Estruturas de Dados II

Prof.ª Mariella Berger

Caixeiro Viajante

1. Objetivo

O objetivo deste trabalho é implementar diferentes soluções para o problema clássico do Caixeiro Viajante.

2. Introdução

O Problema do Caixeiro Viajante (*Traveling Salesman Problem*, doravante chamado de TSP) compõe um clássico da carreira de algoritmos, teoria dos grafos, otimização combinatória e tantas outras áreas de estudos computacionais e matemáticos. Além disso, é um problema fascinante e pertence ao seleto grupo de problemas NP-Completos (o que o torna ainda mais fascinante!).

A forma assimétrica do TSP (*Asymmetric Traveling Salesman Problem*, doravante chamado de ATSP), é uma variante menos explorada do problema original e será também objeto de estudo deste trabalho.

3. Definições

Um caixeiro viajante deve visitar várias cidades partindo de uma cidade inicial qualquer, passando por todas as cidades exatamente uma vez e voltando á cidade inicial do percurso. Sabendo os custos entre todas as cidades, o caixeiro deve fazer esse caminho de forma que o custo total percorrido seja o menor possível. Chamamos uma rota, ou caminho, de tour. Chamamos o caminho de menor custo possível de tour ótimo.

No ATSP os custos entre duas cidades são não simétricos. Este fato torna sua matriz de custos uma matriz assimétrica. O ATSP é fácil de ser resolvido: geram-se todas as permutações possíveis e calcula-se o custo de cada uma delas. A permutação com o menor custo é o caminho procurado. Esta seria uma ótima solução, se não fosse por um detalhe: dadas N cidades, são geradas $N!$ permutações, o que torna esta solução intratável do ponto de vista computacional. Basta observar-se que para se obter uma

solução de rota com 21 cidades, seriam explorados 21! = 51.090.942.171.709.440.000 tours diferentes.

Computacionalmente sabe-se que o ATSP pertence à classe de problemas com complexidade O(N!) e à classe de problemas NP-Completos. Devido a essa alta complexidade, existem diversas heurísticas que encontram soluções para o problema num tempo computacional satisfatório (polinomial), mas nenhuma delas garante que a solução encontrada é a ótima.

Neste trabalho você deverá implementar soluções para os problemas ATSP e TSP simétrico que utiliza como distância entre cidades a distância euclidiana entre dois pontos. Mais detalhes serão discutidos a seguir.

4. Implementação I: Solução Ótima para o ATSP

Inicialmente o seu trabalho é implementar um algoritmo para o ATSP que gera todas as permutações possíveis das cidades e retorna a permutação de menor custo. Sua implementação deverá ser na linguagem **C.** Caso a solução não seja viável, não retorne nenhuma resposta (retorne apenas *).

5. Implemetação II – Heurística Vizinho mais Próximo para o ATSP

O seu trabalho agora é implementar para o ATSP a heurística Vizinho Mais Próximo (*Nearest Neighbour*, doravante chamado de NN) para construir um tour. No algoritmo NN, o tour se inicia com uma cidade qualquer (neste trabalho, a cidade escolhida como inicial é a cidades de índice 0). Enquanto não se insere todas as cidades, o tour ainda é um caminho e não um ciclo. Só se tornará um ciclo quando, ao final, liga-se a última cidade à primeira. Sua implementação deverá ser na linguagem **C.**

6. Implemetação III – Heurística de Melhoramento 2-opt ou 3-opt para o ATSP

Há casos em que o NN pode gerar tours com custos finais absurdos devido à falta de opções de escolha de arcos com custos viáveis. Portanto, o seu trabalho agora é implementar uma heurística de melhoramento para o tour gerado pela heurística NN.

No algoritmo 2-opt, elimina-se 2 arestas não adjacentes, reconecta-as usando duas outras arestas (formando um tour) e verifica-se se houve melhora. Este processo é repetido para todos os pares de arestas. A melhor troca (o novo tour com menor custo) é então realizada.

Você poderá escolher entre implementar a heurístícia de melhoramento 2-opt ou 3-opt para o ATSP. A heurística 3-opt segue o mesmo princípio da heurística 2-opt, diferenciando-se apenas no fato de que são selecionadas 3 arestas no processo. Sua implementação deverá ser na linguagem **C.**

7. Implemetação IV – Heurística Envoltório Convexo para o TSP simétrico

O seu trabalho agora é implementar a heurística que tem por base o algoritmo Envoltório Convexo para resolução do problema do TSP simétrico que utiliza como distância entre cidades a distância euclidiana. Nesta heurística, inicialmente, cria-se um Envoltório Convexo que engloba todas as cidades (algumas cidades ficam internas ao envoltório) e utiliza-se este envoltório como tour inicial. Para cada cidade que não pertença ao tour, inserí-la de forma que sua inserção acarrete o menor custo possível. Sua implementação deverá ser na linguagem **C.**

8. Requisitos

Seu programa vai receber como parâmetro o número n de cidades que o caixeiro viajante deverá visitar.

Para os algoritmos que resolvem o problema ATSP (exato, nn e opt), seu programa lerá do stdin a matriz M(n, n) de distâncias entre os pares de cidades. Exemplo de entrada para n = 3 cidades:

999999 331 450

162 999999 970

856 424 999999

A primeira cidade será identicada pelo índice 0, a segunda pelo índice 1 e assim sucessivamente. A matriz deve ser lida de forma que ao se acessar o valor de M[1][2], o resultado obtido seja 970, que indica que o custo de ir da cidade de índice 1 até a cidade de índice 2 é 970. Os números de cada linha são separados por um espaço.

Já para o algoritmo que resolve o problema TSP simétrico (envoltório convexo), seu programa lerá do stdin as coordenadas cartesianas das cidades. A primeira cidade será identicada pelo índice 0, a segunda pelo índice 1 e assim sucessivamente. As coordenadas de cada ponto (cada linha) são separados por um espaço. Exemplo de entrada para n = 4 cidades:

1.0 1.0

1.0 5.0

5.0 5.0

5.0 1.0

A saída do programa deve conter a ordem das cidades que o caixeiro deve visitar (uma cidade por linha), de acordo com o algoritmo. Em seguida, deve-se imprimir o custo do tour encontrado de acordo com esse algoritmo, seguido de um asterisco. Por exemplo, a saída do programa para a entrada acima (matriz 3x3 apresentada) utilizando o algoritmo NN deve ser a impressão na tela de:

0

1

2

2157

*

9. Formatação de Entrada e Saída

O trabalho será testado da seguinte maneira:

./trab1 n algoritmo

onde:

- ***algoritmo*** é o agoritmo que utilizaremos para criar o tour. Pode ser ***exato***, ***nn, opt e hull***.
- ***n*** é o número de cidades que o caixeiro viajante deverá visitar.

10. Documentação

Na documentação deverá ser apresentada uma comparação minuciosa dos métodos implementados (tempos de execução e qualidade das soluções). Deverá, ainda, descrever como estes foram implementados. A documentação deverá ser feita em LaTeX.

11. **Avaliação**

- A nota do trabalho terá um total de 10 pontos;
- Serão contemplados com nota zero os trabalhos que se enquadrarem em uma ou mais situações abaixo:
  - Plágio;
  - Programa não compila;
  - Não está de acordo com as especificações.

12. **Considerações Importantes**

- Modularize o seu código adequadamente. Crie arquivos .c e .h para cada módulo do seu sistema. Em especial, crie arquivos exclusivos para manipular as estruturas de dados dos tipos abstratos de dados que você estiver representando.
- Seu programa deve ser, obrigatoriamente, compilado com o utilitário make. Crie um arquivo Makefile que gere como executável para o seu programa um arquivo de nome trab1.

13. **Entrega**

Este trabalho deve ser feito em grupo de até **2** componentes e entregue até o dia **03 de setembro de 2015**, às **23:59:59hs**.

Ele deve ser enviado para ***mberger@inf.ufes.br***.

O assunto da mensagem deve ser

ed2:trab1:nome1:nome2

Ex:

ed2:trab1:Mariella Berger:Jociel Andrade

O nome do arquivo com o trabalho enviado em anexo deve ser trab1.tar.gz. Ele será descompactado da seguinte forma:

tar -xvzf trab1.tar.gz

e deverá gerar um arquivo chamado Makefile, com as regras de compilação do

programa, os arquivos com os códigos dos programas e o arquivo com a documentação **em latex**.

Ao digitar:

make all

deve ser gerado o executável trab1 e a documentação em PDF a partir dos fontes em Latex.

O recebimento dos trabalhos é automatizado. Siga as instruções à risca pois algum erro na submissão pode inviabilizar a entrega do seu trabalho.

Não deixe para entregar seu trabalho no último instante. Você poderá perder o prazo e ter seu trabalho invalidado.

No dia seguinte à data de entrega será disponibilizado na página da disciplina a lista com os nomes de todos os trabalhos entregues no prazo. Verifique se seu nome estará na lista para evitar problemas futuros.

14. Observação Importante

Mais detalhes serão discutidos em sala de aula. Considerações feitas em sala terão mais relevância do que as contidas nesta especificação.

15. Dúvidas

Em caso de dúvidas no trabalho contate-me em *mariellaberger@gmail.com*.