**A Influência dos Algoritmos de Dijkstra e A\* em Aplicativos de Navegação**

## Jonathan Barbosa da Silveira1, Sérgio Assunção monteiro1

Centro Universitário Carioca (UNICARIOCA), Rio de Janeiro, RJ, Brasil

jonathanbdasilveira@gmail.com, smonteiro@unicarioca.edu.br

***Abstract.*** *This article addresses the influence of the Dijkstra and A\* algorithms on the navigation application revolution, highlighting the essential role played by the A\* Algorithm in the efficiency of real-time navigation applications, such as Uber, Waze and Google Maps, for example. Dijkstra's algorithm, developed by Edsger W. Dijkstra, provides the basis for calculating efficient routes in weighted graphs, but may be inefficient for large-scale applications that require real-time information updates. The A\* Algorithm, an evolution of the Dijkstra Algorithm, uses heuristics to prioritize promising paths, making it ideal for navigation applications.*

***Resumo.*** *Este artigo aborda a influência dos algoritmos de Dijkstra e A\* na revolução dos aplicativos de navegação, com destaque para o papel essencial desempenhado pelo Algoritmo A\* na eficiência dos aplicativos de navegação em tempo real, como Uber, Waze e Google Maps por exemplo. O algoritmo de Dijkstra, desenvolvido por Edsger W. Dijkstra, fornece a base para o cálculo de rotas eficientes em grafos ponderados, mas pode ser ineficiente para aplicativos de grande escala que necessitam de informações atualizas em tempo real. O Algoritmo A\*, uma evolução do Algoritmo de Dijkstra, utiliza heurísticas para priorizar caminhos promissores, tornando-o ideal para aplicativos de navegação.*

***Palavras chave*** *- algoritmo de Dijkstra, algoritmo A\*, caminho mais curto, rotas*

1. **Introdução**

A tecnologia de navegação desempenha um papel fundamental em nossas vidas cotidianas e transformou a forma como nos deslocamos no mundo. Desde a popularização dos smartphones e a surgimento de aplicativos de navegação, como o Waze, Google Maps e muitos outros, tornou-se mais fácil do que nunca encontrar rotas eficientes para chegar ao nosso destino. Esses aplicativos não apenas nos fornecem direções detalhadas, mas também levam em consideração uma série de variáveis, como o tráfego em tempo real, atualizações de condições de estrada e preferências individuais, para criar rotas personalizadas e eficientes.

No entanto, por trás da aparente simplicidade de inserir um destino em um aplicativo e seguir suas instruções de rota, existe uma complexidade algorítmica que é fundamental para tornar essa experiência possível. Dois dos algoritmos mais influentes e amplamente utilizados nesse domínio são o Algoritmo de Dijkstra e o Algoritmo A\*. Eles formam a papel central de como os aplicativos de navegação determinam as rotas mais eficientes para os usuários.

Neste artigo, exploraremos a evolução desses algoritmos, começando com o Algoritmo de Dijkstra, e avançando para o Algoritmo A\*, uma evolução notável que prioriza a eficiência em cenários de tempo real. Vamos examinar como o Algoritmo A\* se tornou uma pedra angular na tecnologia de navegação, sendo amplamente aplicado em aplicativos populares, com destaque para o Waze.

1. **A Necessidade de Algoritmos de Roteamento**

Antes de mergulharmos nas complexidades dos algoritmos de Dijkstra e A\*, é importante compreender a razão pela qual esses algoritmos desempenham um papel crucial nos aplicativos de navegação. Quando um usuário insere um destino em seu aplicativo de navegação, o software precisa calcular a rota mais eficiente para que o usuário possa seguir. Isso envolve uma análise abrangente de diversos fatores, incluindo a distância entre os pontos, as condições de tráfego em tempo real, a presença de obstáculos nas estradas e quaisquer restrições de velocidade que possam estar em vigor. Os algoritmos de Dijkstra e A\* são essenciais porque são projetados para encontrar soluções eficazes para esse problema, considerando todas essas variáveis e fornecendo ao usuário uma rota otimizada que economiza tempo, minimiza o congestionamento e melhora a experiência de viagem de forma geral.

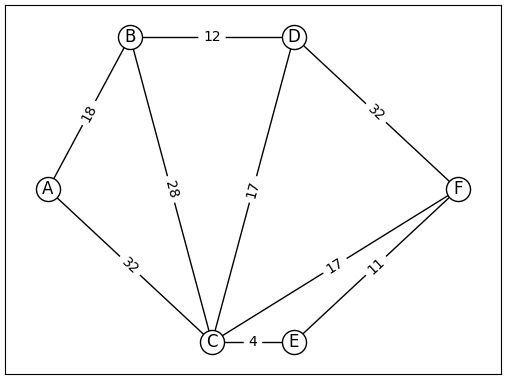
1. **O Algoritmo de Dijkstra**

O Algoritmo de Dijkstra, desenvolvido pelo cientista da computação holandês Edsger W. Dijkstra em 1956 e publicado em 1959, é um algoritmo de busca do menor caminho entre dois nós em grafos ponderados. Ele foi originalmente concebido para otimizar redes de telecomunicações, mas sua aplicação se estendeu muito além disso. O Algoritmo de Dijkstra encontra o caminho mais curto entre dois pontos em um grafo, levando em consideração o peso das arestas.

O Algoritmo de Dijkstra funciona explorando todos os caminhos possíveis a partir do ponto de origem até encontrar o caminho mais curto para o ponto de destino. Ele mantém uma lista de todos os pontos visitados e, a cada passo, escolhe o próximo ponto mais próximo. Esse processo é repetido até que o ponto de destino seja alcançado.

* 1. **Aplicação do Algoritmo de Dijkstra**

Um grafo G é definido como um par ordenado G = (V, A), no qual V é um conjunto não vazio de vértices, e A é um conjunto de pares de vértices chamados de arestas, que representam conexões entre os vértices. As arestas podem ser atribuídas com pesos específicos, criando assim um grafo ponderado, onde o peso denota o custo associado à travessia da aresta. Isso pode ser ilustrado ao representar uma rede de estradas que conecta várias cidades através de um grafo ponderado, onde os pesos das arestas são todos positivos.

No exemplo seguinte, será apresentado como o Algoritmo de Dijkstra funciona, utilizaremos uma tabela e um grafo para ilustrar como o caminho mais curto entre dois pontos pode ser definido através desse algoritmo. Na Figura 1, temos um grafo que descreve um conjunto de cidades V = {A, B, C, D, E, F}, com suas estradas correspondentes e as distâncias que as conectam.

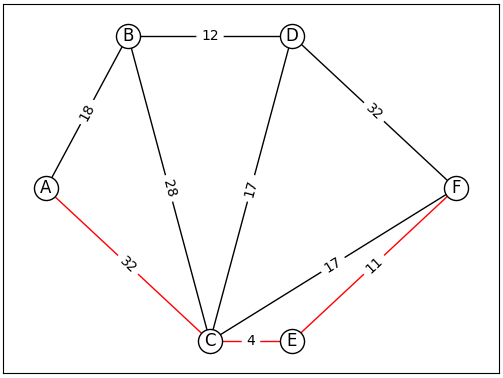
**Figura 1. Grafo ponderado com os valores das arestas e vértices**

Este algoritmo parte de uma estimativa inicial para a distância mínima, que é representada como infinito (∞). Conforme o algoritmo avança, essa estimativa é continuamente redefinida. Ele assume que uma cidade é "fechada" quando um caminho de distância mínima da cidade de origem da busca até aquela cidade é encontrado.

Os dados desse grafo também podem ser lidos através de uma tabela, onde são armazenadas as informações das cidades e as distâncias entre elas. Na Figura 2 esses dados podem ser conferidos.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** | **E** | **F** |
| **A** | 0 | 18 | 32 | 0 | 0 | 0 |
| **B** | 18 | 0 | 28 | 12 | 0 | 0 |
| **C** | 32 | 28 | 0 | 17 | 4 | 17 |
| **D** | 0 | 12 | 17 | 0 | 0 | 32 |
| **E** | 0 | 0 | 4 | 0 | 0 | 11 |
| **F** | 0 | 0 | 17 | 32 | 11 | 0 |

**Figura 2. Tabela com os valores das arestas e vértices**

Inicialmente, todos os vértices são atribuídos com uma distância infinita (nesse caso representados por zero), com exceção da cidade de origem escolhida, que naturalmente tem uma distância de zero com relação a si mesmo. Ao escolher uma cidade de Origem e outra de Destino, o algoritmo vai passar a interpretar continuamente à medida que vai passando entre as estradas e cidades. Vamos usar como exemplo a cidade de Origem A e cidade de Destino F, na Figura 3 será mostrado o resultado do menor caminho encontrado pelo algoritmo segundo a Origem e Destino selecionados.

**Figura 3. Grafo ponderado com o caminho mais curto entre Origem e Destino**

O algoritmo considera o ponto de origem e, com base nos valores associados a todas as estradas até a cidade F, determina a próxima cidade a ser visitada, priorizando as distâncias mais curtas. Isso resulta na seleção do caminho mais curto entre a cidade A e a cidade F, que no exemplo fornecido é A -> C -> E -> F.

É fundamental observar que o Algoritmo de Dijkstra pode ser empregado exclusivamente em grafos ponderados, nos quais os pesos das arestas são estritamente positivos. Sua finalidade é calcular as distâncias entre uma cidade específica e todas as outras cidades no grafo. Isso contrasta com o Algoritmo de Floyd, que tem a capacidade de calcular as distâncias entre todas as cidades no grafo, independentemente dos pesos das arestas.

Durante o desenvolvimento do projeto montei em Python um exemplo de como poderíamos aplicar desse algoritmo utilizando uma interface para receber Origem e Destino e traçar a menor rota. Disponível em: [https://github.com/BonathanRJ/TCC---Programa-de-Rotas/blob/main/project\_python\_v1/dijkstra\_path\_graph.py](https://github.com/BonathanRJ/TCC---Programa-de-Rotas/blob/main/project_python_v1/dijkstra_path_graph.py%20%20%20).

1. **O Algoritmo A\***

O Algoritmo A\* (pronunciado "A-star") é uma evolução do Algoritmo de Dijkstra que visa superar suas limitações em termos de eficiência. Ele foi proposto por Peter Hart, Nils Nilsson e Bertram Raphael em 1968. O Algoritmo A\* também encontra o caminho mais curto entre dois pontos em um grafo, mas utiliza uma heurística para priorizar os caminhos mais promissores.

A heurística no Algoritmo A\* fornece uma estimativa do custo do caminho restante até o ponto de destino, permitindo que o algoritmo priorize a exploração de caminhos mais promissores em termos de eficiência. Isso torna o A\* consideravelmente mais rápido que o Algoritmo de Dijkstra em muitos cenários, sendo uma escolha ideal para aplicações que exigem busca de caminhos eficientes, como sistemas de navegação em tempo real, onde a estimativa heurística melhora significativamente o desempenho da busca de caminho.

1. **Heurística e Eficiência do Algoritmo A\***

Um aspecto fundamental do Algoritmo A\* é a heurística utilizada para avaliar a eficiência dos caminhos. A heurística é uma função que fornece uma estimativa do custo para alcançar o destino a partir de um determinado ponto. Essa estimativa ajuda o algoritmo a priorizar os caminhos que parecem ser mais promissores, economizando tempo e recursos computacionais.

Um exemplo prático da heurística em um aplicativo de navegação seria considerar a distância em linha reta até o destino como uma estimativa do custo restante. Embora essa estimativa possa não ser 100% precisa, ela é geralmente boa o suficiente para tornar o Algoritmo A\* muito eficiente em comparação com o Algoritmo de Dijkstra. A capacidade de tomar decisões inteligentes e encontrar rotas eficientes rapidamente é uma das razões pelas quais o Algoritmo A\* é amplamente utilizado em aplicativos de navegação, incluindo o Waze.

1. **A Evolução dos Algoritmos em Aplicativos de Navegação**

Compreender a evolução dos algoritmos de roteamento é fundamental para entender como os aplicativos de navegação modernos, como o Waze, funcionam. O Waze, lançado em 2008, representou um marco na evolução dos aplicativos de navegação. Ele se destacou ao incorporar a inteligência coletiva de sua base de usuários para fornecer informações em tempo real sobre o tráfego e condições da estrada. No entanto, a base do Waze é a aplicação eficiente de algoritmos de roteamento, como o Algoritmo A\*.

1. **A Influência do Algoritmo A\* no Waze**

O Algoritmo A\* desempenha um papel crucial no Waze, permitindo que o aplicativo calcule rotas eficientes e dinâmicas com base nas condições em tempo real. Vamos explorar como o Waze utiliza o Algoritmo A\* em sua operação.

* 1. **Cálculo de Rotas Inicial**

Quando um usuário insere seu destino no Waze, o aplicativo utiliza o Algoritmo A\* para calcular uma rota inicial. Ele leva em consideração a distância, o tipo de estrada, as restrições de velocidade e outras informações relevantes. A heurística ajuda o Waze a priorizar caminhos mais promissores, economizando tempo e recursos computacionais, estimando tempo e o custo de cada caminho e, com base nessas estimativas, selecionará o caminho mais promissor. Esse processo é repetido até que a rota mais eficiente seja encontrada.

* 1. **Atualizações em Tempo Real**

O Waze se destaca na coleta de dados em tempo real de seus usuários. Os motoristas que utilizam o aplicativo contribuem ativamente com informações sobre o tráfego, acidentes, bloqueios de estradas e outros eventos relevantes, tornando-se uma rede de informações em constante atualização. Esses dados permitem que o Waze ajuste as rotas à medida que as condições mudam instantaneamente. O Algoritmo A\* é reexecutado regularmente para determinar a rota mais eficiente com base nas informações mais recentes, garantindo que os usuários recebam orientações precisas e atualizadas para evitar atrasos e percorrer seus trajetos de forma mais eficaz. Por exemplo, se uma estrada está congestionada, o aplicativo pode propor rotas alternativas, avaliando rapidamente os custos de tempo em diferentes caminhos com base nas informações em tempo real. Isso é essencial para evitar o tráfego e minimizar o tempo de viagem do usuário.

* 1. **O futuro dos aplicativos de rota**

Nos últimos anos, temos visto avanços significativos na integração de inteligência artificial e aprendizado de máquina nos aplicativos de navegação. Algoritmos de aprendizado profundo são utilizados para prever com maior precisão as condições de tráfego, enquanto sistemas de recomendação personalizados sugerem paradas ao longo do caminho com base nas preferências individuais dos usuários. Esses avanços representam uma evolução contínua na busca por tornar a navegação mais eficiente, personalizada e agradável.

1. **Conclusão**

Os algoritmos de Dijkstra e A\* desempenham o papel central nos aplicativos de navegação, permitindo o cálculo de rotas eficientes com base em fatores como distância, tráfego e condições das estradas. Com a evolução desses aplicativos e a coleta de dados em tempo real, eles se tornaram essenciais para a vida cotidiana, simplificando a navegação e economizando tempo.

À medida que a tecnologia de navegação avança, os algoritmos como o Algoritmo A\* e sua aplicação inteligente em aplicativos como o Waze continuam a impulsionar melhorias, tornando as viagens mais eficientes.

**Referências**

R. Kalaichelvi and S. Prasanna Devi. "A Comparative Study on Dijkstra's Algorithm and A\* Algorithm for Shortest Path Problem." International Journal of Computer Applications 142, no. 4 (2016): 18-22.

P. Jeyanthi and P. Sakthivel. "Comparative Study of Dijkstra's Algorithm and A\* Algorithm for Path Finding." International Journal of Advanced Research in Computer Science and Software Engineering 4, no. 7 (2014): 996-1001.

N. Bhaskar and K. S. Ravi Chandran. "Performance Analysis of Dijkstra's Algorithm and A\* Algorithm in Graph Theory." International Journal of Emerging Trends & Technology in Computer Science 2, no. 2 (2013): 126-132.

S. S. Patil and S. D. Patil. "Comparative Study of Dijkstra's Algorithm and A\* Algorithm." International Journal of Engineering Research & Technology 2, no. 2 (2013): 1-6.

M. Elshafei and A. Elnagar. "A Comparative Study between Dijkstra's Algorithm and A\* Algorithm in Finding Shortest Path." International Journal of Computer Science and Mobile Computing 3, no. 5 (2014): 322-328.