GRUPO DE ESTUDO DE EDUCAÇÃO A DISTÂNCIA / CETEC
TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS - MODALIDADE EaD

Gustavo Furtado Ferreira Jardim

ALGORITMO RÁPIDO DE GPS PARA ENCONTRAR ROTAS

São Paulo, Mogi Das Cruzes

Gustavo Furtado Ferreira Jardim

Trabalho de Conclusão de Curso apresentado ao Curso Técnico em Desenvolvimento de Sistemas - modalidade EaD, orientado pelo Prof. Henderson Tavares de Souza, como requisito parcial para obtenção do título de Técnico em Desenvolvimento de Sistemas.

2025

RESUMO

Este trabalho tem como objetivo apresentar o desenvolvimento e a análise de um algoritmo eficiente para cálculo de rotas em sistemas de GPS. Diante do crescimento das demandas por navegação em tempo real, é fundamental que os algoritmos sejam capazes de processar grandes volumes de dados geoespaciais de forma rápida e precisa. A pesquisa aborda técnicas de otimização aplicadas aos algoritmos de roteamento, com foco na melhoria do tempo de resposta e na redução do uso de recursos computacionais. Entre os métodos estudos estão algoritmos clássicos, como Dijkstra e A*, além de aprimoramentos como pré-processamento com grafos hierárquicos (Hierarchical Routing), atalhos (Contraction Hierarchies) e algoritmos baseados em heurísticas. O trabalho visa demonstrar como a escolha e a adaptação desses algoritmos impactam diretamente na eficiência e na qualidade das rotas geradas, contribuindo para o desenvolvimento de soluções mais rápidas e eficazes em sistemas de navegação GPS.

1. INTRODUÇÃO

Com o avanço da tecnologia e o crescimento das cidades, os sistemas de navegação por GPS tornaram-se indispensáveis para milhões de pessoas em todo o mundo. Seja no transporte urbano, na logística de empresas ou na mobilidade individual, a capacidade de encontrar rotas eficientes e rápidas é essencial para otimizar tempo, reduzir custos e melhorar a experiência dos usuários.

Por trás desses sistemas existem algoritmos complexos que precisam processar grandes volumes de dados geográficos em frações de segundos. Desafios como tráfego em tempo real, alterações nas vias, construções e condições adversas tornam ainda mais necessária a utilização de soluções computacionais rápidas e precisas.

Tradicionalmente, algoritmos como Dijkstra e A* são amplamente utilizados para encontrar o menor caminho em grafos, que representam mapas de ruas e estradas. No entanto, à medida que o tamanho dos mapas cresce, a aplicação direta desses algoritmos se torna ineficiente em termos de tempo de processamento, especialmente em dispositivos móveis com recursos limitados.

Diante desse cenário, este trabalho tem como objetivo estudar, implementar e analisar soluções otimizadas para o problema de cálculo de rotas em sistemas de GPS. Serão exploradas técnicas modernas, como hierarquização de grafos, algoritmos com pré-processamento de dados, heurísticas eficientes e métodos de busca adaptativos, que possibilitam reduzir significativamente o tempo de resposta sem comprometer a precisão das rotas.

Este estudo busca não apenas compreender o funcionamento dos principais algoritmos de roteamento, mas também propor melhorias e adaptações que possam ser aplicadas em contextos práticos, contribuindo para o desenvolvimento de sistemas de navegação mais rápidos, inteligentes e acessíveis.

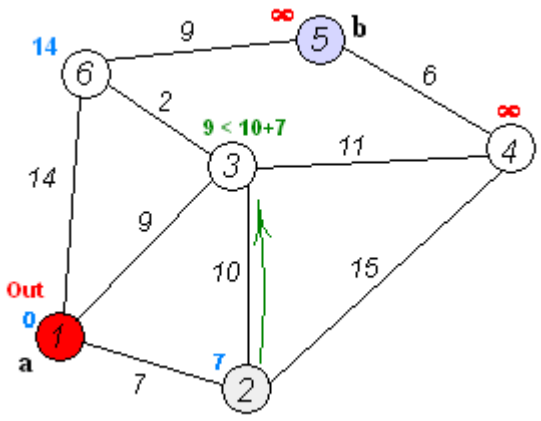
2. ALGORITMO DE DIJKSTRA

O algoritmo de Dijkstra, desenvolvido por Edsger W. Dijkstra em 1956 e publicado em 1959, é um dos algoritmos mais conhecidos e utilizados na área de ciência da computação para a resolução do problema de caminho mínimo em grafos. Ele é amplamente aplicado em sistemas de navegação (como é o nosso caso), redes de computadores e diversas outras áreas que demandam a busca pela rota mais curta entre dois pontos.

O algoritmo de Dijkstra é baseado em grafos ponderados, nos quais cada aresta possui um custo ou peso, representando por exemplo, a distância, o tempo ou outro critério relevante. Seu funcionamento consiste em explorar os nós (ou vértices) do grafo, partindo de um nó de origem, e gradualmente expandir os caminhos possíveis, sempre escolhendo o caminho de menor custo acumulado até alcançar o destino ou calcular os menores caminhos para todos os nós.

2.1 Funcionamento do algoritmo de Dijkstra

Atribui-se a todos os nós um valor de distância infinita, exceto para o nó de origem que recebe distância zero, a seguir cria-se um conjunto de nós não visitados, seleciona-se o nó com a menor distância acumulada entre os não visitados, Para cada nó vizinho do nó atual, calcula-se a soma da distância do nó atual mais o custo da aresta até o vizinho, se essa soma for menor que a distância atual registrada para o vizinho, atualiza-se o valor. Após processar todos os vizinhos, o nó atual é marcado como visitado e removido do conjunto de não visitados. Repete-se o processo até que o nó de destino seja visitado (se for uma busca por caminho entre dois pontos) ou até que todos os nós tenham sido processados (se for encontrar os menores caminhos a partir do nó de origem para todos os nós).



3. ALGORITMO A*

O algoritmo A* (ou A estrela) é uma extensão do algoritmo de Dijkstra, desenvolvido por Peter Hart, Nils Nilsson e Bertram Raphael em 1968. A principal característica que diferencia o A* dos algoritmos tradicionais de busca de caminho mínimo é o uso de uma **função heurística,** que permite guiar a busca de forma mais eficiente em direção ao objetivo.

Por ser mais rápido e eficiente em muitos casos, o A* é amplamente utilizado em sistemas de navegação de GPS, jogos, inteligência artificial e robótica, onde encontrar o caminho mais curto de maneira rápida é essencial.

3.1 Funcionamento do algoritmo A*

O algoritmo A* combina duas informações principais durante a busca: O custo real do caminho até o nó atual (g(n)), que é o custo acumulado desde o ponto de origem até aquele nó e uma estimativa do custo do nó atual até o destino (h(n)), chamada de função heurística.

A decisão sobre qual nó expandir é feita através da função:

$$f(n)=g(n)+h(n)$$

Onde f(n) é o custo total estimado do caminho passando pelo nó n, g(n) é o custo real do caminho desde o início até o nó n, e h(n) é a estimativa do custo restante até o destino.

3.2 Passos do algoritmo

O funcionamento básico do A* começa pela inicialização onde começa no nó de origem, com g(n) = 0 e calcula h(n) para estimar f(n), todos os outros nós têm f(n) inicial infinito. A seguir seleciona o nó da fronteira com o menor f(n) para expandir, para cada vizinho do nó atual nós calculamos o g(n) que é g(atual) + custo da estrela, o h(n) com a heurística, e f(n) = g(n) + h(n), se o vizinho não foi visitado ou encontrou-se um g(n) menor, atualiza o caminho.

3.3 A Heurística

A eficiência do A* depende fortemente da função heurística h(n). A heurística deve ser admissível: nunca superestima o custo até o destino (isso garante que o A* encontre sempre o caminho mais curto) e consistente: para cada nó n e vizinho n’, o custo de ir de n até n’ somado a h(n’) deve ser maior ou igual a h(n).

A Distância Euclidiana é usada quando o movimento é livre em linha(reta)

$$h(n)= \sqrt{(xdestino-xn)2+(ydestino-yn)2}$$

A Distância Manhattan é usada quando só se pode mover em direções ortogonais(norte, sul, leste, oeste)

$$h(n)=|xdestino-xn|+|ydestino-yn|$$

A Distância Geodésica é usada em mapas com coordenadas geográficaas (latitude e longitude)

4. CONSIDERAÇÕES FINAIS

Ao longo deste trabalho, foram apresentados dois dos algoritmos mais utilizados na resolução do problema de caminho mínimo em grafos: Dijkstra e A*. Ambos desempenham papéis fundamentais em sistemas de roteamento, como GPS, redes de comunicação e inteligência artificial. No entanto, cada um apresenta características específicas, com vantagens e desvantagens que devem ser consideradas conforme o contexto de aplicação.

4.1 Vantagens e Desvantagens do Algoritmo de Dijkstra

Ele é simples, direto e bem compreendido, também é fácil de implementar e depurar, garante o menor caminho, calcula todos os caminhos mínimos, porém explora muitos nós sem priorizar o destino, tornando-o menos eficiente em grafos grandes quando o interesse é apenas um destino específico, em mapas extensos, pode ser lento e consumir muita memória, e também não possui mecanismos de otimização baseados em direção ou proximidade do destino.

4.2 Vantagens e Desvantagens do Algoritmo A*

Utiliza heurística para priorizar os nós que estão mais próximos do destino, reduzindo significativamente o número de nós visitados, quando equipada com uma boa heurística pode ser mais eficiente que Dijkstra, encontra o caminho mais curto com menos esforço computacional, permite personalização da heurística para diferentes critérios, como menor distância, menor tempo ou menor custo, altamente eficiente quando o objetivo é encontrar o caminho entre um ponto de origem e um ponto de destino específicos. Porém por ser dependente da heurística, uma heurística mal projetada pode fazer o A* ter desempenho tão ruim quanto Dijkstra, ou até pior, também não é adequado para múltiplos destinos, o A* é mais indicado para buscas ponto-a-ponto. Se for necessário calcular caminhos de um ponto para todos, Dijkstra ou outras técnicas são mais adequadas, outro ponto negativo é que em grafos muito grandes, o A* pode consumir bastante memória, principalmente se muitos caminhos forem avaliados, e também não lida diretamente com dados dinâmicos, assim como Dijkstra, precisa de adaptações para cenários dinâmicos como trânsito em tempo real.

REFERÊNCIAS

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Algoritmos: Teoria e Prática.** 3. ed. Rio de Janeiro: Elsevier, 2013.

DIJKSTRA, E. W. **A Note on Two Problems in Connexion with Graphs.** Numerische Mathematik, v. 1, p. 269–271, 1959. Disponível em: https://doi.org/10.1007/BF01386390

HART, P. E.; NILSSON, N. J.; RAPHAEL, B. **A Formal Basis for the Heuristic Determination of Minimum Cost Paths.** IEEE Transactions on Systems Science and Cybernetics, v. 4, n. 2, p. 100–107, 1968. Disponível em: https://doi.org/10.1109/TSSC.1968.300136

SEDGEWICK, R.; WAYNE, K. **Algorithms.** 4. ed. Boston: Addison-Wesley, 2011.

RUSSELL, S.; NORVIG, P. **Inteligência Artificial.** 3. ed. Rio de Janeiro: Elsevier, 2013.

LAKSHMANAN, L. V. S.; JIANG, C. **Data-Driven Algorithms for Traffic and Route Optimization.** ACM Computing Surveys (CSUR), v. 52, n. 4, p. 1-36, 2019. Disponível em: https://doi.org/10.1145/3329123

WIKIPEDIA. **A-star algorithm.** Disponível em: https://en.wikipedia.org/wiki/A*_search_algorithm. Acesso em: 24 mai. 2025.

WIKIPEDIA. **Dijkstra's algorithm.** Disponível em: https://en.wikipedia.org/wiki/Dijkstra's_algorithm. Acesso em: 24 mai. 2025.