 17) e calcula-se a energia total do sistema (linha 19). A partir daí, cada processo faz um conjunto de passos independentes de SA (linhas 20 a 36), selecionando os átomos de suas respectivas regiões. Após isso, volta-se ao início do laço onde novos passos são executados.

Para validar e verificar o funcionamento do sistema paralelo, foram feitas algumas medições de tempo de execução com 2 ($2\times1\times1$) , 4 ($2\times2\times1$) e 8 ($2\times2\times2$) processos usando um universo de 8, 32, 128, 512 e 2048 átomos. Além disso, foram realizadas medições com 27 ($3\times3\times3$) processos para 512 e 2048 átomos. Foi definido 30 como número de passos independentes e 1000 como intervalo entre mudanças do tamanho do universo. As medições foram realizadas no *cluster* Labtec do Instituto de Informática da Universidade Federal do Rio Grande do Sul (II UFRGS). Esse *cluster* é composto por nós bi-processados Pentium III 1.1 GHz com 1 Gb de memória RAM e rede Gigabit Ethernet. Cada processador ficou responsável por um processo. Os resultados das execuções podem ser vistos na Tabela 1, onde, além dos tempos paralelos, são mostrados os seqüenciais. Cabe salientar que os testes de execuções foram feitos com números de passos limitados (8×10^5), sendo esse número muito inferior ao necessário para alcançar a convergência do sistema (principalmente com um grande número de átomos). Porém, através desses resultados, pode-se observar o comportamento do mesmo.

Com base nesses dados, pôde-se observar que na medida em que o número de átomos aumenta, a eficiência do programa também aumenta, chegando a alguns casos em que a eficiência foi superior a 100%. Esse acontecimento deve-se principalmente ao fato de que na versão paralela, a rotina de mudança de tamanho do universo acontece menos vezes (diretamente proporcional ao número de processos). Também é observado que quanto maior é o número de processos envolvidos, menor é a eficiência. Isso se deve ao fato de que, em operações globais, tal como a mudança do tamanho do universo, esse maior número de processos gera certo gargalo. Além disso, quanto mais eixos são divididos, maior é a quantidade de mensagens trocadas e conseqüentemente maior a demora.

4. Conclusões

Nesse artigo foi apresentada a paralelização do método *Simulated Annealing* aplicada ao algoritmo de geração de redes aleatórias contínuas.

Os tempos coletados nos testes foram satisfatórios mostrando que através do uso de técnicas propostas pela literatura junto com divisão de domínio, é possível reduzir significativamente o tempo de processamento. Também se deve salientar que, com a versão paralela do programa, a possibilidade de alcançar os objetivos almejados pelos pesquisadores do Instituto de Física ficou maximizada.

Os trabalhos futuros estarão concentrados em fazer uma análise detalhada sobre o comportamento de cada etapa do processo de *Simulated Annealing* (diferentes temperaturas). Essa análise tem o intuito de descobrir quais os melhores valores para o número de passos independentes e a distância entre mudanças de tamanho de universo nas diversas temperaturas da simulação. Também serão verificadas formas de executar tarefas que na versão atual são centralizadas, como a mudança de tamanho do universo, de forma distribuída. Isso fará com que o desempenho do sistema seja aumentado. Além disso, serão feitas implementações diferenciadas, tentando explorar de forma mais eficiente características das arquiteturas (multi-processadores, multi-cores,...), a fim de conseguir um melhor desempenho.

Referências

[1] G. T. Barkema and N. Mousseau. High-quality continuous random networks. *Phys. Rev. B*, 62(8):4985–4990, Aug 2000.

[2] F. H. Jornada and C. A. Perottoni. Geração de redes contínuas aleatórias de carbono por simulated annealing. Technical report, Instituto de Física da Universidade Federal do Rio Grande do Sul, 2007.

[3] S. W. Mahfoud and D. E. Goldberg. Parallel recombinative simulated annealing: a genetic algorithm. *Parallel Computing*, 21(1):1–28, 1995.

[4] C. A. Perottoni and J. A. H. Jornada. The carbon analogues of type-i silicon clatrates. *J. Phys.: Condens. Matter*, 13:5981–5998, 2001.

[5] F. Wooten, K. Winer, and D. Weaire. Computer generation of structural models of amorphous si e ge. *Physical Review Letters*, 54(13):1392–1395, 1985.

[6] W. H. Zachariasen. The atomic arrangement in glass. *J. American Ceramic Society*, 54(9):3841–3851, 1932.