Teoria dos Grafos - O Problema do Caixeiro Viajante, Algoritmo Heurístico e Algoritmo Exato

Daniel de Souza Rodrigues - 18.2.8112 - Sistemas de Informação

[1]Instituto de Ciências Exatas e Aplicadas – Universidade Federal de Ouro Preto (UFOP)
Rua 36, Número 115 - Bairro Loanda, João Monlevade - CEP: 35931-008

daniel.sr@aluno.ufop.edu.br

Abstract. *In this article will be presented one of the most studied problem of computing called "Traveling Salesman Problem", this problem has several applications in logistics and planning and it is a problem of the NP-Difficult class, we will approach two algorithms being one of them heuristic "Greedy Algorithm of the Nearest Neighbor "and an exact" Brute-Force Algorithm "aiming to understand and find satisfactory solutions to the problem presented.*

Resumo. *Neste artigo será apresentado um dos problema mais estudados da computação que se chama "Problema do Caixeiro Viajante", este problema possui diversas aplicações em logística e planejamento e é um problema da classe NP-Difícil, abordaremos dois algoritmos sendo um deles heurístico "Algoritmo Guloso do Vizinho Mais Próximo" e um exato "Algoritmo de Força-Bruta" visando compreender e encontrar soluções satisfatórias para o problema apresentado.*

1. Introdução

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 da classe NP-difícil inspirado na necessidade dos vendedores em realizar entregas em diversos locais, percorrendo o menor caminho possível, reduzindo o tempo necessário para a viagem e os possíveis custos com transporte e combustível.[Netto and Jurkiewicz 2017]

1.1. Entendendo problemas *NP*

Na teoria da complexidade computacional, *NP* é o acrônimo em inglês para Tempo polinomial não determinístico (Non-Deterministic Polynomial time) que denota o conjunto de problemas que são decidíveis em tempo polinomial por uma máquina de Turing não-determinística. Uma definição equivalente é o conjunto de problemas de decisão que podem ter seu certificado verificado em tempo polinomial por uma máquina de Turing determinística.[Stackoverflow 2021]

- *NP*-Difícil (ou *NP*-Hard, ou *NP*-Complexo) na teoria da complexidade computacional, é uma classe de problemas que são, informalmente, "Pelo menos tão difíceis quanto os problemas mais difíceis em *NP*"
- *NP*-Completo é uma maneira de mostrar que certas classes de problemas não são solucionáveis em tempo realista.

2. Ciclos Hamiltonianos e o Problema do Caixeiro Viajante (PCV)

Um ciclo hamiltoniano é um ciclo que inclui cada vértice de um grafo exatamente uma vez[Fonseca 2021], como mostra o exemplo abaixo:

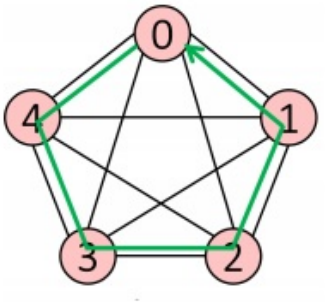

Figura 1. Ciclo Hamiltoniano

O Problema do Caixeiro Viajante (PCV) consiste em encontrar o **ciclo hamiltoniano de custo mínimo** em um grafo ponderado.

- Apesar de intuitivamente parecer fácil encontrar um ciclo hamiltoniano, esse problema é extremamente desafiador e pertence a classe *NP*-Completo.

3. Possíveis Soluções

Nesta seção serão abordados dois algoritmos que visam solucionar o Problema do Caixeiro Viajante (PCV).

3.1. Abordagem Exata - Algoritmo de Força-Bruta

Este algoritmo visa resolver o problema do caixeiro viajante examinando todas as rotas possíveis, por ser uma solução exata ele garante a melhor rota possível mas em compensação o seu tempo de execução em alguns casos é inadequado à realidade.

Abaixo podemos ver o pseudocódigo do Algrotimo de Força-Bruta.[Fonseca 2021]

Algoritmo 3: FORCA BRUTA.

Entrada: (i) Um grafo $G = (V, E, w)$.
Saída: (i) O ciclo hamiltoniano de custo mínimo C^{best}.

```
1 cost^min ← inf;
2 C^best ← null;
3 para cada permutação de vértices C faça
4 |   C ← C ∪ {C[0]};
5 |   cost^C ← Calcule o custo da rota C;
6 |   se cost^min > cost^C então
7 |   |   cost^min ← cost^C;
8 |   |   C^best ← C;
9 retorna C^best;
```

Figura 2. Pseudocódigo Força-Bruta

3.2. Abordagem Heurística - Algoritmo Vizinho Mais Próximo

Conhecido de maneira informal como “Algoritmo Guloso do Vizinho Mais Próximo”, essa abordagem heurística trabalha da seguinte maneira: A partir de um vértice origem, o próximo vértice a ser visitado será sempre o vértice de menor custo que ainda não foi visitado.

Abaixo podemos ver o pseudocódigo do Algrotimo Vizinho Mais Próximo.[Fonseca 2021]

Algoritmo 2: VIZINHO MAIS PRÓXIMO.

Entrada: (i) Um grafo $G = (V, E, w)$.
Saída: (i) Um ciclo hamiltoniano C.

```
1  u ← 0;
2  C ← ∅;
3  Q ← V − {u};
4  enquanto Q ≠ ∅ faça
5      v ← adjacente não visitado de menor distância a u;
6      C ← C ∪ {v};
7      Q ← V − {v};
8      u ← v;
9  C ← C ∪ {C[0]};
10 retorna C;
```

Figura 3. Pseudocódigo Algoritmo Vizinho Mais Próximo

3.3. Abordagem Exata *vs* Abordagem Heurístca

Na abordagem Exata temos como vantagem o resultado entregue, pois sempre será a melhor rota possível, mas deve ser considerado que:

- O tempo computacional é super elevado até para análises em grafos com poucos vértices.

Na abordagem Heurística temos como vantagem o tempo de execução de baixo custo até mesmo para grafos com muitos vértices, mas existe uma grande desvantagem a ser considerada.

- Raramente a melhor rota é encontrada pois como o algoritmo trabalha de forma a visitar cada vértice não visitado, no final do ciclo ele é forçado a percorrer o último caminho existente que leva a origem, na maioria dos casos esse último passo sacrifica todos os outros vértices percorridos pois ele pode possuir um custo muito alto, sendo assim a rota entregue não possui o ciclo hamiltoniano de custo mínimo.

4. Testes dos Algoritmos

Para execução dos algoritmos foram ultilizadas as seguintes ferramentas:

1. Foi utilizada a linguagem Python para implementação dos algoritmos.
2. Link do Repositório: Problema do Caixeiro Viajante - GitHub Daniel Rodrigues
3. Todos os testes foram executados com Processador: $AMD - Ryzen3 - 2200g$
4. Memorória RAM: $Ballistix$ - $8GB\ DDR4\ 2400mhz$

4.1. Validação dos algorimos implementados em Python

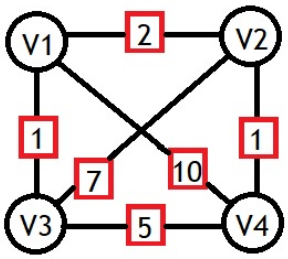

Figura 4. Grafo de Teste - Validação

O ”Grafo de Teste - Validação”foi utilizado para validar os resultados fornecidos, com ele é possível realizar a verificação de forma manual caso necessário, vale levar em consideração que esse grafo admite mais de uma rota otíma possivel.

Tabela 1. Resultados obtidos através da execução do Algoritmo

ALGORITMO	ROTA	CUSTO	TEMPO
Força-Bruta	[0, 1, 3, 2, 0]	9.0	0.0s
Vizinho Mais Próximo	[0, 2, 3, 1, 0]	9.0	0.0s

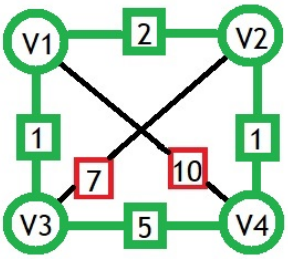

Figura 5. Rota Resultante - Validação

Dado que todos os resultados gerados são válidos e podem ser conferidos manualmente utilizando o pseudocódigo de cada algoritmo, assume-se que a implementação dos algoritmos está correta e que grafos maiores poderão ser processados e os resultados gerados serão verídicos.

4.2. Testes com grafos gerados aleatoriamente

Os mesmos grafo gerado aleatoriamente foram utilizado para testar cada um dos algoritmos, consirerações:

- E.T = Extrapolou o tempo
- E.M = Extrapolou a memória
- R.G.E = Rota grande demais para ser exibido
- $|V|$ = Quantidade de vértices
- w^{min} = Peso mínimo de cada aresta
- w^{max} = Peso máximo de cada aresta

Tabela 2. Resultados obtidos através da execução do Algoritmo Guloso do Vizinho Mais Próximo

$\|V\|$	w^{min}	w^{max}	CUSTO	TEMPO	ROTA
5	1	5	11	0.0s	[0, 2, 1, 4, 3, 0]
5	1	500	1287	0.0s	[0, 1, 3, 2, 4, 0]
8	1	5	15	0.0s	[0, 3, 5, 1, 2, 6, 7, 4, 0]
8	1	500	1123	0.0s	[0, 5, 4, 6, 1, 3, 7, 2, 0]
10	1	5	16	0.0s	[0, 5, 2, 8, 7, 6, 9, 1, 4, 3, 0]
10	1	500	738	0.0s	[0, 8, 5, 3, 4, 1, 7, 2, 9, 6, 0]
20	1	5	27	0.0s	R.G.E
20	1	500	1986	0.0s	R.G.E
50	1	5	58	0.0s	R.G.E
50	1	500	2341	0.0s	R.G.E

Tabela 3. Resultados obtidos através da execução do Algoritmo de Força-Bruta

$\|V\|$	w^{min}	w^{max}	CUSTO	TEMPO	ROTA
5	1	5	11	0.0s	[0, 2, 1, 4, 3, 0]
5	1	500	1070	0.0s	[0, 1, 3, 4, 2, 0]
8	1	5	9	0.08s	[0, 5, 3, 7, 4, 1, 2, 6, 0]
8	1	500	903	0.6s	[0, 1, 3, 7, 2, 5, 4, 6, 0]
10	1	5	16	8.3s	[0, 3, 4, 1, 9, 2, 8, 7, 6, 5, 0]
10	1	500	608	8.59s	[0, 1, 4, 3, 5, 8, 6, 7, 2, 9, 0]
20	1	5	E.T	E.T	E.T
20	1	500	E.T	E.T	E.T
50	1	5	E.T	E.T	E.T
50	1	500	E.T	E.T	E.T

5. Conclusão

Com os resultados obtidos é possível ver claramente as desvantagens de tempo computacional existentes na utilização do algoritmo *"Força-Bruta"*, visto que para grafos que possuem apenas 20 vértices são necessários mais de 10 minutos de processamento, já com a utilização do algoritmo *"Vizinho Mais Próximo"* esse processamento leva poucos milissegundos.

Cabe ressaltar que apesar do algoritmo *"Vizinho Mais Próximo"* não entregar sempre resultados com a verdadeira rota ótima, a sua utilização se torna muito mais interessante para grafos de grande porte, sendo possível fazer várias execuções e traçar caminhos ágeis em tempo hábil.

Todos esses fatores devem ser levados em consideração para escolher o algoritmo que melhor condiz com as necessidades da sua aplicação.

Referências

[Fonseca 2021] Fonseca, G. H. G. (2021). A07 percursos abrangentes. Online; acessado em 20 de abril de 2021.

[Netto and Jurkiewicz 2017] Netto, P. O. B. and Jurkiewicz, S. (2017). *Grafos: introdução e prática*. Editora Blucher.

[Stackoverflow 2021] Stackoverflow (2021). O que é um problema np completo? Online; acessado em 20 de abril de 2021.