O Problema do Caxeiro Viajante: Aplicação e Análise de Algoritmos

Marcos Henrique Santos Cunha[1] **Natália Pereira de Almeida**[1]

[1]Instituto de Ciências Exatas e Aplicadas – Universidade Federal de Ouro Preto (UFOP)
Minas Gerais – MG – Brasil

marcos.hsc@aluno.ufop.edu.br; natalia.pa@aluno.ufop.edu.br

Abstract. *The main objective of this article is to delineate the Traveling Salesman Problem* (TSP)*. It will be shown its real applications, together with their solution algorithms, such as the Nearest Neighbour and Brute Force algorithms. Besides that, their advantages, disadvantages and pseudocode will also be presented, in order to explain with clarity and precision the problem. Then, tests and implementation results will be presented, where the comparison between their execution time will be observed and discussed. At the end, the obtained conclusions from the construction of this article will be discussed.*

Resumo. *Este artigo tem por objetivo central delinear o Problema do Caxeiro Viajante* (PCV)*. Salienta-se que serão abordadas as suas aplicações reais, juntamente com seus algoritmos de solução, vulgo os algoritmos do Vizinho mais Próximo e o de Força Bruta. Ademais, destaca-se que também serão apresentadas as vantagens e desvantagens de cada um e seus pseudocódigos, a fim de apresentar com precisão e clareza o problema em questão. Em seguida, serão apresentados os testes e resultados da implementação, onde será observada e discutida a comparação entre seus tempos de execução. Ao final, as conclusões obtidas a partir deste trabalho serão discutidas.*

1. Introdução

Aplicações de conceitos e algoritmos que envolvam a Teoria dos Grafos são comumentes utilizados no cotidiano de diversas empresas, em virtude do alto grau de utilidade, competitividade e lucros que a solução destes problemas podem trazer, ainda que não seja encontrada a melhor solução, isto é, sendo encontrada uma aproximação da mesma.

Destaca-se que, um dos problemas presentes é o Problema do Caxeiro Viajante. De acordo com [GOLDBARG and LUNA 2000], o Problema do Caixeiro Viajante (PCV), em inglês é conhecido como Travelling Salesman Problem (TSP), é um dos mais notórios e tradicionais problemas da computação. Ele recebe esse nome pois pode ser descrito como sendo o problema de um vendedor que viaja por várias cidades e deseja determinar a rota mais curta para visitar cada cidade uma única vez, e voltar a sua origem. Embora, aparentemente sua solução pareça fácil, esse é um problema NP-Completo, ou seja, não existe algoritmo capaz de garantir a melhor solução em tempo polinomial. Tal fato está correlacionado à alta quantidade de rotas a serem analisadas em grafos grandes. Por exemplo, em um grafo de 10 vértices K^{10}, temos $10!$ rotas distintas, e para um grafo completo K^n temos $n!$ diferentes combinações de caminhos possíveis.

É válido evidenciar quais são as aplicações reais desse problema, e quais são os principais algoritmos que os resolvam. Em primeiro plano, pode-se afirmar que o Problema do Caxeiro Viajanete está conrrelacionado a diversas questões de logística e planejamento, tais como, perfuração de placas de circuitos impressos, roteamento de ônibus escolares e agendamentos de equipes de trabalho.

Ademais, em conformidade com [REINELT 1994], o estudo do (PCV) tem atraído pesquisadores de diferentes campos, entre os quais estão presentes a pesquisa operacional, matemática, física, biologia e a inteligência artificial. Isso se deve ao fato de que através do (PCV) é possível encontrar estratégias de solução que envolvem questões de algo grau de relevância, como a busca tabu, algoritmos genéticos, redes neurais, entre outros.

Em segundo plano, destaca-se que, nas primeiras seções deste artigo, serão apresentados os dois algoritmos que solucionam o PCV: o Algoritmo do Vizinho mais Próximo e o Algoritmo de Força Bruta. Por conseguinte, também serão abordado os testes e os resultados de cada um desses algoritmos para diferentes rotas com o objetivo central de apresentar as vantagens e desvantagens de cada um. Por fim, a última seção será denifinida com as conclusões obtidas a partir da análise dos resultados, seguido das referências bibliográficas. Ressalta-se que, todo o código referente ao assunto discorrido neste artigo, o qual foi implementado na lingugagem Python, poderá ser visualizado através de um repositório do GitHub [CUNHA and ALMEIDA 2020]. Dessa maneira, será possível obter de forma clara e objetiva todo o conteúdo que será abordado.

2. Algoritmos

Nesta seção, serão detalhados os algoritmos citados anteriormente, junto com os seus pseudocódigos, bem como também a explicação do funcionamento de cada um.

2.1. Algoritmo do Vizinho mais Próximo

Esse algoritmo é uma abordagem heurística, visto que ele não garante encontrar a melhor solução para o problema, buscando encontrar uma solução aproximada, suficientemente satisafatória. O seu funcionamento consiste em caminhar sempre para o vizinho mais próximo não visitado, até que se forme um ciclo hamiltoniano. Este algoritmo apresenta uma complexidade computacional de pior caso da ordem de $O(n^2)$, para um grafo completo K^n.

Na Figura 1 [FONSECA 2020], é apresentado um exemplo simplificado de rastreio do algoritmo para a abordagem do transporte do Correios na região da cidade de João Monlevade, Minas Gerais. O problema foi representado como um grafo, onde cada região da cidade que necessita de entregas foi mapeada como um vértice, e as arestas representam as distâncias entre cada uma das regiões. Nesta figura, pode-se observar que o algoritmo encontrou a rota 0-1-4-5-2-3-0 com custo de 22.4. Contudo, já podemos observar que o algoritmo não foi eficiente para esse caso, uma vez que existem rotas menores, vulgo 0-1-4-3-2-5-0 com custo de 16.9.

Este algoritmo raramente encontra a melhor solução, porém possui uma grande vantagem que o permite ter alto desempenho. Essa está interligada no fato dele garantir um baixíssimo tempo de execução, mesmo para grafos grandes, podendo executar em frações de segundos.

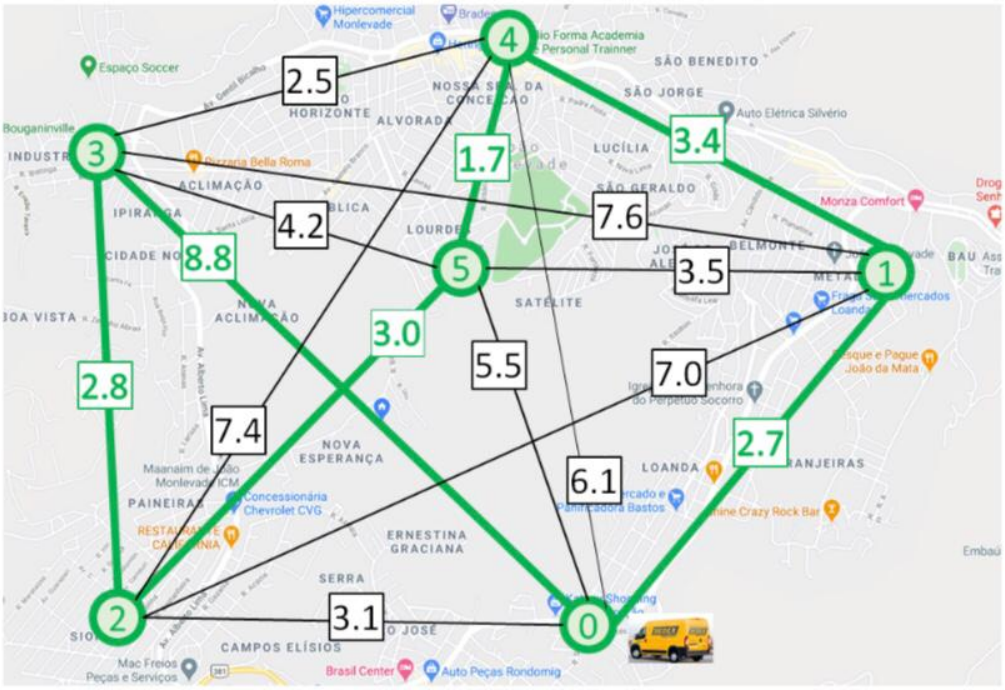

Figura 1. Rota feita pelo Algoritmo do Vizinho mais Próximo, na região de João Monlevade, MG, para exemplificar o transporte de Correios

No Algoritmo 1 [FONSECA 2020], é delineado o pseudocódigo do Algoritmo do Vizinho Mais Próximo. Nas linhas 1-3 é feita a inicialização necessária para a execução do algoritmo. Logo em seguida, procura-se o vértice $v \in Q$ de menor distância a u, então este é adicionado à sequência de vértices visitados C, e então removido do conjunto Q, para que não seja mais considerado. Os os próximos vértices considerados serão os adjacentes à v. Este processo é repetido até que o conjunto Q se encontre vazio, ou seja, não existam mais vértices a serem visitados. Ao final, basta adicionar a conexão do último vértice até a origem, fechando um ciclo hamiltoniano.

Algoritmo 1: Algoritmo do Vizinho Mais Próximo

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

1 $u \leftarrow 0$;
2 $C \leftarrow \emptyset$;
3 $Q \leftarrow V - \{u\}$;
4 **enquanto** $Q \neq \emptyset$ **faça**
5 $\quad v \leftarrow$ adjacente não visitado de menor distância a u;
6 $\quad C \leftarrow C \cup \{v\}$;
7 $\quad Q \leftarrow Q - \{v\}$;
8 $\quad u \leftarrow v$;
9 $C \leftarrow C \cup \{C[0]\}$;
10 **retorna** C;

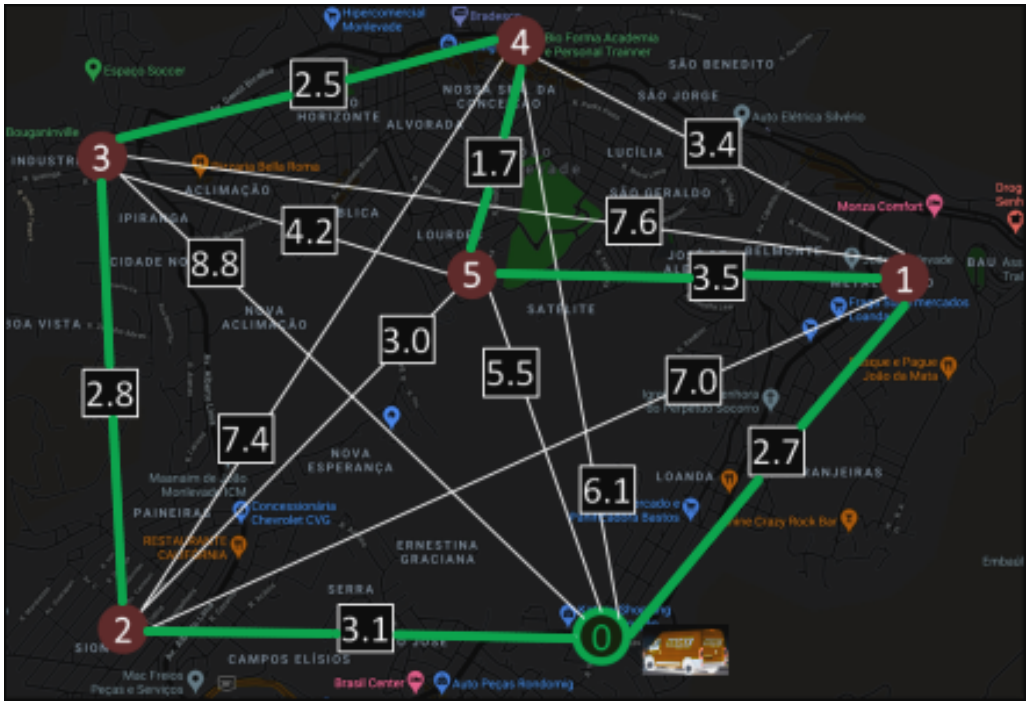

Figura 2. Rota feita pelo Algoritmo de Força Bruta, na região de João Monlevade, MG, para exemplificar o transporte de Correios

2.2. Algoritmo Força Bruta

Essa abordagem de Algoritmo para a solução do PCV garante sempre encontrar a solução ótima, isto é, aquela que apresenta o menor custo, independentemente do tamanho do grafo analisado. Seu funcionamento consiste em analisar todas as permutações possíveis de rotas no grafo, buscando a rota de menor custo. Dada sua natureza combinatória, este algoritmo apresenta uma complexidade computacional da ordem de $O(n!)$, para um grafo completo K^n.

Na Figura 2 [FONSECA 2020], visualiza-se o rastreio para o mesmo problema apresentado na seção 2.1, porém dessa vez utilizando o algoritmo de força bruta. Como esperado, o algoritmo encontrou a solução ótima para o problema, que consiste na rota 0-1-5-4-3-2-0, que apresenta custo 16.3.

Contudo, a desvantagem deste algoritmo está interligada diretamente com sua eficiência para encontrar a melhor rota. Essa afirmação é válida, pois, como explicado anteriormente, o Algoritmo de Força Bruta analisa todas as permutações possíveis de rotas em um grafo, apresentando custo computacional fatorial. Este fato torna o seu tempo de execução proibitivo, para quase todos os grafos. Um exemplo evidente desse quesito pode ser visto na Figura 2 [FONSECA 2020], em que há um grafo completo pequeno K^6, sobre o qual a execução deste algoritmo resultaria na análise de $6! = 720$ rotas diferentes a serem analisadas. Logo, é possível concluir que o tempo de execução só tende a crescer de forma a se tornar proibitivo à medida em que o número de vértices aumenta.

Por fim, faz-se necessário delinear o pseudocódigo do Algoritmo Força Bruta com o objetivo de auxiliar na compreensão do conteúdo a ser abordado nas demais seções. Este pseudocódigo pode ser encontrado no Algoritmo 2 [FONSECA 2020]. As linhas 1-2

consistem em inicializar as variáveis necessárias para a execução correta do algoritmo, e as linhas 3-8 são responsáveis por fazer a comparação entre cada permutação de vértices possível do grafo G, buscando pelo ciclo de menor custo C^{best}. Ao final das comparações, o melhor ciclo encontrado é retornado, e a execução é finalizada.

Algoritmo 2: Algoritmo de Força Bruta

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

```
1 cost^min ← ∞;
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;
```

3. Experimentação e Resultados

Nesta seção, será feita uma comparação entre os tempos de execução de cada um dos algoritmos, para diversos tipos de grafos. Além disso, será definida a metodologia utilizada para a execução dos testes.

3.1. Metodologia

Os grafos utilizados nos testes são grafos ponderados, completos e não-orientados, cujas arestas contém pesos gerados aleatoriamente, por um algoritmo que recebe como parâmetros o número de vértices e os pesos mínimo e máximo, que representam o intervalo no qual o peso de uma aresta qualquer do grafo deve figurar. O computador utilizado para os testes possui um processador Intel Core i5-4460, com quatro núcleos, operando a 3.20GHz cada, e 8GB de memória RAM.

Para facilitar o entendimento dos resultados, é apresentada na tabela 1, uma associação de uma sigla para cada uma das configurações de grafos, que serão utilizadas posteriormente nos resultados.

Os resultados de tempo de execução serão representados em segundos, com uma precisão de seis casas decimais. Além disso, utilizarão-se siglas para se referenciar aos algoritmos na tabela, sendo VMP (Vizinho Mais Próximo) e FB (Força Bruta). Caso um algoritmo tenha passados dos 10 minutos de processamento sem retornar seu resultado, atribui-se o valor E.T (Extrapolou o tempo).

3.2. Resultados

Na Tabela 2 são mostrados os resultados de tempos de execução dos algoritmos, utilizando-se dos grafos mostrados na Tabela 1.

A primeira informação que se mostra evidente após a análise da tabela, é que o algoritmo VMP manteve tempos de execução extremamente rápidos, mesmo para grafos

Tabela 1. Tabela explicativa para parâmetros utilizados nos testes.

Sigla	Vértices	Peso mínimo	Peso máximo
P1	5	1	100
P2	5	1	10
P3	10	1	10
P4	10	1	100
P5	50	1	10
P6	50	1	100
P7	100	1	10
P8	100	1	100

Tabela 2. Resultados de tempo de execução.

Configuração	VMP	FB
P1	0.006637	0.006845
P2	0.008138	0.005042
P3	0.008666	5.095603
P4	0.011424	5.294198
P5	0.006152	E.T
P6	0.006287	E.T
P7	0.010192	E.T
P8	0.010420	E.T

de tamanho maior. O tempo não ultrapassou um segundo em nenhum dos testes executados. Isto mostra que o algoritmo VMP pode ser, de fato, considerado como uma opção válida para aplicações do PCV, desde que mantenha-se a consciência de que este algoritmo não garante a melhor solução e, inclusive, raramente a encontra.

Além disso, verifica-se que o algoritmo FB somente apresentou tempos de execução competitivos ao VMP em grafos extremamente pequenos. Perceba que, ao aumentar o tamanho do grafo, seu tempo de execução se mostrou muito maior que os tempos do VMP, tendo inclusive extrapolado o tempo máximo de 10 minutos de execução, para os grafos com 50 e 100 vértices. Isto se dá devido à sua complexidade computacional da ordem de $O(n!)$, que tende a ser proibitiva para grafos com muitos vértices. Este fato torna o algoritmo FB inviável para a grande maioria das aplicações do PCV, uma vez que não conseguirá executar em tempo viável, por mais que seja capaz de garantir sempre encontrar a solução ótima.

Na comparação entre os dois resultados de tempos de execução, verifica-se com clareza que o algoritmo VMP superou o FB neste quesito com dominância, uma vez que seus resultados jamais ultrapassaram um segundo de tempo máximo de execução, enquanto o FB passou do tempo máximo estipulado diversas vezes. Porém, existe o fato de que, em todas as execuções nas quais o FB finalizou, a solução ótima foi encontrada, enquanto o VMP raramente encontraria esta solução. Em suma, evidenciam-se os pontos positivos e negativos de cada algoritmo: o VMP, que executa em um tempo baixíssimo mas não encontra a melhor solução, ou o FB, que executa em um tempo muito maior, porém garante encontrar a melhor solução. Estes fatores devem ser levados

em consideração no momento da escolha entre os algoritmos, e também deve ser analisada a tendência de que o tempo de execução do FB se torne proibitivo à medida que o tamanho do grafo aumenta. Logo, para grafos grandes, a melhor opção dentre os dois seria o VMP, e para grafos muito pequenos, poderia ser considerado o uso do FB.

4. Conclusão

Destarte, diante de todo o estudo exposto sobre o Problema do Caxeiro Viajante, é possível concluir que esse é de suma importância não apenas para a solução de questões empresarias de logistíca e planejamento, mas também para inúmeras áreas, como a biologia, física e matemática. Além disso, ao levar em consideração os pontos positivos e negativos, bem como os resultados dos algoritmos em questão, vulgo o Vizinho mais Próximo e o Força Bruta, é concebível assegurar que não existe o melhor algoritmo que resolva todos os problemas. Em contrapartida, é possível determinar o melhor algoritmo para cada tipo de problema; ou seja, é necessário, primeiramente que haja uma análise crítica do problema, avaliando principalmente o tamanho do grafo e seu objetivo, a fim de que, posteriormente, seja possível definir qual dos algoritmos deve ser escolhido para alcançar maior eficiência e desempenho.

Embora técnicas já tenham sido fortemente testadas e aprimoradas ao longo dos anos, é extremamente necessário que o PCV continue em evolução, uma vez que, quando uma tecnologia é descoberta ou aperfeiçoada, as mais variadas áreas da sociedade são beneficiadas. Dessa forma, é plausível que investimentos continuem sendo feitos para que este problema seja continuamente abordado em pesquisas científicas, similarmente ao que foi reproduzido neste artigo. Dessa forma, aplicações do PCV poderão obter melhores soluções, tornando-se mais viáveis e eficientes e, portanto, contribuindo para os avanços tecnológicos do mundo moderno.

Referências

CUNHA, M. H. S. and ALMEIDA, N. P. d. (2020). Algoritmos para o problema do caixeiro viajante. `https://github.com/Marcoshsc/TravelingSalesmanProblem`.

FONSECA, G. H. G. (2020). A07: Percursos abrangentes.

GOLDBARG, M. C. and LUNA, H. P. L. (2000). *Otimização combinatória e programação linear: Modelos e Algoritmos*. Campus.

REINELT, G. (1994). *The Traveling Salesman: Computational Solutions for TSP Applications*. Springer.