Trabalho Prático Grafos - Análise de métodos heurísticos para resolução do problema do caixeiro viajante

Cláudio Silva - 3492 [1], *Isabella Ramos - 3474* [2] *Luciano Belo - 3897* [3] *Guilherme Aguiar- 3496* [4]

[1234]Universidade Federal de viçosa - Campus Florestal

author@university.edu, guilherme.aguiar@ufv.br

Abstract

O objetivo deste trabalho consiste em apresentar dois métodos que incorporam uma heurística construtiva e de melhoria para solucionar o problema do caixeiro viajante. Para tal, foram implementadas as heurísticas construtivas de vizinho mais próximo e vizinho mais distante e ambas são otimizadas utilizando a heurística de melhoria 2-opt, constituindo, assim, dois métodos distintos. Todas as heurísticas são incorporadas em uma biblioteca de tratamento de grafos previamente construída. Para cada método foram feitas 30 simulações com análise da melhor e pior performance e demais métricas.

1. Introdução

Problema do Caixeiro Viajante (PCV) mundialmente conhecido como *Travelling Salesman Problem* (TSP) consiste em um problema de determinação de rota com o menor custo para um vendedor cujo proposito é visitar um conjunto finito de cidades. Para tal, deve-se determinar a menor rota para percorrer tais cidades (visitando uma única vez cada uma delas) e retornando à cidade de origem. No domínio da teoria dos grafos (Figura 4), cada cidade é identificada com um nó (ou vértice) e as rotas que ligam cada par de nós são os arcos/arestas. Cada uma das arestas terá como peso associado a distância correspondente. Desde que seja possível ir diretamente de uma cidade para qualquer outra, o gráfico diz-se completo. Uma viagem que passe por todas as cidades corresponde a um ciclo Hamiltoniano, representado por um conjunto específico de linhas.

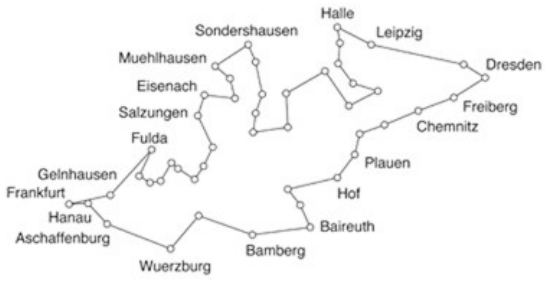

Figure 1: *Grafo representando um ciclo entre cidades.*

Um dos objetos de estudo deste trabalho é a distância total do ciclo (*tour*), obtida pelo somatório das distâncias das linhas presentes no mesmo. Formalmente, o problema pode ser representado por um grafo G(V, E), com número de vértices maior que 3 e pesos referentes a cada uma das arestas.

Foram implementados dois métodos heurísticos para a construção e resolução deste problema, que serão descritos com mais detalhes a seguir. Tais métodos foram aplicados as instâncias: **berlin52**, **ch130** e **d198**, da TSPLIB [1]. Tratando-se de um problema de otimização NP-difícil, o estudo propõe apresentar os resultados obtidos, tendo em vista que nem sempre a solução ótima será alcançada. A organização desse trabalho é descrita a seguir. As seções 2 e 3 são compostas das exposições das heurísticas utilizadas. A seção 4 apresenta a integração dos métodos na biblioteca.. A seção 5 apresenta os resultados alcançados. Por fim, a seção 6 conclui o trabalho.

2. Heurísticas Construtivas

As instâncias da TSPLIB utilizam o padrão EUC 2D [2], onde as distancias calculadas serão os pesos das arestas. Elas fornecem dados de um determinado número de cidades com a sua posição gerada através de coordenadas, contendo o número do vértice (representando a cidade) e duas coordenadas de posicionamento. Dito isso, para construir um grafo que represente um problema PCV, utilizou-se heurísticas construtivas, sendo elas a inserção do vizinho mais próximo e a Inserção do vizinho mais distante.

2.1. Inserção do vizinho mais próximo

No nosso trabalho, uma das heurísticas construtivas utilizadas foi a inserção do vizinho mais próximo[3]. O objetivo do algoritmo é construir uma rota adicionando a cada passo, a cidade mais próxima da última cidade inserida e que ainda não foi visitada. Essa heurística gera rapidamente um caminho curto, mas que dificilmente será o ótimo.

Os passos para a construção do algoritmo são explicadas nos itens abaixo:

1. Inicializar a rota com apenas um nó escolhido aleatoriamente.
2. Encontrar o nó k fora da rota atual cuja aresta que o liga ao nó atual possui custo mínimo.
3. Adicionar o nó k no final da lista de nós visitados.
4. Se todos os nós foram adicionados parar, senão voltar ao passo 2.

Na nossa biblioteca, a função responsável pela heurística do vizinho mais próximo é chamada de *nearest_neighbor_tour*.

2.2. Inserção do vizinho mais distante

A outra heurística construtiva utilizada é a inserção do vizinho mais distante. Que, de maneira análoga ao vizinho mais próximo, tem como objetivo construir uma rota, adicionando a cada passo a cidade mais distante da última cidade inserida e que ainda não foi visitada.

Os passos para a construção do algoritmo são explicadas nos itens abaixo:

1. Inicializar a rota com apenas um nó escolhido aleatoriamente.

2. Encontrar o nó k fora da rota atual cuja aresta que o liga ao nó atual possui custo máximo.
3. Adicionar o nó k no final da lista de nós visitados.
4. Se todos os nós foram adicionados parar, senão voltar ao passo 2.

Na nossa biblioteca, a função responsável pela heurística do vizinho mais próximo é chamada de *furthest_neighbor_tour*.

3. Heurística de Otimização

A heurística de otimização escolhida foi a 2-opt [4]. A ideia principal da heurística de otimização 2-opt, consiste em realizar buscas locais de aresta ótima para um dado vértice em análise e realizando diversas permutações até que uma a aresta de menor custo seja encontrada. Para isso, é necessário que uma heurística construtiva seja previamente executada criando um caminho ruim para que posteriormente possa ser melhorado com a 2-opt. O algoritmo recebe como entrada uma rota ruim e possui como retorno uma rota otimizada.

A parte central do algoritmo funciona da seguinte forma: a cada vértice marcado percorre-se todos os demais pontos restantes exceto ele mesmo e calcula para cada um o valor da possível aresta, utilizando as coordenadas geográficas fornecidas. Para a melhor aresta assinala-se os os vértices inicial e final envolvidos. Após o loop percorre-se o vetor da posição do vértice inicial há posição do vértice final e insere os vértices de forma invertida para que a inserção dos mesmos não prejudique a disposição dos demais.

A função é invocada em um loop externo que contabiliza as mudanças de otimalidade bem como análise o tempo gasto, cujo limite no loop interno é de 3 min.

4. Integração do algoritmo com a biblioteca

A figura ilustra como é feita a integração dos métodos na biblioteca

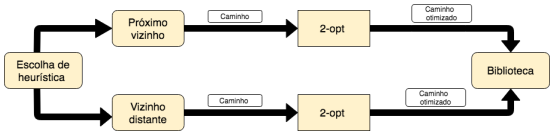

Figure 2: *Esquema de integração dos métodos na biblioteca.*

5. Execução dos métodos

Junto ao código fonte encontra-se o arquivo *makefile*, responsável pela compilação e execução de nossa biblioteca. Os códigos foram executados em sistemas operacionais diferentes, portanto apresentam opções diferentes. Como trata-se de uma adição de funcionalidades da biblioteca criada no Trabalho Prático I, optou-se por utilizar os métodos que geram os grafos como uma funcionalidade de inserção, tendo o arquivo desejado junto a linha de comando. Ao final de qualquer uma das inserções o grafo gerado ótimo é armazenado, podendo ter as funcionalidades da biblioteca aplicadas sobre ele.

5.1. Modo normal

O modo de execução normal mantém a inserção de grafos por meio de arquivo ".txt" para a criação da lista de adjacências, utilizando os seguintes comandos:

python3 main.py -o -normal *caminho do arquivo.txt*

5.2. Inserção: vizinho mais próximo

Os gráficos serão executados 30 vezes, utilizando a heurística de vizinho mais próximo com otimização (2-opt), gerando um arquivo ".txt" com os valores ótimos obtidos e o ciclo hamiltoniano que obteve o melhor resultado (menor distância) é armazenado na memória do programa, para ser utilizado.

python3 main.py -o -nn *caminho do arquivo.tsp*

5.3. Inserção: vizinho mais distante

Similar a inserção por vizinho mais próximo, utilizando a flag -fn:

python3 main.py -o -fn *caminho do arquivo.tsp*

6. Resultados

O objetivo dessa seção é apresentar os resultados colhidos com os testes dos dois métodos implementados. Para cada um dos métodos em cada um dos cenários foram feitas 30 execuções com posição de cidade inicial aleatória para garantir maior variabilidade nos caminhos construídos em cada um dos testes. Para critério de parada da heurística de otimização contabiliza-se 10 iterações sem otimização ou 3 minutos de execução de loop interno.

Método	Melhor	Pior	Média	Desvio-padrão
1	7748.34	8310.11	7903.2	140.16
2	8030.78	8309.82	8108.29	102.26

Table 1: *Instância berlin52 - Resultado de 30 execuções*

Método	Melhor	Pior	Média	Desvio-padrão
1	6329.35	6733.88	6477.17	130.32
2	6536.55	6690.16	6603.55	68.24

Table 2: *Instância ch130 - Resultado de 30 execuções*

Método	Melhor	Pior	Média	Desvio-padrão
1	15987.96	17200.19	16380.6	395.11
2	16186.1	16438.98	16219.82	87.43

Table 3: *Instância d198 - Resultado de 30 execuções*

Os resultados obtidos são exibidos nos gráficos e nas tabelas. Com os gráficos é possível obter uma clara visão sobre o comportamento do algoritmo frente ao caso ótimo para cada um dos possíveis cenários. O primeiro gráfico apresenta as menores distâncias alcançadas pelas heurísticas implementadas e o segundo exibe a médias das distâncias.

Método	Resultado ótimo
berlin52	7542
ch130	6110
d198	15780

Table 4: *Resultados ótimos das instâncias*

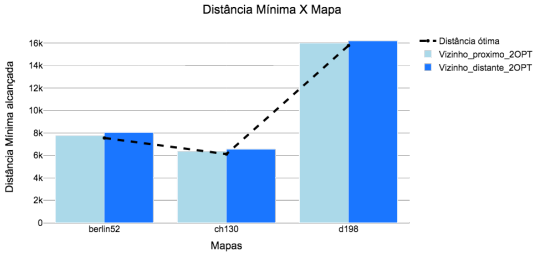

Figure 3: *Relação de distância mínima.*

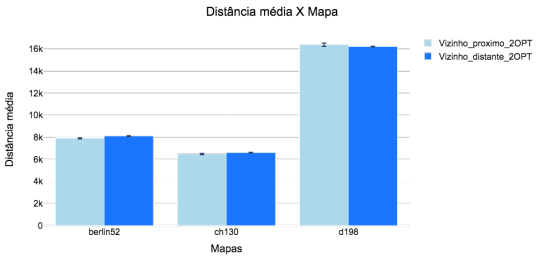

Figure 4: *Relação de distância média.*

Percebe-se que para todos os casos a combinação de heurísticas de vizinho próximo e otimização 2opt resulta em caminhos de distância associada menor. Isso se deve ao fato de que a construção da heurística de vizinho mais próximo sempre escolhe as menores distâncias para formar as primeiras arestas, o que ocorre ao contrário na heurística de vizinho mais distante.

7. Conclusão

Este estudo apresentou resultados interessantes, com relevância para avaliar o funcionamento de dois métodos de resolução de um PCV. Os algoritmos apresentaram soluções viáveis para o problema mas, como esperado, nem sempre as distâncias encontradas foram as ótimas. Tratando-se de heurísticas que também utilizam valores aleatórios para a construção dos grafos, é esperado a oscilação encontrada na seção de resultados. A utilização de outras heurísticas de melhoramento, mais eficientes que a 2-opt, poderia ser feita, buscando resultados melhores.

O limite de tempo e de iterações sem melhoria como critério de parada também poderia ser alterado buscando resultados melhores, pois o critério de parada tem impacto maior, conforme o número de cidades (vértices) aumenta. Outro detalhe que também merece menção é o comportamento do tempo de execução. Nos 30 testes realizados, tratando-se de um algoritmo que não é polinomial visando a resolução de um problema NP-completo, o tempo necessário aumenta muito à medida que o número de cidades da instância aumenta.

8. References

[1] G. Reinhelt, "{TSPLIB}: a library of sample instances for the tsp (and related problems) from various sources and of various types," *URL: http://comopt. ifi. uniheidelberg. de/software/TSPLIB95*, 2014.

[2] Valmir. [Online]. Available: https://docs.ufpr.br/ volmir/PO$_I I_1 2_T SP.pdf$

[3] C. A. Hurkens and G. J. Woeginger, "On the nearest neighbor rule for the traveling salesman problem," *Operations Research Letters*, vol. 32, no. 1, pp. 1–4, 2004.

[4] M. Englert, H. Röglin, and B. Vöcking, "Worst case and probabilistic analysis of the 2-opt algorithm for the tsp," in *SODA*. Citeseer, 2007, pp. 1295–1304.