Inteligência Artificial – 2016
Profa. Karina Valdivia Delgado

Primeiro Exercício Programa (EP1)

Problema: Roteamento de veículos

Composição dos grupos: de 3 alunos.

O problema de roteamento de veículos (*Vehicle Routing Problem* - VRP) é um problema NP-hard e tem aplicação prática no processo de distribuição de produtos e serviços. O objetivo é encontrar um conjunto ótimo de rotas para uma frota de veículos de modo a atender um conjunto de clientes com demandas conhecidas. Cada veículo tem uma capacidade fixa e deve partir e voltar para um ponto denominado depósito (ver Figura 1). Enquanto métodos exactos consomem muito tempo para ser aplicados no mundo real, métodos heurísticos e metaheurísticos retornam rotas quase ótimas considerando o tempo disponível.

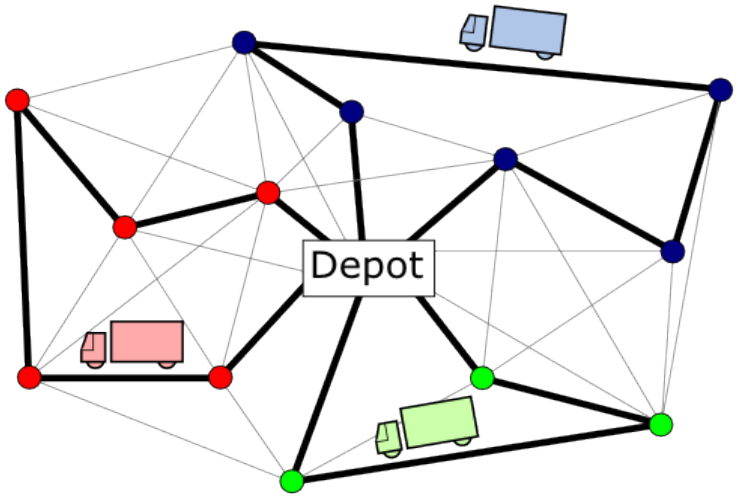

Figura 1: Problema de roteamento de veículos

Uma descrição detalhada do problema pode ser encontrado em What-is-VRP? de http://www.bernabe.dorronsoro.es/vrp/

1. Especificação do exercício programa

O objetivo do EP1 é usar busca local para resolver o problema de roteamento de veículos. O EP1 consiste em escolher e implementar um método metaheurístico para o problema (Tabu Search ou Simulated Annealing) já existentes na literatura.

Para a implementação podem apenas utilizar parte do código disponível na internet de http://hoonzis.blogspot.fr/2010/05/vehicle-routing-problem.html, código que tem a implementação dos algoritmos Clark & Wright e Sweep (se necessário), ou implementar o programa completo.

a) **Parte I:** Escolher um artigo que descreva claramente o algoritmo Tabu ou Simulated Annealing para resolver o problema, de modo que o algoritmo possa ser reproduzido.
b) **Parte II:** Implementar em Java o algoritmo escolhido, resolver as instâncias descritas na Seção 2 e elaborar um relatório. A entrega do programa no Tidia (incluindo arquivos fonte) e do relatório de no máximo 12 páginas, deve ser feita até o final do dia **15/04/2016.** Não serão aceitos envios por email. O relatório deve incluir:
    - Resumo
    - Introdução
    - Definição do problema VRP Capacitado
    - Pseudocógico e descrição do algoritmo escolhido
    - Implementação
        - Diagrama de classes
        - Dificuldades encontradas durante a implementação
    - Experimentos:
        - Configuração dos experimentos: computador, parâmetros utilizados
        - Uma tabela com o custo total da solução (soma do custo das rotas), o tempo em mili-segundos gasto pelo algoritmo implementado e o melhor resultado conhecido para cada instância.
        - Análise dos resultados obtidos
    - Conclusões

2. Instâncias a serem resolvidas

São 9 instâncias do Augerat, que podem ser encontrados em Instances-Capacitated-VRP-Augerat-et.al. de http://www.bernabe.dorronsoro.es/vrp/:

- A-n32-k5
- A-n33-k5
- A-n33-k6
- B-n31-k5
- B-n34-k5
- B-n35-k5
- P-n16-k8
- P-n19-k2
- P-n20-k2

A lista dos melhores resultados (comprimento total das rotas) conhecidos até o momento para

essas instâncias estão em Known-Best- Results de http://www.bernabe.dorronsoro.es/vrp/.

3. Entradas

Ver o formato detalhado das entradas das instâncias do Augerat em http://www.bernabe.dorronsoro.es/vrp/data/Doc.ps

A seguir a instância A-n32-k5 que possui 32 vértices (DIMENSION), o número mínimo de veículos a serem utlizados é 5 (Min no of trucks) e cada veículo tem capacidade máxima de 100 (CAPACITY). As coordenadas de cada um dos 32 vértices são listadas na seção NODE_COORD_SECTION. Na seção DEMAND_SECTION temos a demanda da cada um dos vértices.
Nessa instância o depósito é o vértice 1, que é dado na seção DEPOT_SECTION.

==========
NAME : A-n32-k5

COMMENT : (Augerat et al, Min no of trucks: 5, Optimal value: 784)

TYPE : CVRP

DIMENSION : 32

EDGE_WEIGHT_TYPE : EUC_2D

CAPACITY : 100

NODE_COORD_SECTION

1 82 76

2 96 44

3 50 5

4 49 8

5 13 7

6 29 89

7 58 30

8 84 39

9 14 24

10 2 39

11 3 82

12 5 10

13 98 52

14 84 25

15 61 59

16 1 65

17 88 51

18 91 2

19 19 32

20 93 3

21 50 93

22 98 14

23 5 42

24 42 9

25 61 62

26 9 97

27 80 55

28 57 69

29 23 15

30 20 70

31 85 60

32 98 5

DEMAND_SECTION

1 0

2 19

3 21

4 6

5 19

6 7

7 12

8 16

9 6

10 16

11 8

12 14

13 21

14 16

15 3

16 22

17 18

18 19

19 1

20 24

21 8

22 12

23 4

24 8

25 24

26 24

27 2

28 20

29 15

30 2

31 14

32 9

DEPOT_SECTION

1

-1

EOF
==========

Para todas as instâncias assumiremos que será usada a distância euclidiana para calcular a distância entre dois vértices. Sejam x[i] e y[i] as coordenadas do vértice i. A distância de i até j (dij) pode ser calculada por:

xd = x[i] − x[j]

yd = y[i] − y[j]

dij = nin (sqrt (xd ∗ xd + yd ∗ yd))

Em que sqrt é a função raiz quadrada, e nint é a função inteiro mais próximo (NearestIntegerFunction). Por exemplo, a distância entre o depósito com coordenadas 82 76 e o cliente 1 (vertice 2) com coordenadas 96 44 é 35.

4. Formato de Saída

Para cada rota temos a lista de clientes atendidos, o custo (comprimento) de cada rota e a demanda atendida. Após mostrar os dados de cada uma das rotas temos o custo total da solução (soma do custo das rotas) e o tempo em mili-segundos gasto pelo algoritmo. Um exemplo de saída é mostrado a seguir:

```
Rota #1: 0 27 24 0 custo: 59 demanda atendida: 44
Rota #2: 0 26 7 13 17 19 31 21 0 custo: 155 demanda atendida: 98
Rota #3: 0 6 3 2 23 4 11 28 14 0 custo: 230 demanda atendida: 98
```

```
Rota #4: 0 30 16 1 12 0 custo: 73 demanda atendida: 72
Rota #5: 0 20 5 25 10 29 15 22 9 8 18 0 custo: 268 demanda atendida: 98
Custo 785
Tempo 300
```

Por exemplo, Rota #1: 0 27 24 0 representa a rota que começa no depósito (vértice 1), passa pelo cliente 27(vértice 28) e cliente 24 (vértice 25) e finalmente retorna ao depósito (vértice 1). A distância Euclidiana entre eles é: 26, 8 e 25 respectivamente; fazendo um total de 59 (custo da rota) e a demanda atendida dos cliente é 20+24=44.

Note que a solução encontrada é 785 e está bem próxima da melhor solução conhecida, definida na entrada, que é 784.