Universidade Federal de Alfenas
Instituto de Ciências Exatas
Ciência da Computação

Relatório de Pesquisa Operacional
Problema do Caixeiro Viajante - parte 2

Alunos:
Alexandre William Miya - RA: 2014.1.08.004
Gustavo Alves Miguel - RA: 2014.1.08.013

Professor:
Humberto César Brandão de Oliveira

Junho
2018

Conteúdo

1 Apresentação

O Problema do Caixeiro Viajante (PCV) é um problema que tenta determinar a menor rota para percorrer uma série de cidades (visitando uma única vez cada uma delas), retornando à cidade de origem. Ele é um problema de otimização NP-difícil inspirado na necessidade dos vendedores em realizar entregas em diversos locais (as cidades) percorrendo o menor caminho possível, reduzindo o tempo necessário para a viagem e os possíveis custos com transporte e combustível.

2 Descrição da Atividade

O problema do caixeiro-viajante consiste na procura de um circuito que possua a menor distância, começando numa cidade qualquer, entre várias, visitando cada cidade precisamente uma vez e regressando à cidade inicial.

As variáveis podem ser definidas como:

1. i : representa a cidade partida;
2. j : representa a cidade destino;
3. d_{ij}: matriz de distância euclidiana entre as cidades i e j;
4. x_{ij}: matriz booleana representando a matriz de adjacência do grafo;

3 Modelo Matemático

A função objetivo pode ser expressa da seguinte forma:

$$\text{Min} \sum_{i \in I} \sum_{j \in J} d_{\mathrm{ij}}.x_{\mathrm{ij}}$$

Sujeito a:

$$\forall \, \mathrm{i} \in I, \, x_{\mathrm{ii}} = 0 \ ,$$

$$\forall \, \mathrm{i} \in I, \, \sum_{j \in J} x_{\mathrm{ij}} = 1 \ ,$$

$$\forall \, \mathrm{j} \in J, \, \sum_{i \in I} x_{\mathrm{ij}}) = 1 \ ,$$

Além disto, tratamos a existência de ciclos na solução. Nesta implementação o tratamento dos ciclos é feito utilizando a classe **LazyConstraintCallback** para adicionar as restrições de ciclo dinamicamente.

4 Resultados

O experimento foi realizado em computador com processador i7 e 16Gb de memória RAM. A seguir é apresentado o resultado encontrado com a solução:

```
Total (root+branch&cut) =    0.03 sec. (9.14 ticks)

0 --> 3 --> 4 --> 2 --> 9 --> 8 --> 1 --> 6 --> 5 --> 7 --> 0

Valor min: 264.14260909856404
CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo)
```

Figura 1: Resultado para o problema do caixeiro viajante com 10 cidades **sem** a classe lazyconstraintcallback.

```
Total (root+branch&cut) =    0.13 sec. (32.55 ticks)

0.0 -0.0 0.0 0.0 -0.0 0.0 -0.0 1.0 -0.0 -0.0
-0.0 0.0 -0.0 -0.0 -0.0 -0.0 0.0 -0.0 1.0 -0.0
0.0 -0.0 0.0 0.0 1.0 -0.0 -0.0 0.0 -0.0 -0.0
1.0 -0.0 0.0 0.0 0.0 -0.0 -0.0 0.0 -0.0 -0.0
-0.0 -0.0 0.0 1.0 0.0 -0.0 -0.0 -0.0 -0.0 -0.0
-0.0 -0.0 0.0 0.0 -0.0 0.0 1.0 0.0 0.0 0.0
-0.0 1.0 -0.0 -0.0 -0.0 -0.0 0.0 -0.0 -0.0 -0.0
0.0 -0.0 0.0 0.0 -0.0 1.0 -0.0 0.0 -0.0 -0.0
-0.0 0.0 -0.0 -0.0 -0.0 0.0 -0.0 -0.0 0.0 1.0
-0.0 -0.0 1.0 -0.0 -0.0 -0.0 -0.0 -0.0 0.0 0.0

0 --> 7 --> 5 --> 6 --> 1 --> 8 --> 9 --> 2 --> 4 --> 3 --> 0

Valor min: 264.14260909856404
CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo)
```

Figura 2: Resultado para o problema do caixeiro viajante com 10 cidades **com** a classe lazyconstraintcallback.

5 Considerações

Quando testado para instâncias maiores que 10 cidades, a implementação demonstra ser demasiadamente lenta.Ressaltamos também que a implementação

sem o lazyconstraintcallback para a instância de 10 cidades apresentou-se ser mais eficiente em comparação com a utilização do lazyconstraintcallback