2º Trabalho - Implementação do Problema do Caixeiro Viajante em MPI e OMP

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].

Suponha, 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),** $\mathbf{1 \leq k \leq N-1}$, 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:

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

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].

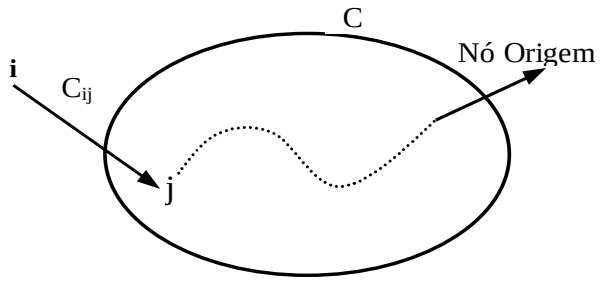

Figura 1 – Caminho de custo mínimo do vértice **i** até o vértice **0**.

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

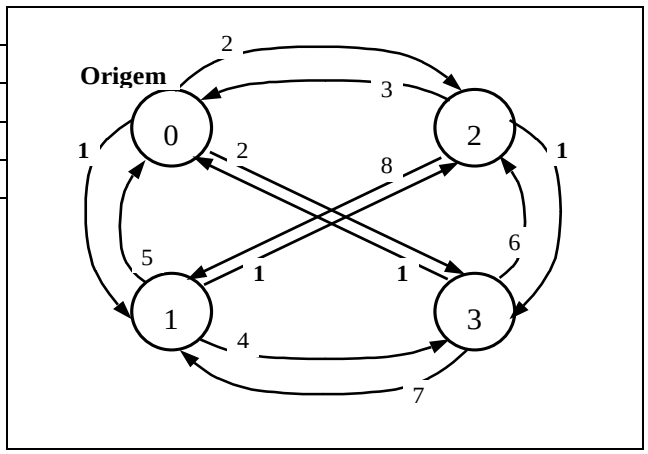

Figura 2 – Grafo com 4 vértices

Execução sequencial considerando o algoritmo descrito e o exemplo dado:

f(0, {1,2,3}) = min {C01 + f(1, {2, 3}), C02 + f(2, {1, 3}), C03 + 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 {C12 + f(2, {3}), C13 + f(3, {2}) } = min{ (1 + 2), (4 + 9) } = 3;

f(2, {1,3}) = min {C21 + f(1, {3}), C23 + f(3, {1}) } = min{ (8 + 5), (1 + 12) } = 13;

f(3, {1,2}) = min {C31 + f(1, {2}), C32 + f(2, {1}) } = min{ (7 + 4), (6 + 13) } = 11;

f(1, {2}) = C12 + C20 = 1 + 3 = 4;

f(1, {3}) = C13 + C30 = 4 + 1 = 5;

f(2, {1}) = C21 + C10 = 8 + 5 = 13;

f(2, {3}) = C23 + C30 = 1 + 1 = 2;

f(3, {1}) = C31 + C10 = 7 + 5 = 12;

f(3, {2}) = C32 + C20 = 6 + 3 = 9;

Implemente em C/MPI/OpenMP (Linux) um algoritmo concorrente que resolva **o PCV descrito aqui** para N cidades, sendo **N>1**.

Não implemente outro algoritmo para o PCV. Implemente a versão da solução descrita neste enunciado. O principal objetivo do trabalho é analisar o uso adequado dos modelos de programação MPI e OpenMP pelos grupos, não a escolha da solução base para o PCV. Esta escolha já foi feita na descrição do trabalho proposto.

O trabalho deve usar eficientemente ambos os paradigmas passagem de mensagens e memória compartilhada, considerando os melhores recursos que eles oferecem para otimizar o desempenho do problema proposto (na forma e versão propostas aqui).

Determine o custo do caminho mínimo e o caminho mínimo. A sua solução deve apresentar um código com qualidade, com uso adequado/correto das rotinas MPI e de OpenMP, com objetivo principal de obter o menor tempo de resposta final para a aplicação, dada a sua execução em um *cluster* com nós *multicore*.

A entrada deve ser feita como: ***pcv <arquivo-entrada.txt>*** (sendo ***pcv*** o binário e ***arquivo-entrada.txt*** um arquivo texto contendo o número de cidades e a matriz de adjacência com os custos dos caminhos. O ***arquivo-entrada.txt*** para o exemplo acima tem este conteúdo:

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

A saída deve ocorrer no console (*STDOUT*) e deve conter apenas a relação do caminho mínimo a ser percorrido e o custo do caminho mínimo. A saída esperada para o exemplo acima está apresentada abaixo para ilustrar. Não há espaço ao final das linhas e na segunda linha há um espaço separando cada número.

```
4
0 1 2 3 0
```

Além do programa paralelo em C/MPI/OMP, o trabalho deverá conter também uma descrição detalhada do projeto do algoritmo desenvolvido (PCAM) e o *makefile* para a sua compilação. Para a submissão junte todos esses três arquivos em um ZIP e submeta o mesmo pelo e-disciplinas.

Dúvidas sobre o trabalho devem ser solucionadas com o professor.

Referência Bibliográfica:

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