Instituto de Ciências Matemáticas e de Computação
(ICMC)

SCC0530 - Inteligência Artificial

Algoritmos de busca

Lucas Bichara - Nº USP: 11296738
Rafael Jun Teramae Dantas - Nº USP: 12563686
Leonardo Gonçalves Chahud- Nº USP: 5266649

Sumário

Introdução

A busca por informações relevantes e eficientes em redes geográficas tem se tornado cada vez mais essencial em diversas áreas, como logística, transporte e planejamento urbano. A capacidade de encontrar o caminho mais curto ou otimizado entre dois pontos em uma rede geográfica é um problema complexo, mas pode ser solucionado por meio de algoritmos de busca. Neste trabalho, exploraremos quatro tipos de algoritmos amplamente utilizados nesse contexto: busca em *profundidade*, busca em *largura*, algoritmo *Best-First* e algoritmo *A** e para tal usaremos uma rede geográfica.

Uma rede geográfica é uma estrutura que representa a conectividade entre locais em um espaço geográfico. Essa estrutura é composta por nós, que representam os locais, e arestas, que representam as conexões entre esses locais. A análise de redes geográficas é importante em diversas áreas, como logística, transporte e planejamento urbano, pois permite entender a conectividade entre diferentes locais e identificar caminhos mais eficientes para alcançar um determinado objetivo. A modelagem de redes geográficas pode ser usada para prever o fluxo de tráfego ou energia em diferentes pontos da rede, o que é útil para otimizar o uso dos recursos disponíveis.

Um dos algoritmos mais comuns é o de busca em profundidade, que consiste em explorar um ramo da rede até encontrar um nó objetivo ou até que não haja mais nós a serem explorados. Este algoritmo é útil quando se busca uma solução mais rápida, mas pode não ser eficiente em redes muito grandes ou complexas.

Já o algoritmo de busca em largura explora todos os nós vizinhos antes de avançar para o próximo nível da rede. Isso garante que a solução encontrada seja a mais próxima possível do nó objetivo, mas pode ser mais lento do que a busca em profundidade.

O algoritmo Best-first é uma variação da busca em largura, que utiliza uma heurística para determinar qual caminho seguir. Isso permite que o algoritmo encontre uma solução mais rapidamente do que a busca em largura convencional.

Por fim, o algoritmo A* é uma combinação da busca em largura e da busca Best-first. Ele utiliza uma função de avaliação para determinar qual caminho seguir, levando em consideração tanto a distância até o nó objetivo quanto o custo acumulado até o momento. Isso permite que o algoritmo encontre a solução mais rápida possível, mesmo em redes grandes e complexas.

Em resumo, os algoritmos de busca em redes são ferramentas fundamentais para a análise de redes geográficas. Cada um deles possui suas vantagens e desvantagens, e a escolha do melhor algoritmo dependerá do contexto específico em que ele será aplicado.

Redes Geográficas

Redes geográficas são um tipo de rede que modela sistemas distribuídos em que os nós estão localizados em posições físicas no espaço. Essas redes são amplamente utilizadas em diversas áreas, como telecomunicações, transporte, sensoriamento remoto e computação distribuída.

A implementação de uma rede geográfica pode ser realizada por meio da definição de vértices e arestas. Cada vértice é definido por suas coordenadas geográficas (latitude e longitude, por exemplo), enquanto as arestas representam as conexões entre os vértices. As arestas podem ser ponderadas ou não, dependendo da aplicação.

Foi implementado da seguinte forma: a classe *Vertice* define um vértice com as coordenadas x e y. Já a classe *Aresta* define uma aresta com dois vértices e um peso. A função *gera_vertices* cria uma lista de vértices aleatórios com base em um número n. A função *gera_arestas* cria uma lista de arestas com base nos vértices gerados e em um parâmetro eps, que controla a probabilidade de duas arestas serem conectadas em função da distância geográfica entre elas.

A função *gera_rede_geografica* utiliza as funções *gera_vertices* e *gera_arestas* para gerar uma rede geográfica completa com n vértices e um parâmetro eps definido pelo usuário. A função *retorna_vertices_conectados* retorna uma lista de vértices conectados a um determinado vértice de origem. Já a função *distancia_heuristica* calcula a distância euclidiana entre dois pontos no espaço, por fim, a função *mostra_arestas* imprime todas as arestas da rede na tela.

Em resumo, as redes geográficas são uma ferramenta poderosa para modelar sistemas distribuídos em que a localização física dos nós é relevante. A implementação dessas redes pode ser realizada por meio da definição de vértices e arestas, e o parâmetro eps pode ser ajustado para controlar a conectividade da rede. O código apresentado é um exemplo de implementação simples de uma rede geográfica que pode ser adaptado para diferentes aplicações.

A seguir estão 3 figuras onde são visualizações das redes geográficas. Em cada uma delas é variando o lambda. Como esse lambda é o expoente negativo de uma exponencial, quanto menor seu valor, maior sua influência no $\mathbb{P}$.

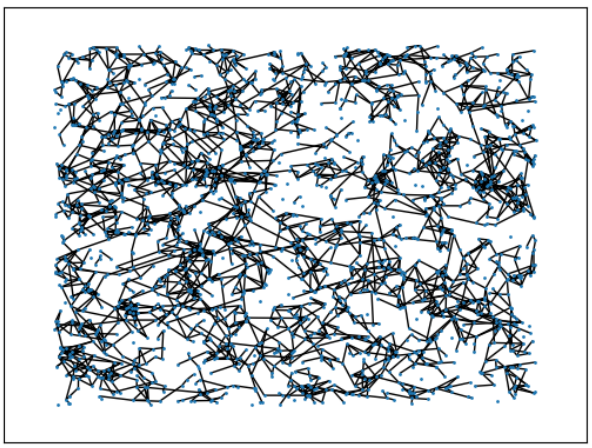

figura 1: Exemplo de rede geográfica n=2000,λ=0.03

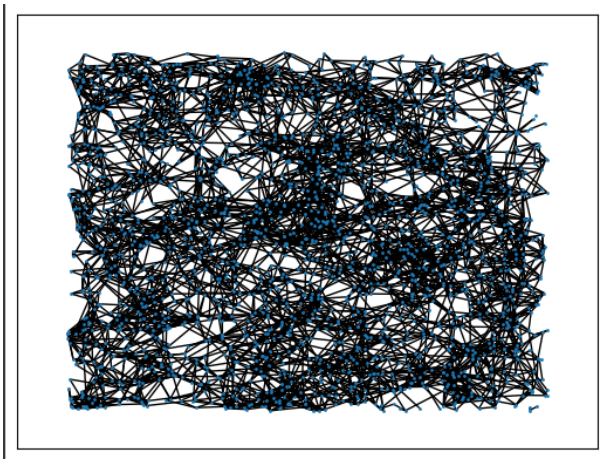

figura 2: Exemplo de rede geográfica n=2000,λ=0.02

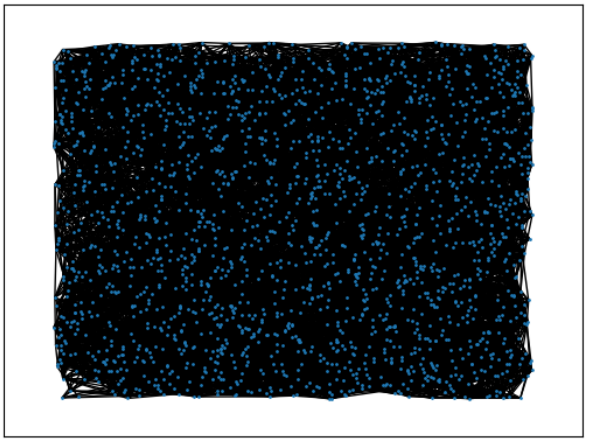

figura 3: Exemplo de rede geográfica n=2000, λ=0.01

Algoritmos de Busca e as Implementações

Busca em Profundidade

A implementação consiste em adicionar o vértice de origem em uma lista. Essa lista é usada para armazenar os vértices que serão visitados em ordem de profundidade.

Em seguida, o algoritmo entra em um loop que executa enquanto a lista não estiver vazia. Em cada iteração, o vértice mais profundo da lista é retirado e é verificado se esse vértice é o vértice de destino. Se for, a busca é concluída.

Caso contrário, o algoritmo adiciona todos os vértices conectados ao vértice atual na lista, desde que esses vértices ainda não tenham sido visitados. Esse processo é repetido até que o vértice de destino seja encontrado ou até que não haja mais vértices para serem visitados.

Ao final da busca, é exibido o caminho percorrido, o peso total do caminho encontrado e o tempo de execução do algoritmo.

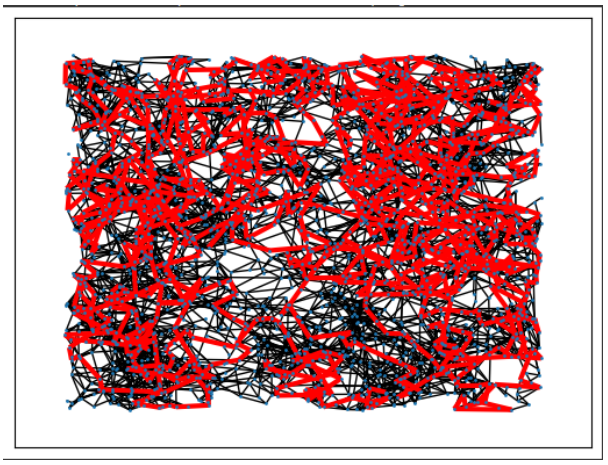

figura 4:Caminho busca em Profundidade n=2000,**λ**=0.02

Busca em Largura

A busca em largura é um algoritmo de busca de grafos que começa em um vértice raiz e explora todos os vértices vizinhos antes de se mover para os vértices seguintes. Este algoritmo é amplamente utilizado em diversas aplicações, como em sistemas de navegação, redes sociais e jogos.

A implementação deste algoritmo é relativamente simples e pode ser feita utilizando uma fila para armazenar os vértices que ainda não foram visitados. O algoritmo começa inserindo o vértice raiz na fila e, em seguida, explora todos os seus vizinhos. Cada um desses vizinhos é adicionado à fila, e o processo é repetido até que todos os vértices tenham sido visitados.

O algoritmo começa inserindo o vértice de origem na fila com peso zero. Em seguida, ele entra em um loop enquanto a fila não estiver vazia. A cada iteração do loop, o algoritmo retira o próximo vértice da fila e verifica se ele é o vértice de destino. Se for o vértice de destino, a busca é concluída com sucesso. Caso contrário, o algoritmo verifica se o vértice já foi visitado. Se o vértice ainda não foi visitado, ele é adicionado à lista de visitados e o peso total da busca é atualizado.

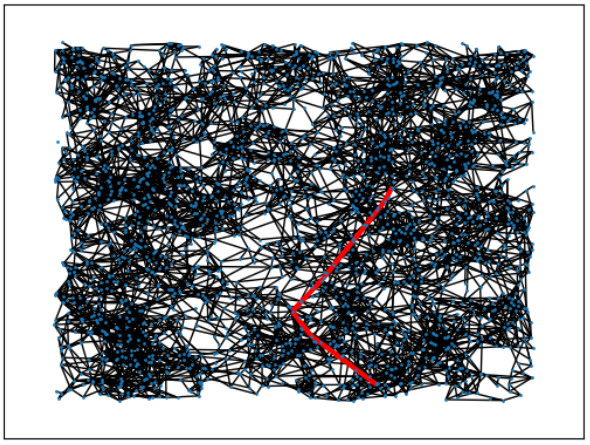

figura 5: Caminho busca em largura n=2000,**λ**=0.02

Best-First Search

O algoritmo de busca *Best First Search* é uma técnica utilizada em Inteligência Artificial para encontrar o caminho mais curto entre dois pontos em um grafo. Ele é baseado em uma heurística que avalia a distância entre o nó atual e o destino final, escolhendo sempre o caminho que parece mais promissor.

O algoritmo percorre a lista de nós a serem visitados, escolhendo sempre o próximo nó com base na heurística de distância até o destino final. Caso o nó escolhido seja o destino final, a busca é considerada bem sucedida e o caminho percorrido é exibido na tela, juntamente com o peso total, caso contrário, o nó escolhido é adicionado à lista de nós visitados e seus nós conectados são adicionados à lista de nós a serem visitados, desde que ainda não tenham sido visitados anteriormente.

Nesse experimento, foi usado a distância heurística sendo a distância euclidiana.

O algoritmo *Best First Search* é uma técnica eficiente para encontrar o caminho mais curto entre dois pontos em um grafo.

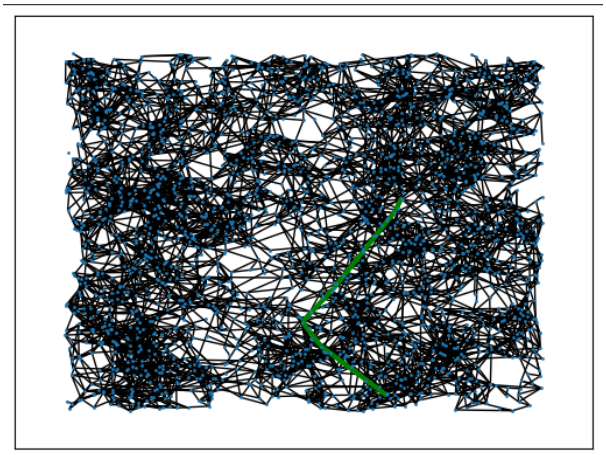

figura 6: Caminho Best First n=2000,**λ**=0.02

A*

O algoritmo de busca *A** é uma técnica utilizada para encontrar o caminho mais curto entre dois pontos em um grafo ponderado. Ele utiliza uma heurística que estima a distância restante entre o nó atual e o destino final, permitindo que ele escolha a melhor opção de caminho.

O algoritmo A* começa com a definição da função de busca, que recebe a origem, o destino e uma lista de arestas do grafo. Em seguida, é definido um tempo inicial e a lista de nós a serem visitados é iniciada com a origem. O algoritmo usa uma variável para indicar se o caminho foi encontrado com sucesso ou não e outra para armazenar o peso total do caminho encontrado.

O algoritmo percorre os nós conectados ao nó atual, ordenando-os de acordo com a distância heurística e adicionando-os à lista de nós a serem visitados. O loop continua enquanto houver nós conectados e o último nó da lista não estiver na lista de visitados. Se o nó atual for igual ao destino, o caminho foi encontrado com sucesso e o peso total do caminho é atualizado. Caso contrário, o nó atual é adicionado à lista de visitados e o peso total do caminho é atualizado com o peso do nó atual.

Nesse experimento, foi usado a distância heurística sendo a distância euclidiana.

Em resumo, o algoritmo A* é uma técnica eficiente para encontrar o caminho mais curto em um grafo ponderado. A utilização de heurísticas adequadas permite obter resultados precisos em um curto espaço de tempo.

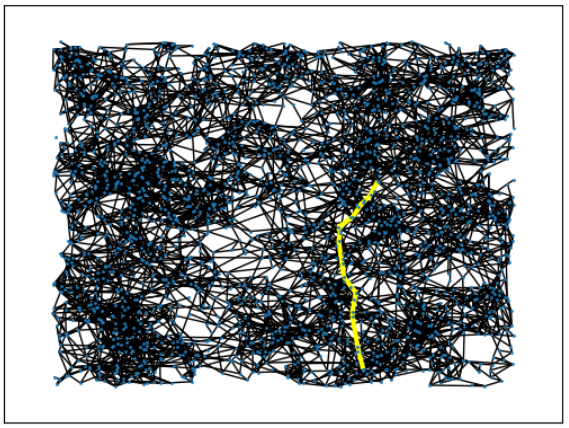

figura 7: Caminho A* n=2000,**λ**=0.02

Experimentos

Para n = 2000, λ = 0.01

```
Algoritmo: Busca em Largura
Distância média percorrida: 1265.4060341280308
Tempo médio gasto: 0.04015810489654541

Algoritmo: Busca em Profundidade
Distância média percorrida: 276011.1278017887
Tempo médio gasto: 0.5343597650527954

Algoritmo: Best-First Search
Distância média percorrida: 1261.8120680867912
Tempo médio gasto: 0.0007667303085327149

Algoritmo: A*
Distância média percorrida: 1483.632417261745
Tempo médio gasto: 0.019356346130371092
```

Para n = 2000, λ = 0.02

```
Algoritmo: Busca em Largura
Distância média percorrida: 1403.0575620404632
Tempo médio gasto: 0.004415488243103028

Algoritmo: Busca em Profundidade
Distância média percorrida: 102921.55655981504
Tempo médio gasto: 0.048856592178344725

Algoritmo: Best-First Search
Distância média percorrida: 1386.6012634974945
Tempo médio gasto: 0.0008963823318481445

Algoritmo: A*
Distância média percorrida: 1596.0604799012306
Tempo médio gasto: 0.01146693229675293
```

Para n = 2000, λ = 0.03

```
Algoritmo: Busca em Largura
Distância média percorrida: 1793.436018544712
Tempo médio gasto: 0.003268122673034668

Algoritmo: Busca em Profundidade
Distância média percorrida: 35602.67568824817
Tempo médio gasto: 0.011331915855407715

Algoritmo: Best-First Search
Distância média percorrida: 1782.3255867517462
Tempo médio gasto: 0.0015323162078857422

Algoritmo: A*
Distância média percorrida: 1760.6718142294073
Tempo médio gasto: 0.01572127342224121
```

Conclusão

O algoritmo A* é geralmente considerado o mais eficiente e eficaz para o problema de encontrar caminhos em grafos geográficos, quando há informações heurísticas disponíveis. Ele combina a busca em largura com uma heurística admissível, permitindo encontrar a solução ótima garantida, desde que a heurística seja bem escolhida. Comparado aos outros algoritmos, o A* é capaz de encontrar o caminho mais curto em termos de custo acumulado (ou distância) de maneira mais eficiente. No entanto, é importante notar que o desempenho do A* pode depender fortemente da qualidade da função heurística utilizada.

A busca em largura é útil para encontrar a solução mais curta em termos de número de arestas percorridas, mas pode ser ineficiente em grafos com alto fator de ramificação.

A busca em profundidade é adequada para encontrar soluções em profundidade, mas não garante encontrar a solução mais curta e pode ficar presa em loops ou caminhos muito longos.

O Best-First Search é útil quando informações heurísticas estão disponíveis e é desejável explorar os nós com maior chance de levar à solução. No entanto, seu desempenho depende fortemente da qualidade da função heurística utilizada.

Portanto, no contexto específico de grafos geográficos, onde o objetivo é encontrar o caminho mais curto em termos de distância, o Best-First Search com uma boa heurística é a melhor escolha.