O Algoritmo de Dijkstra:

Uma implementação baseada em computação paralela

Silvio Romero de Araújo Júnior
Departamento de Engenharia Elétrica
Centro Universitário da FEI
São Bernardo do Campo, Brasil
silvioromerojr@yahoo.com.br

Reinaldo A. C. Bianchi
Departamento de Engenharia Elétrica
Centro Universitário da FEI
São Bernardo do Campo, Brasil
rbianchi@fei.edu.br

***Resumo* —** Buscar o caminho mais curto entre dois pontos possui diversas aplicações em computação. E neste âmbito o algoritmo de Dijkstra exerce um papel fundamental. Usando este algoritmo, encontra-se distância mais curta e assim pode ser aplicado em protocolos de roteamento e aplicações baseadas em mapas, por exemplo. Com a ajuda da computação paralela, essas operações podem ser realizadas visando reduzir o tempo e a eficiência do algoritmo.

Palavras chave – Dijkstra; OpenMP; Shortest Path Algorithms

I. INTRODUÇÃO

Encontrar a menor distância para todos os objetos em um gráfico é uma tarefa comum na solução de muitos problemas científicos do dia a dia. O algoritmo para encontrar o caminho mais curto, encontra sua aplicação em muitos campos, como mapas do Google, protocolo de roteamento etc. Existem dois tipos para encontrar o caminho mais curto, o caminho mais curto de fonte única e o caminho mais curto de todos os pares, usando dois algoritmos: algoritmo de Dijkstra. O algoritmo de Dijkstra requer pesos de borda não negativos, enquanto o algoritmo de Floyd exige bordas de peso negativo. O algoritmo de caminho mais curto é o algoritmo básico para o hotspot de pesquisa. Para melhorar o sistema de transporte, o melhor uso do caminho mais curto para implementar a rede de protocolo da Internet. É usado como no algoritmo paralelo. Caso contrário, na implementação serial, é muito difícil melhorar seu desempenho [2].

A. Objetivo

Utilizar o OpenMP para implementar o algoritmo de Dijkstra de forma paralela e avaliar seus benefícios.

B. Motivação

No algoritmo de implementação serial, leva-se muito tempo para encontrar a menor distância de todos os pares de vértices no gráfico. Portanto, é uma tarefa difícil encontrar a menor distância da origem ao destino. Desta forma, usar o algoritmo paralelo ou o processamento paralelo leva menos tempo que a implementação serial. O que otimiza o algoritmo resultando na otimização de recursos computacionais.

C. Metodologia

Efetuou-se uma pesquisa bibliográfica com base em livros abordando os conceitos aplicados ao relatório. Após esta etapa, aplicou-se a linguagem de programação C++ para implementação computacional e análise do algoritmo de Dijkstra usando o OpenMP.

II. REVISÃO DA LITERATURA

O algoritmo de Dijkstra encontra o menor caminho de um nó origem (s) até os outros nós de uma rede orientada para o caso em que todos os pesos das arestas não gerem ciclos negativos. O algoritmo identifica, a partir de um vértice do grafo, qual é o custo mínimo entre esse vértice e todos os outros do grafo. No início, o conjunto S contém somente esse vértice, chamado origem. A cada passo, selecionamos no conjunto de vértices sobrando, o que é o mais perto da origem. Depois atualizamos, para cada vértice sobrando, a sua distância em relação à origem. Se passando pelo novo vértice acrescentado, a distância fica menor, é essa nova distância que será memorizada. Uma ilustração em pseudo-código do algoritmo de Dijkstra é apresentada abaixo:

1. INÍCIO

2. Selecionar o nó de origem também chamado de nó inicial.

3. Definir um conjunto vazio N que será usado para manter nós nos quais um caminho mais curto foi encontrado.

4. Rotular o nó inicial e insirir em N.

5. Repitir as Etapas 5 a 7 até que o nó de destino esteja em N ou não haja mais nós rotulados em N.

6. Considerar cada nó que não está em N e esteja conectado por uma aresta do nó recém-inserido.

7. (a) Se: o nó que não está em N não tiver rótulo, então "setar" o rótulo do nó igual ao o rótulo do nó recém-inserido mais o peso da aresta.

(b) Senão: o nó que não está em N já estiver rotulado, então "setar" seu novo rótulo igual ao mínimo (rótulo vértice recém-inserido mais o peso da aresta, rótulo antigo)

7. Escolha um nó que não esteja em N e tenha o menor rótulo atribuído a ele e adicione-o a N.

9. FIM

III. Implementação Proposta

Na implementação do código proposto (enviado como anexo) no presente relatório utilizou-se os seguintes recursos de Hardware:

- MacBook Pro (Retina, 15-inch, Early 2013);
- Processador: 2,4 GHz Intel Core i7;
- Memória: 8 GB 1600 MHz DDR3.

Com relação aos softwares utilizados, citam-se:

- Sistema operacional UBUNTU 18.04;
- Compilador gcc versão 7.4.

IV. Experimentos e Resultados

Para a medição do tempo de execução dos algoritmos, usou-se a função clock da linguagem C, inclusa na biblioteca <time.h>. Para avaliação do algoritmo, alterou-se o número de vértices, foi executado o programa para grafos esparsos e densos. Abaixo, pode-se verificar as tabelas com os dados dos testes para execução do programa.

Tabela 1 – Resultados para o Algoritmo de Dijkstra

Dijkstra	Tempo de Execução (ms)	
Vértices	Grafo Esparso	Grafo Denso
5	322,02	299,92
10	320,69	303,71
25	323,16	309,04
50	325,36	311,44
100	322,19	322,58
200	339,34	409,10

V. Considerações Finais

Neste relatório, estudou-se o algoritmo de Dijkstra para resolver o problema do caminho mais curto entre os pares e o de grafos. Ele pode ser convertido em paralelo usando o OpenMP, o que resulta no número de iterações que reduzem o tempo gasto. Otimizando os recursos computacionais

References

[1] Barroso, L. C.; Barroso, M. M. A; Filho, F. F. C.; et al. Cálculo Numérico (Com Aplicações), 2ªed., São Paulo, Editora Harbra, 1987.

[2] Chapra, S. C.; Canale R. P. Métodos Numéricos para Engenharia, Quinta Edição, São Paulo, McGraw-Hill, 2008.