Trabalho prático de implementação
Busca em árvore e o problema do caixeiro viajante

Computação Concorrente (MAB-117)
Prof. Silvana Rossetto

[1]DCC/IM/UFRJ
24 de abril de 2015

1. Descrição do problema

Vários problemas podem ser resolvidos usando a busca em árvore. Um exemplo bastante conhecido é o problema do "caixeiro viajante" (*traveling salesperson problem*). Nesse problema, um caixeiro viajante recebe uma lista de N cidades que ele precisa visitar e o custo da viagem entre cada par de cidades. Seu problema é visitar cada uma das cidades uma única vez, retornar à cidade de partida e fazer isso com o menor custo possível. A rota que começa na sua cidade de origem, visita cada cidade uma vez e retorna à cidade de origem é chamada *tour*, então o problema é encontrar o *tour* com o menor custo.

Este problema é NP-completo, i.e., não existe um algoritmo para resolver o problema (todos os seus casos) de forma significantemente melhor do que a busca exaustiva. A busca exaustiva significa examinar todas as possíveis soluções do problema e escolher a melhor (de menor custo). O número de possíveis soluções cresce exponencialmente quando o número de cidades é incrementado.

Formalmente, temos um grafo $G = (N, V, W)$ consistindo de N nós (cidades), um conjunto de arestas $V = (i, j)$ conectando as cidades, e um conjunto de pesos não-negativos $W = w(i, j)$ com o custo de cada aresta (i, j) (distância da cidade i a cidade j). O grafo é **direcionado**, então a aresta (i, j) só pode ser percorrida na direção de i para j e a aresta (j, i) pode existir ou não. Da mesma forma, $w(i, j)$ não necessariamente é igual a $w(j, i)$ se as duas arestas existem. Quando uma aresta não existe, seu peso w pode ser representado por um valor infinito (muito alto).

Existem várias soluções aproximativas para o problema do caixeiro viajante. Vamos considerar uma especialmente simples que usa um algoritmo de busca em árvore. A ideia é construir uma árvore onde os nós folhas representam tours completos (rotas que visitam todas as cidades) e os demais nós representam tours parciais (rotas que visitam parte das cidades). Cada nó da árvore tem um custo associado que corresponde ao custo da rota parcial. Essa informação pode ser usada para eliminar alguns nós da árvore (quando encontramos um tour parcial com custo maior ou igual ao melhor custo já encontrado). A Figura 1 mostra um exemplo de grafo com 4 cidades. O custo para ir da cidade 1 a cidade 2, por exemplo, é 2 e para ir da cidade 2 para a cidade 1 é 18.

Se escolhermos a cidade 0 como ponto de partida, podemos construir a árvore de busca apresentada na Figura 2. Para encontrar o *tour* de menor custo, podemos fazer uma busca (caminhamento) nessa árvore, usando, por exemplo, a estratégia de **busca em profundidade**. Sempre que encontrarmos um nó cujo custo seja maior que a melhor rota já encontrada, podemos abondar a busca a partir desse nó.

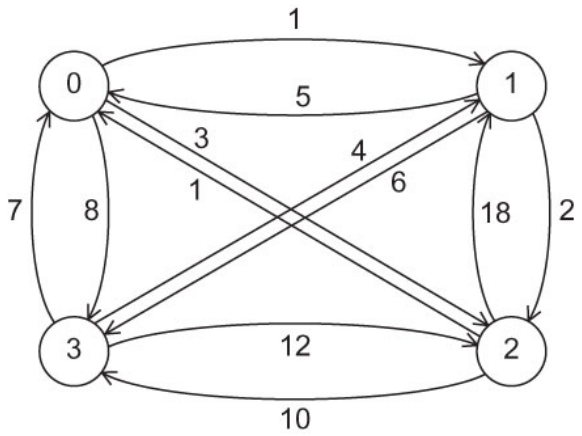

Figura 1. Exemplo de um grafo de entrada.

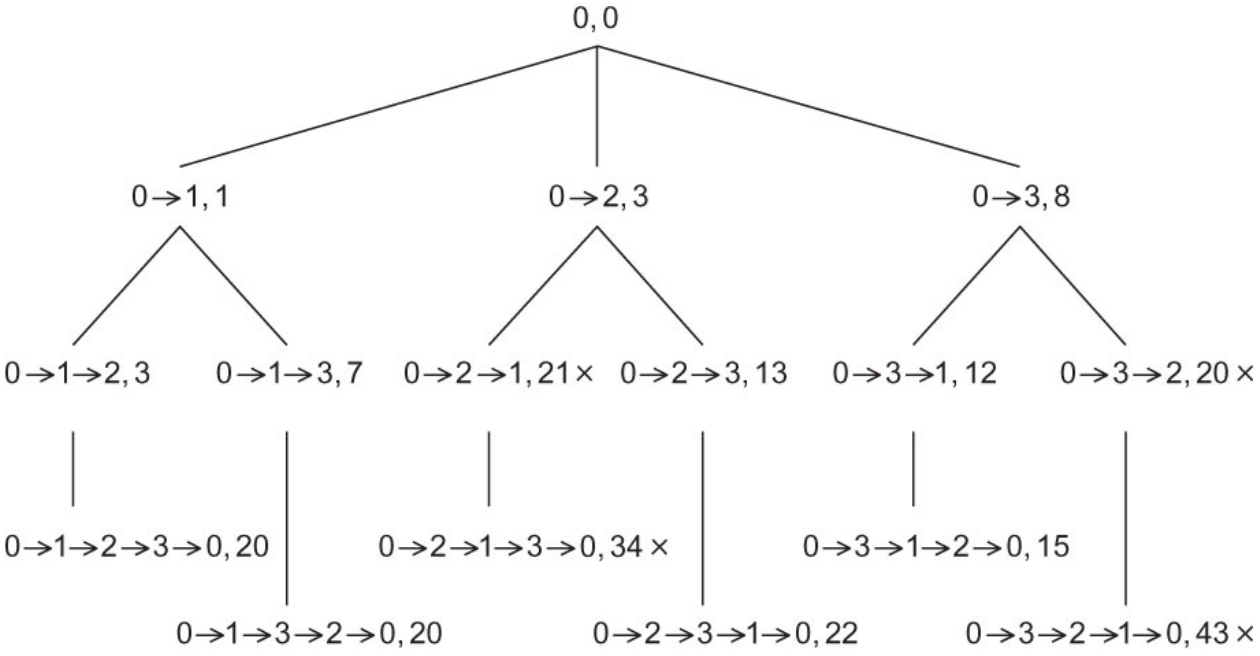

Figura 2. Exemplo da árvore de busca para o grafo com 4 cidades, partindo da cidade 0.

1.1. Tarefa

Neste trabalho, você deverá implementar uma **solução sequencial** e uma **solução paralela** para percorrer os nós de uma árvore gerada para modelar o problema do caixeiro viajante e encontar a rota de menor custo.

As duas soluções deverão ser implementadas na linguagem C. A solução paralela deverá ser desenvolvida usando **OpenMP**.

Para avaliar o desempenho da solução paralela, as seguintes métricas deverão ser usadas: **speedup** e **eficiência**. O **speedup** (S) é a razão entre o tempo total da versão sequencial (T_s) e o tempo total da versão paralela (T_p) ($S = T_s/T_p$). O melhor caso quando paralelizamos um problema é conseguir dividir a tarefa igualmente entre as threads sem adicionar carga de trabalho extra. Quando isso é possivel, temos $T_p = T_s/p$ e o speedup é linear (i.e., com p processadores/threads o programa executa p vezes mais rápido). Entretanto, com as sobrecargas do paralelismo (criação das threads, sincronizações, etc.), o speedup não chega a ser linear. Além disso, essa sobrecarga pode aumentar com o aumento do número de processadores/threads. A medida de **eficiência** (E) é usada para mostrar como o speedup muda com o aumento do número de processadores/threads (para a mesma entrada), e é calculada fazendo: S/p, onde S é o speedup e p o número de processadores/threads.

2. Entrada

A entrada para o programa deverá incluir:

- o número N de cidades
- a cidade de partida
- o custo de viagem entre cada par de cidades

Casos de teste podem ser encontrados nesse link: `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95` na seção *Asymmetric traveling salesman problem (ATSP)*.

3. Saída

A saída do programa deverá informar:

- a rota (*tour*) de menor custo (sequência de cidades)
- o custo total dessa rota
- o tempo total de processamento

4. Etapas do trabalho

A execução do trabalho deverá ser organizada nas seguintes etapas:

1. Compreender o problema, pensar e esboçar uma solução sequencial e paralela;
2. Construir um conjunto de casos de teste para avaliação;
3. Projetar uma aplicação de teste em separado;
4. Implementar as soluções projetadas e avaliar a corretude dos programas;
5. Avaliar o ganho de desempenho obtido com a solução paralela (speedup e eficiência), refinar a implementação e repetir a avaliação;
6. Redigir os relatórios e documentar os códigos.

5. Itens que deverão ser entregues

1. Relatório: documentação do projeto das soluções sequencial e paralela (esboço da lógica principal dos algoritmos da aplicação), testes realizados e resultados obtidos.
2. Código fonte e roteiro para compilação e execução.

6. Data de entrega e critérios de avaliação

O trabalho poderá ser feito em **dupla** e deverá ser entregue até o dia **22 de maio**.

Os seguintes itens serão avaliados com o respectivo peso:

- Compilação e execução correta das soluções: 4 pontos
- Relatório do trabalho: 2 pontos
- Estratégias e otimizações para melhorar o desempenho do programa: 2 pontos
- Modularidade, organização e documentação do código fonte: 1 ponto
- Avaliação geral do trabalho: 1 ponto

Os alunos integrantes da equipe poderão ser chamados para apresentar/explicar o trabalho.