2º Trabalho - Implementação do Problema do Caixeiro Viajante

Considere **G = (VG, aG)**, um grafo orientado, sendo **VG** o conjunto de vértices e **aG** o conjunto de arestas com custos $\mathbf{c_{i,j}}$ positivos associados às arestas **(i,j)** [1]. Quando não existir uma aresta **(i,j)**, $\mathbf{c_{i,j}}$ tem um valor infinito. Supondo **VG = n > 1** (sendo **N** o número de vértices), uma viagem **G** é um circuito (orientado) que contém cada vértice em VG uma e somente uma vez. A soma dos custos das arestas na viagem é chamada custo da viagem. O **Problema do Caixeiro Viajante (PCV)** consiste em achar uma viagem mínima, isto é, dentre todas as viagens possíveis em **G**, uma viagem de custo mínimo. Pode haver mais do que uma de tais viagens [1].

O algoritmo trivial para resolver o PCV consiste em enumerar todas as **N!** permutações dos **N** vértices em VG, calcular os custos de cada viagem correspondente a cada uma das permutações e escolher uma viagem de custo mínimo [1]. O objetivo deste trabalho é **implementar este algoritmo** trivial em C/MPI.

Suponha, apenas a título de exemplo e sem perda de generalização, que uma viagem começa e termina no vértice **0**, após visitar cada uma dos vértices **1, 2, 3, ..., N-1** só uma vez [1]. Assim, qualquer viagem é formada por uma aresta **(0,k), 1** $\leq$ **k** $\leq$ **N-1**, e um caminho **M** de **k até 0**. O caminho **M** visita cada um dos vértices em **VG - {0, k}** uma e só uma vez. Se a viagem considerada é de custo mínimo, o caminho **M** será também de custo mínimo [1]. Considere **f(i, C)** o custo mínimo do vértice **i até 0** e que visita todos os vértices de **C**. Assim, tem-se:

$$\mathbf{f(0, VG-\{0\}) = \min\{c_{0,k} + f(k, VG-\{0,k\})\}, 1 \leq k \leq N-1}$$

Generalizando tem-se:

$$\mathbf{f(i, C) = \min_{j \in C} \{c_{i,j} + f(j, C-\{j\})\}}$$

isto é, o caminho mínimo do vértice **i** até **0**, que visita todos os vértices em **C**, é constituído por **(i,j)** e um caminho de **j** até **0** que visita todos os vértices em **C-{j}**, para uma escolha adequada de **j** em **C** (vide Fig.1) [1].

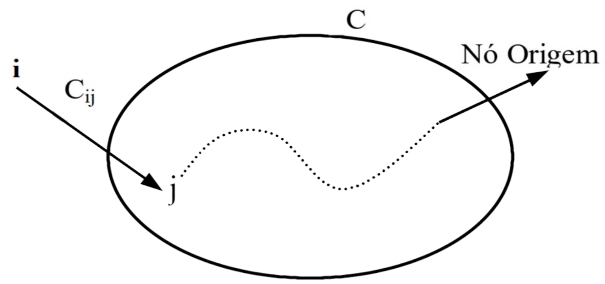

Figura 1 – Caminho de custo mínimo do vértice **i** até o vértice **0**.

Considere o seguinte exemplo [1]:

	0	**1**	2	3
0	0	**1**	2	2
1	5	0	**1**	4
2	3	8	0	**1**
3	**1**	7	6	0

Matriz de Distâncias

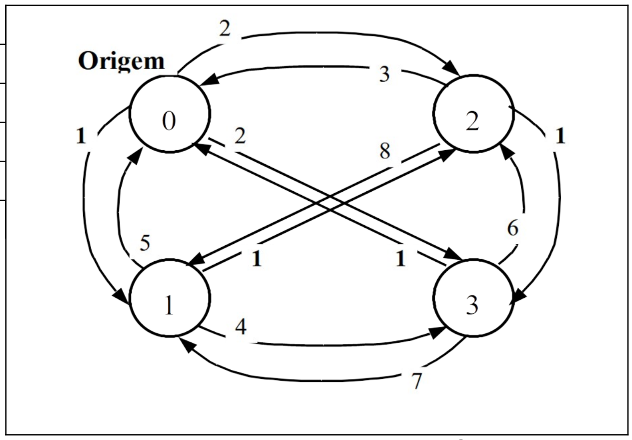

Figura 2 – Grafo com 4 vértices

Execução sequencial considerando o algoritmo descrito e o exemplo dado:

f(0, {1,2,3}) = min {C_{01} + f(1, {2, 3}), C_{02} + f(2, {1, 3}), C_{03} + f(3, {1, 2}) };

f(0, {1,2,3}) = min {(1 + 3), (2 + 13), (2 + 11) }

f(0, {1,2,3}) = 4

f(1, {2,3}) = min {C_{12} + f(2, {3}), C_{13} + f(3, {2}) } = min{ (1 + 2), (4 + 9) } = 3;

f(2, {1,3}) = min {C_{21} + f(1, {3}), C_{23} + f(3, {1}) } = min{ (8 + 5), (1 + 12) } = 13;

f(3, {1,2}) = min {C_{31} + f(1, {2}), C_{32} + f(2, {1}) } = min{ (7 + 4), (6 + 13) } = 11;

f(1, {2}) = C_{12} + C_{20} = 1 + 3 = 4;

f(1, {3}) = C_{13} + C_{30} = 4 + 1 = 5;

f(2, {1}) = C_{21} + C_{10} = 8 + 5 = 13;

f(2, {3}) = C_{23} + C_{30} = 1 + 1 = 2;

f(3, {1}) = C_{31} + C_{10} = 7 + 5 = 12;

f(3, {2}) = C_{32} + C_{20} = 6 + 3 = 9;

Implemente em C/MPI um algoritmo paralelo que resolva **o PCV descrito aqui** para N cidades, sendo **N>1**. Você pode, opcionalmente, acrescentar o uso de OpenMP e de instruções SIMD na sua solução, sendo que, neste caso, você usaria os paradigmas SIMD, OpenMP e MPI. Os usos de instruções SIMD e de OpenMP não são obrigatórios, mas o uso do MPI é obrigatório.

Determine no seu algoritmo o custo do caminho mínimo e o caminho mínimo. A solução deve apresentar um código com qualidade, com uso adequado/correto das rotinas MPI, com objetivo principal de obter ganho de desempenho em relação ao tempo de resposta para a aplicação, dada a sua execução em uma plataforma MIMD com memória distribuída (*cluster* com múltiplos nós, todos multicore com extensões SIMD).

A entrada deve ser feita da seguinte forma:

pcv N (sendo ***pcv*** o binário e ***N*** o número de cidades)

A saída deve ocorrer no console e conter **apenas**:

relação do caminho mínimo a ser percorrido e

custo do caminho mínimo.

Os custos $\mathbf{c_{i,j}}$ devem ser gerados pseudo-aleatoriamente pelo seu código.

Além do programa em si, também apresente o seu PCAM, conforme conteúdo visto em sala de aula.

Submeta no e-disciplinas um arquivo compactado (obrigatoriamente no padrão zip) contendo: o código fonte sequencial (deve se chamar pvc-seq.c), código fonte paralelo (deve se chamar pvc-par.c), o projeto do algoritmo paralelo no PCAM (deve se chamar pcam-pvc.pdf) e os resultados das execuções (deve-se chamar resultados-pvc-mpi.pdf) contendo a tabela dos resultados, gráficos e a explicação dos resultados obtidos. Inclua também um makefile que realize a compilação e a execução dos códigos sequencial e paralelo. Coloque em todos os arquivos os nomes dos integrantes do grupo que participaram do desenvolvimento do trabalho. Questões omissas e/ou ambíguas serão fixadas pelo professor. Para saná-las, por favor entre em contato com o professor para a orientação adequada.

Referência Bibliográfica:

[1] Terada, Routo "Desenvolvimento de Algoritmos e Estruturas de Dados". São Paulo. McGraw-Hill, Makron, 1991.