Paralelização do Algoritmo de Geração de Redes Aleatórias Contínuas

Gustavo Romano, Nicolas Maillard, Ricardo Vargas Dorneles

Universidade Federal do Rio Grande do Sul - UFRGS

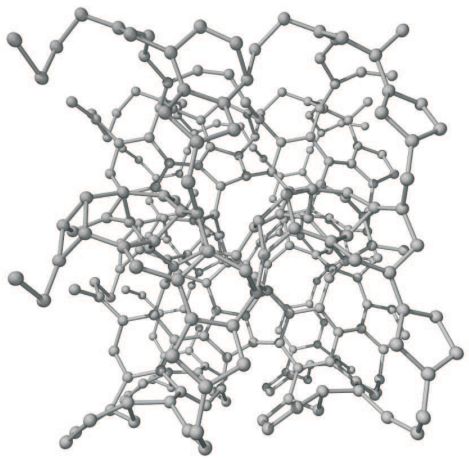
10 de Agosto de 2007

Roteiro

1. Introdução
   - Introdução
2. Algoritmo Proposto
   - Algoritmo Proposto
3. Paralelização
   - Paralelização
4. Testes e Resultados
   - Testes
   - Resultados
5. Conclusões
   - Conclusão
   - Trabalhos Futuros

Introdução

- Nanotecnologia tem despertado interesse na ciência dos materiais
- Carbono tem se destacado na possibilidade de obter novos compostos
  - Diamante, grafite e nanotubos
  - Propriedades físicas distintas pela forma com que os átomos arranjam-se
- Estruturas amorfas (sp^3, sp^2 e sp)
- Modeladas através de redes aleatórias contínuas (CRN)



Introdução

- Instituto de Física da Universidade Federal do Rio Grande do Sul (UFRGS)
- CRN de carbono com hibridização mista sp^3/sp^2
- Propor novos materiais (reais ou hipotéticos) cuja dureza e módulo volumétrico sejam significativamente superiores aos do diamante
- Algoritmos tradicionais de CRN não são eficientes redes mistas

Algoritmo Seqüêncial

```
le_parametros();{lê parâmetros de entrada do sistema}
posiciona_atomos();{posiciona os átomos aleatoriamente no sistema}
do_cells();{coloca cada átomo em sua célula do sistema}
hibri_all();{hibridiza todos os átomos}
tote();{calcula a energia total do sistema}
temperatura = temp_ini;{inicializa a temperatura}
for iter = 0 to MAX_ITER − 1 do
    if iter%RESIZE_UNI_EACH == 0 then
        muda_tamanho_universo();{muda tam. do universo}
    end if
    at = atomos[rand(NATM)];{pega um átomo aleat.}
    E_local_old = local_energy(at);{calc. energia local}
    Hibri_old = hibri_cost(at);{calc. custo de hibrid.}
    desloca_atomo(at);{desloca átomo direções x, y e z}
    E_local_new = local_energy(at);{calc. energia local}
    Hibri_new = hibri_cost(at);{calc. custo de hibrid.}
    dE = E_local_new - E_local_old;{calc. var. de energia}
    dH = Hibri_new - Hibri_old;
    dfc = (1.0 - COST_HIBRI)*(dE) + COST_HIBRI*(dH); {calc. var. da função custo}
    if aceita(dfc, temperatura) then
        atualiza_posicao();{atualiza a posição do átomo}
    else
        desfaz_movimento();{volta posição anterior}
    end if
    temperatura = temperatura * λ {diminui a temperatura}
end for
```

Figura: Algoritmo seqüencial para geração de CRN

Algoritmo Seqüêncial

```
le_parametros();{lê parâmetros de entrada do sistema}
posiciona_atomos();{posiciona os átomos aleatoriamente no sistema}
do_cells();{coloca cada átomo em sua célula do sistema}
hibri_all();{hibridiza todos os átomos}
tote();{calcula a energia total do sistema}
temperatura = temp_ini;{inicializa a temperatura}
for iter = 0 to MAX_ITER − 1 do
    if iter%RESIZE_UNI_EACH == 0 then
        muda_tamanho_universo();{muda tam. do universo}
    end if
    at = atomos[rand(NATM)];{pega um átomo aleat.}
    E_local_old = local_energy(at);{calc. energia local}
    Hibri_old = hibri_cost(at);{calc. custo de hibrid.}
    desloca_atomo(at);{desloca átomo direções x, y e z}
    E_local_new = local_energy(at);{calc. energia local}
    Hibri_new = hibri_cost(at);{calc. custo de hibrid.}
    dE = E_local_new - E_local_old;{calc. var. de energia}
    dH = Hibri_new - Hibri_old;
    dfc = (1.0 - COST_HIBRI)*(dE) + COST_HIBRI*(dH); {calc. var. da função custo}
    if aceita(dfc, temperatura) then
        atualiza_posicao();{atualiza a posição do átomo}
    else
        desfaz_movimento();{volta posição anterior}
    end if
    temperatura = temperatura * λ {diminui a temperatura}
end for
```

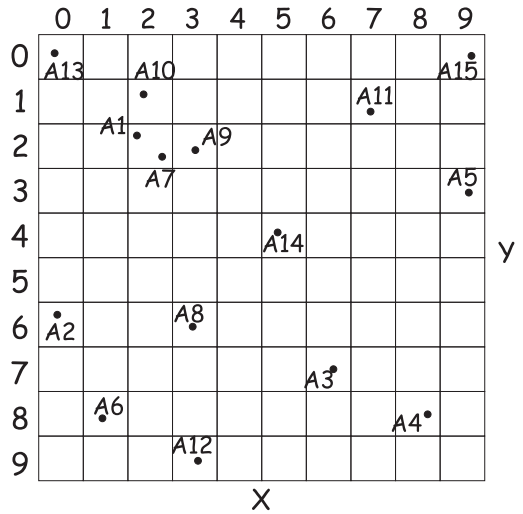


Figura: Algoritmo seqüencial para geração de CRN

Algoritmo Seqüêncial

```
le_parametros();{lê parâmetros de entrada do sistema}
posiciona_atomos();{posiciona os átomos aleatoriamente no sistema}
do_cells();{coloca cada átomo em sua célula do sistema}
hibri_all();{hibridiza todos os átomos}
tote();{calcula a energia total do sistema}
temperatura = temp_ini;{inicializa a temperatura}
for iter = 0 to MAX_ITER − 1 do
    if iter%RESIZE_UNI_EACH == 0 then
        muda_tamanho_universo();{muda tam. do universo}
    end if
    at = atomos[rand(NATM)];{pega um átomo aleat.}
    E_local_old = local_energy(at);{calc. energia local}
    Hibri_old = hibri_cost(at);{calc. custo de hibrid.}
    desloca_atomo(at);{desloca átomo direções x, y e z}
    E_local_new = local_energy(at);{calc. energia local}
    Hibri_new = hibri_cost(at);{calc. custo de hibrid.}
    dE = E_local_new - E_local_old;{calc. var. de energia}
    dH = Hibri_new - Hibri_old;
    dfc = (1.0 - COST_HIBRI)*(dE) + COST_HIBRI*(dH); {calc. var. da função custo}
    if aceita(dfc, temperatura) then
        atualiza_posicao();{atualiza a posição do átomo}
    else
        desfaz_movimento();{volta posição anterior}
    end if
    temperatura = temperatura * λ {diminui a temperatura}
end for
```

Figura: Algoritmo seqüencial para geração de CRN

Algoritmo Seqüêncial

- Simulações feitas com o programa geram boas redes
- Muito demoradas
  - Casa de semanas para redes de 128 átomos
- Para encontrar um material que atenda as propriedades desejadas
  - Novas simulações com um universo superior a 1024 átomos (preferencialmente superior a 2048)
  - Meses de simulação

Paralelização

- Primeira abordagem = Divisão de domínio;

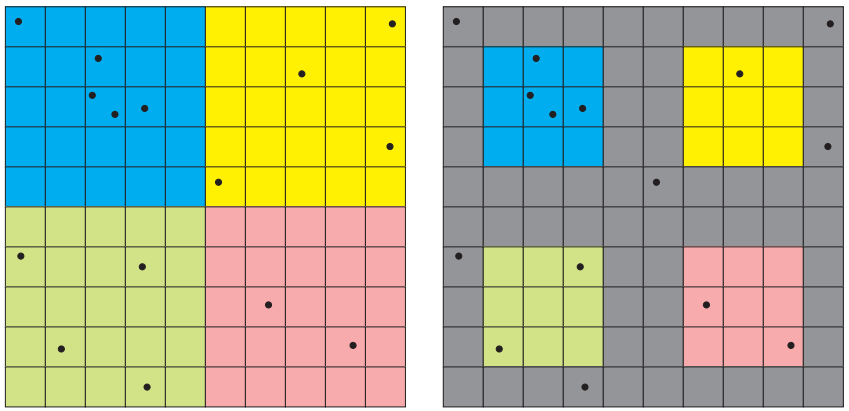


Figura: Distribuição dos dados entre os processos. (A) sem regiões neutras, (B) com regiões neutras

Paralelização

- Deslocamento para evitar áreas não processadas e diminuir comunicações

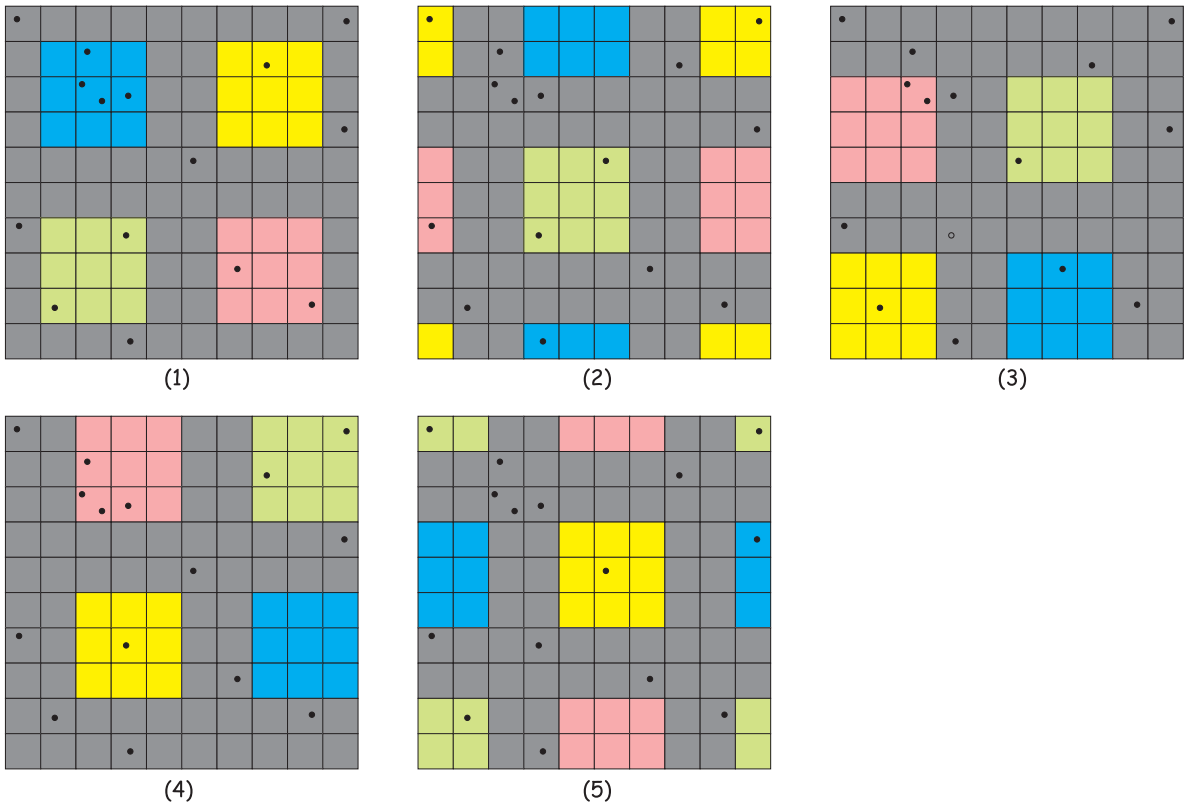


Figura: Funcionamento do deslocamento da área de responsabilidade de cada processo

Paralelização

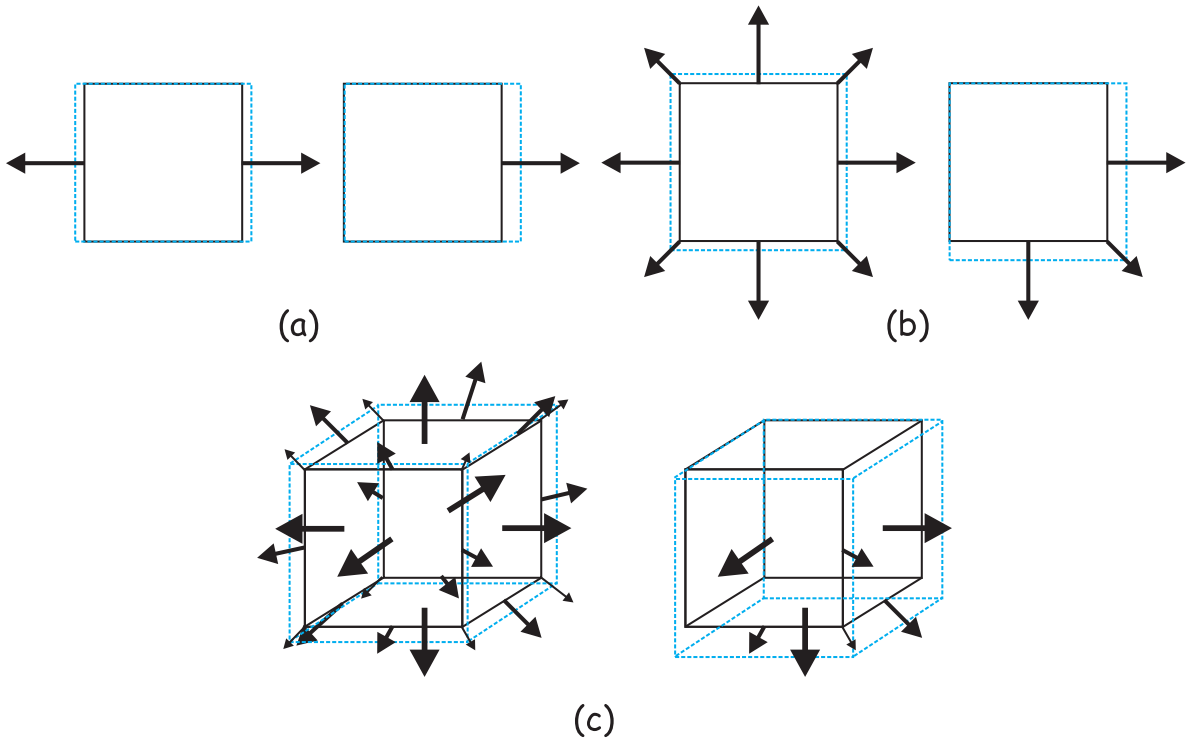


Figura: Mensagens trocadas no sistema convencional e naquele com deslocamento

Algoritmo Paralelizado

```
if rank==0 then
    le_parametros();{lê parâmetros de entrada do sistema}
    posiciona_atomos();{posiciona os átomos}
end if
temperatura = temp_ini;{inicializa a temperatura}
Max_iters = (Max_Steps/size/Pas_Independ)
for iter = 0 to Max_iters do
    if iter%RESIZE_UNI_EACH == 0 then
        recolhe_dados() {procs. enviam dados proc. 0}
        muda_tamanho_universo();{muda tam. do universo}
        distribui_dados();{Distrib. pos. átms p/ procs.}
        do_cells();{coloca cada átomo em sua célula}
        hibri_all();{refaz a hibridização dos átomos}
    else
        troca_dados();{troca pos. vizinhos.}
        hibri_novos();{hibri átomos receb}
        desloca_area();{desloc. área atu.}
    end if
    tote();{calcula a energia total do sistema}
    for passo = 0 to Passos_Independ do
        PASSOS INDEPENDENTES{executa passso independentes (igual seqüêncial, exceto pelo átomo
        selecionado ser da área de atuação de do processador)}
    end for
end for
```

Figura: Algoritmo paralelo para geração de CRN

Testes

- Medições de tempo de execução com 2 $(2 \times 1 \times 1)$, 4 $(2 \times 2 \times 1)$ e 8 $(2 \times 2 \times 2)$ processos usando um universo de 8, 32, 128, 512 e 2048 átomos.
- Medições com 27 $(3 \times 3 \times 3)$ processos para 512 e 2048 átomos
- Realizadas no *cluster* Labtec do Instituto de Informática da Universidade Federal do Rio Grande do Sul (II UFRGS)
  - Nós bi-processados Pentium III 1.1 GHz
  - 1 Gb de memória RAM
  - Rede Gigabit Ethernet
- Números de passos limitados (8×10^5)

Resultados

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: Tempo de execução e eficiência das simulações (tempos segundos)

Resultados

- Na medida em que o número de átomos aumentava, a eficiência do programa também aumentava
- Casos em que a eficiência foi superior a 100%.
  - Rotina de mudança de tamanho do universo acontece menos vezes (diretamente proporcional ao número de processos)
- Quanto maior o número de processos envolvidos menor a eficiência
  - Operações globais (mudança do tamanho do universo) - maior número de processos gera certo gargalo
  - Quanto mais eixos são divididos, maior é a quantidade de mensagens trocadas

Conclusões

- Foi apresentada a paralelização do método *Simulated Annealing* aplicada ao algoritmo de geração de redes aleatórias contínuas
- Os tempos coletados nos testes foram satisfatórios
- Com a versão paralela do programa, a possibilidade de alcançar os objetivos almejados pelos pesquisadores do Instituto de Física ficou maximizada

Trabalhos Futuros

- Fazer uma análise detalhada sobre o comportamento de cada etapa do processo de *Simulated Annealing* (diferentes temperaturas)
- Verificar formas de executar tarefas que na versão atual são centralizadas, como a mudança de tamanho do universo, de forma distribuída
- Implementações diferenciadas, tentando explorar de forma mais eficiente características das arquiteturas (multi-processadores, multi-cores,...)