MESTRADO EM ENGENHARIA INFORMÁTICA

UCE: COMPUTAÇÃO PARALELA E DISTRIBUÍDA

ALGORITMOS E MÉTODOS NUMÉRICOS

Ano letivo de 2012/2013

Projecto 1: Utilização da técnica de "simulated annealing"

Descrição do problema

Trata-se de distribuir n alunos (assume-se que n é par) por $n/2$ quartos de uma residência universitária por forma a evitar, tanto quanto possível, eventuais incompatibilidades entre parceiros de quarto. É construída uma matriz

$$D = (d_{ij})$$

tal que que

$$0 \leq d_{ij} \leq 10$$

para cada par (i, j), $i \neq j$. O valor de d_{ij} "mede" um potencial conflito entre os estudantes i e j: no caso ideal é $d_{ij} = d_{ji} = 0$ (assume-se que a matriz D é simétrica), no caso de extrema incompatibilidade entre os estudantes i e j, é $d_{ij} = d_{ji} = 10$. O objectivo é encontrar a solução que minimiza a soma (custo)

$$\sum_{k=1}^{n/2} D\left(i_k, j_k\right).$$

Note-se que o número de possíveis distribuições distintas é dado por

$$C(n) = (n-1) \times (n-3) \times \cdots \times 3 = \sqrt{\prod_{k=1}^{n/2} \frac{2k-1}{2k}}.\sqrt{n!}.$$

Na expressão anterior, o fator $\sqrt{\prod_{k=1}^{n/2} \frac{2k-1}{2k}}$ decresce muito lentamente com n, vale cerca de 0.5 para n=10, 0.28 para n=100 e 0.16 para n=1000. Portanto, $C(n)$ é um número da ordem de grandeza de $\sqrt{n!}$, isto é, cresce muito rapidamente com n, tornando impraticável a análise exaustiva de todas as possibilidades.

A alternativa é usar um método heurístico que não garante a solução óptima. Parte-se de uma distribuição inicial aleatória; em cada passo, são escolhidos, ao acaso, dois estudantes de quartos diferentes; se a troca entre eles conduzir a uma solução de menor custo, a troca é sempre realizada; uma troca que conduza a uma solução de maior custo também pode ser aceite, dependendo da temperatura do sistema. O sistema vai "arrefecendo" à medida que o processo evolui e a probabilidade de uma solução com custo mais elevado ser aceite vai diminuindo. Nisto consiste a técnica conhecida por "simulated annealing" (veja [1], pag.259 para mais detalhes).

O projecto

1. Desenvolva em C, $C++$ ou $Fortran$ uma implementação sequencial do algoritmo, que designaremos por **Alg1**, que nunca aceita uma solução que tenha custo maior do que a anterior (e portanto não usa "simulated annealing"). Dado n (par), o código gera aleatoriamente uma matriz D de inteiros entre 0 e 10 (incompatibilidades) e uma distribuição inicial; em seguida, inicia o processo de melhoria da solução. A busca termina após um número **max** de tentativas **consecutivas** sem alteração da solução.

2. Modifique o código anterior para produzir uma implementação, que designaremos por **Alg2**, que implementa a técnica "simulated annealing", isto é, aceita ou não uma solução com custo mais elevado dependendo da diferença de custos e da temperatura do sistema. Use uma temperatura inicial igual a 1, e faça, em cada passo, $T = 0.999T$.

3. Para testar **Alg1** e **Alg2**, introduza na matriz incompatibilidades D uma solução de custo zero (isto é, atribua o valor de zero a $n/2$ elementos da matriz D, não repetindo linhas nem colunas). Repita a execução do código para diferentes valores de n entre 20 até 200 e com diferentes números pseudo-aleatórios, procurando encontrar a solução de custo mínimo. Num gráfico represente os custos das soluções produzidas, para valor de n, pelos dois códigos. Use diferentes valores de **max** (maiores para valores de n maiores).

4. Execute os mesmos códigos, variando o valor da temperatura inicial, por exemplo usando o valor $T_0 = 10$.

5. Desenvolva uma implementação paralela em C e MPI de **Alg1**. O algoritmo é "embaraçosamente paralelo" já que cada processo executará o mesmo código mas usando diferentes números pseudo-aleatórios e produzindo, portanto, diferentes distribuições. A solução paralela será a solução de menor custo de entre o conjunto das soluções obtidas pelos diferentes processos.

6. Teste o seu código paralelo no cluster SEARCH, usando um número de processos tão grande quanto lhe for possível.

7. Escreva um relatório do trabalho desenvolvido (não mais do que 10 páginas). Em particular, compare a qualidade das soluções obtidas com as implementações sequenciais **Alg1** e **Alg2** e também a qualidade das soluções obtidas com a implementações sequencial e paralela de **Alg2**.

[1] Michael J. Quinn, Parallel Programming in C with MPI and OpenMP, McGraw-Hill 2003.

[2] Milena Lazarova, Parallel simulated annealing for solving the room assignment problem on shared and distributed memory platforms, Proceedings of CompSysTech'08 (9th Int. Conf. on Computer Systems and Technologies), ACM, 2008, pp.II.13-1 - II.13.6.