Centro de Educação Federal Tecnólogica de Minas Gerais
Campus Divinópolis

Caminho Mínimo em um Grafo: Um Estudo de Caso com Destinos Sequênciais

Gabriel Oliveira Alves

Professor: Dr. Allison Marques
Disciplina de Inteligência Artificial

9 de Setembro de 2024

Sumário

Resumo

O trabalho aborda o problema do caminho mais curto para entregas sequenciais na cidade de Pimenta-MG, modelado como um grafo cujos nós representam pontos de referência nos bairros e as arestas, a distância euclidiana entre esses pontos. O objetivo foi analisar o desempenho de algoritmos de busca em cenários com três destinos sequenciais. Foram implementados três algoritmos: A* com heurística baseada na diferença absoluta dos graus dos nós (busca informada), além de BFS e DFS (buscas não informadas). As simulações foram realizadas em Python, utilizando a biblioteca NetworkX para modelar e manipular o grafo.

palavras-chave: Caminho mais curto, Algoritmo A*, BFS, DFS, Entregas sequenciais, Grafo, NetworkX.

1 Introdução

1.1 O problema

Nos últimos anos, Pimenta-MG tem passado por um processo de crescimento acelerado, impulsionado principalmente pela expansão do agronegócio. Esse desenvolvimento tem refletido diretamente no aumento das atividades de construção civil, tanto no setor industrial quanto no doméstico. Como resultado, a demanda por materiais de construção também cresceu, impactando empresas locais que precisam ajustar suas operações para atender seus clientes de forma eficiente.

A loja de materiais de construção "SUPREMA", recentemente começou a enfrentar dificuldades em gerenciar suas entregas. O desafio surge especialmente quando há necessidade de realizar múltiplas entregas em diferentes bairros da cidade, cada uma com destinos distintos e necessidades específicas. Esse cenário se traduz no problema de otimização de rotas para que as entregas sejam feitas no menor tempo possível, utilizando os recursos de forma eficiente.

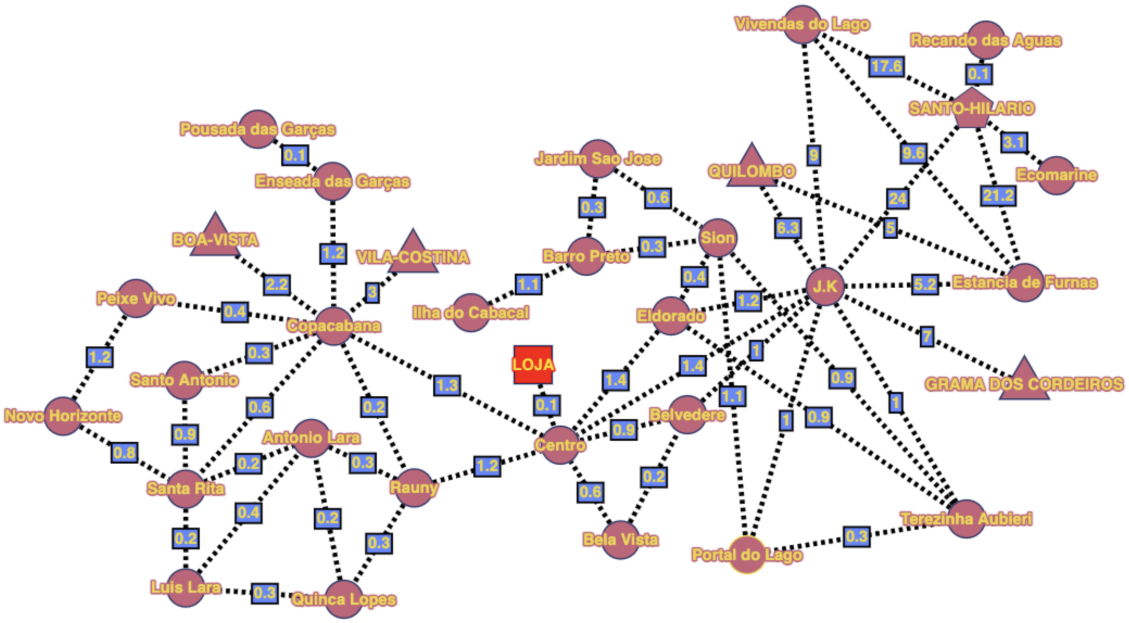

Figura 1: Pimenta representado por um Grafo Ponderado

1.2 Algoritmos

Para solucionar o problema de otimização de rotas, diferentes algoritmos de busca foram aplicados. O algoritmo A* foi utilizado como uma abordagem de busca informada, onde uma heurística foi adotada para guiar o processo de busca. A heurística escolhida baseia-se na diferença absoluta dos graus dos nós, o que permite estimar a proximidade dos destinos de maneira eficiente. Além disso, foram aplicados os algoritmos BFS (Busca em Largura) e DFS (Busca em Profundidade) como abordagens

de busca não informada, servindo como referência para comparar o desempenho em termos de tempo e precisão das rotas encontradas.

2 Implementação

Todos os algoritmos foram implementados manualmente, apesar de a biblioteca NetworkX oferecer versões já prontas desses algoritmos. A escolha pela implementação própria foi motivada pela necessidade de personalizar a solução para que os algoritmos retornassem, além do caminho encontrado e seu custo, também a quantidade de nós explorados durante o processo de busca. Essa abordagem permitiu um controle mais detalhado das métricas de desempenho, essenciais para a análise comparativa dos métodos utilizados.

O grafo na figura 2, abaixo representa o grafo da figura 1, mas plotado por meio da biblioteca `NetworkX`.

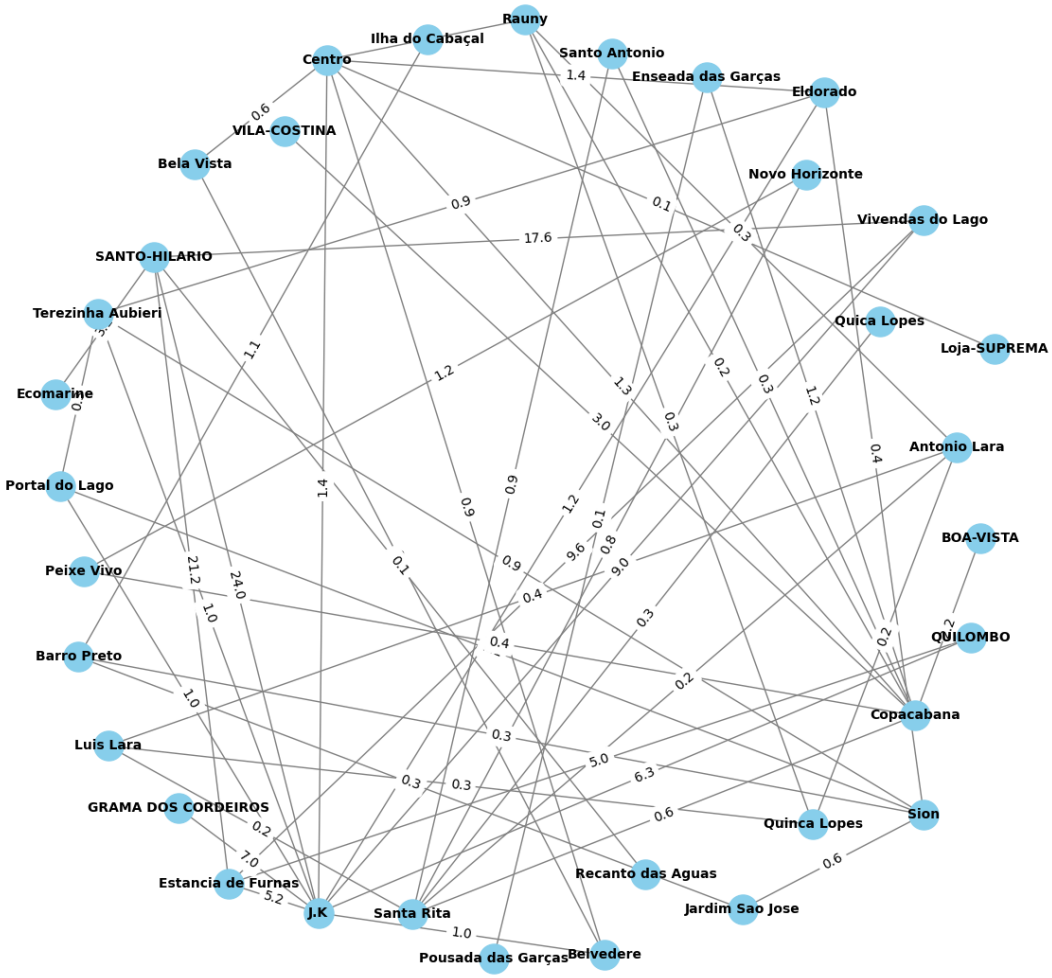

Figura 2: Pimenta representado por um Grafo Ponderado

2.1 A* (A-estrela)

O A* é um algoritmo de busca que encontra o caminho mais curto entre dois nós em um grafo. Ele usa dois custos.

- **g(n)**: Custo real do caminho desde o nó inicial até o nó atual.
- **h(n)**: Função heurística que estima o custo do nó atual até o destino.

O A* combina ambos para calcular f(n) = g(n) + h(n), priorizando nós com menor f(n) para exploração, garantindo o menor caminho se a heurística for admissível (não superestima o custo). Seu custo computacional é $O(|E| + |V|log|V|)$, Sendo `E` a quantidade de arestas e `V` a quantidade de nós no grafo.

A heurística utilizada considera a diferença de graus entre dois nós, por definição u e v. Sendo $h(u, v) = |grau(u) - grau(v)|$ que mede o quanto um nó está "conectado"e usa isso como uma estimativa da distância entre os nós, mas é uma heurística simplista que não necessariamente reflete a distância geográfica ou peso real no grafo.

2.2 BFS (Busca em Largura)

A BFS é um algoritmo de busca que explora todos os nós de um grafo de maneira uniforme, nível por nível, a partir de um nó inicial. Ele garante encontrar o caminho com o menor número de arestas em grafos não ponderados.

- **Fila**: Utiliza uma fila para armazenar nós a serem explorados.
- **Exploração em Níveis**: Explora todos os vizinhos de um nó antes de passar para o próximo nível.

A BFS é ideal para encontrar caminhos com o menor número de arestas, mas não considera pesos de arestas. Seu custo computacional é $O(|E| + |V|)$, onde `E` é a quantidade de arestas e `V` é a quantidade de nós no grafo.

2.3 DFS (Busca em Profundidade)

A DFS é um algoritmo de busca que explora o grafo o mais profundamente possível a partir de um nó inicial antes de retroceder. Ele não garante encontrar o menor caminho, mas é eficiente para explorar grandes áreas de um grafo.

- **Pilha**: Utiliza uma pilha (pode ser implementada recursivamente) para armazenar o caminho atual.
- **Exploração em Profundidade**: Explora completamente cada caminho antes de voltar e tentar um novo.

A DFS pode ser usada para descobrir componentes conexos ou detectar ciclos, mas não encontra necessariamente o caminho mais curto. Seu custo computacional é $O(|E|+|V|)$, semelhante à BFS, onde `E` é a quantidade de arestas e `V` é a quantidade de nós no grafo.

3 Resultados

Os resultados mostraram que a heurística de graus foi a única possível de implementar devido à ausência de coordenadas rastreáveis no grafo, o que impediu o uso de heurísticas baseadas em fórmulas de distância, como a euclidiana(usadas na construção original do grafo) e a de Manhattan. Com o algoritmo A*, também foi possível simular o algoritmo de Dijkstra ao não utilizar nenhuma heurística.

Para conseguir comparar a performace dos algoritmos, foram executadas 26.970 vezes para cada algoritmo - visto que são 3 destinos sequenciais - que equivale a combinação de 3 em 3 para 31 nós. O tempo médio foi calculado após 10 execuções, neste formato. A tabela a seguir mostra os resultados obtidos.

Algoritmo	**Média de Tempo (s)**	**Desvio Padrão Tempo (s)**	**Média de Nós Explorados**	**Desvio Padrão de Nós Explorados**
A* c/ Heurística	0.00086	0.000031	42.15	17.51
A* s/ Heurística	0.00079	0.000024	50.15	15.81
BFS	0.0000166	0.0000042	48.88	18.80
DFS	0.0000162	0.0000041	49.31	17.64

Tabela 1: Resultados de tempo e nós explorados para diferentes algoritmos

Pode-se observar que, apesar de apresentar o maior tempo médio, o algoritmo A* com heurística explorou a menor quantidade de nós, demonstrando uma boa eficiência computacional. No entanto, o caminho gerado por esse algoritmo foi mais longo em comparação ao BFS e ao A* sem heurística.

As imagens abaixo ilustram o comportamento de cada algoritmo para uma sequência de destinos entre três bairros, considerando que cada entrega parte da loja e segue sequencialmente para os três destinos. A lista de "bairros-destinos"das imagens a seguir foram *J.K, Peixe Vivo, Barro Preto*

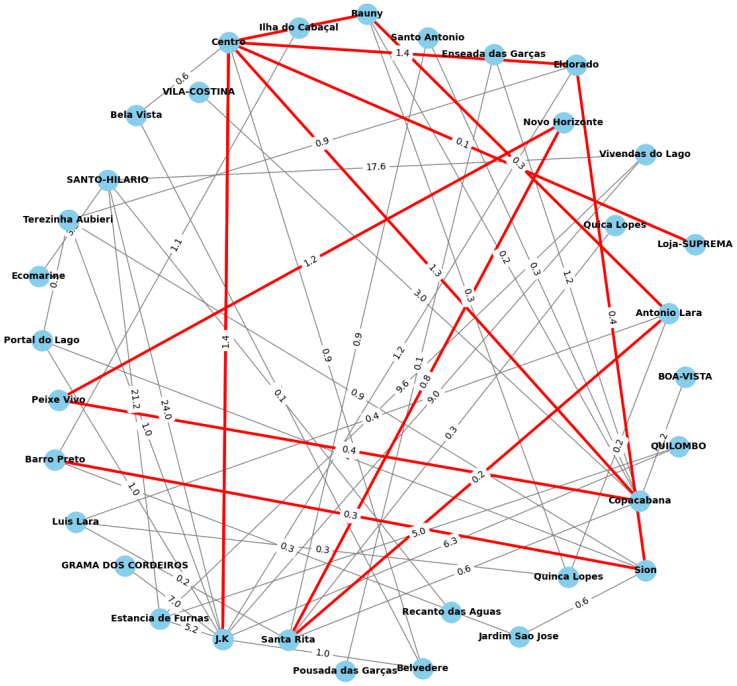

Figura 3: A* com Heurística

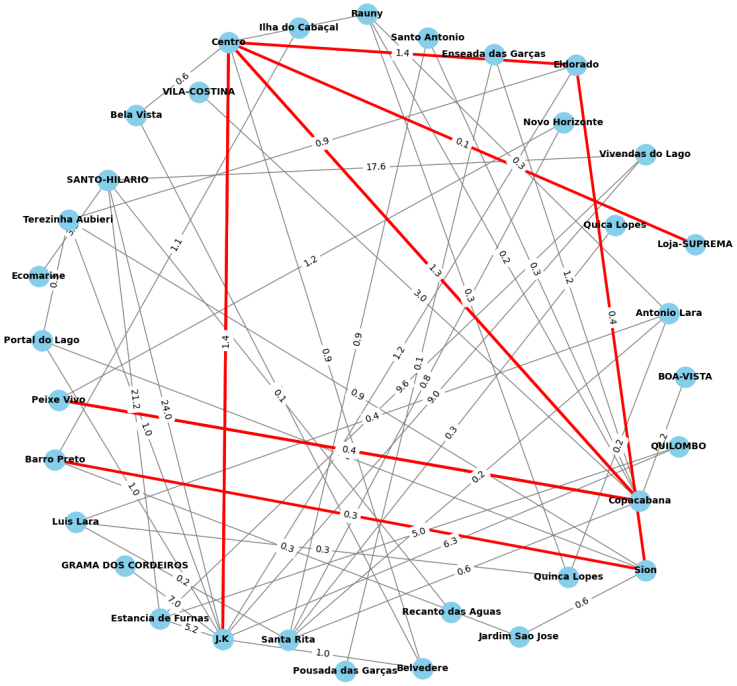

Figura 4: A* sem Heurística (Dijkstra)

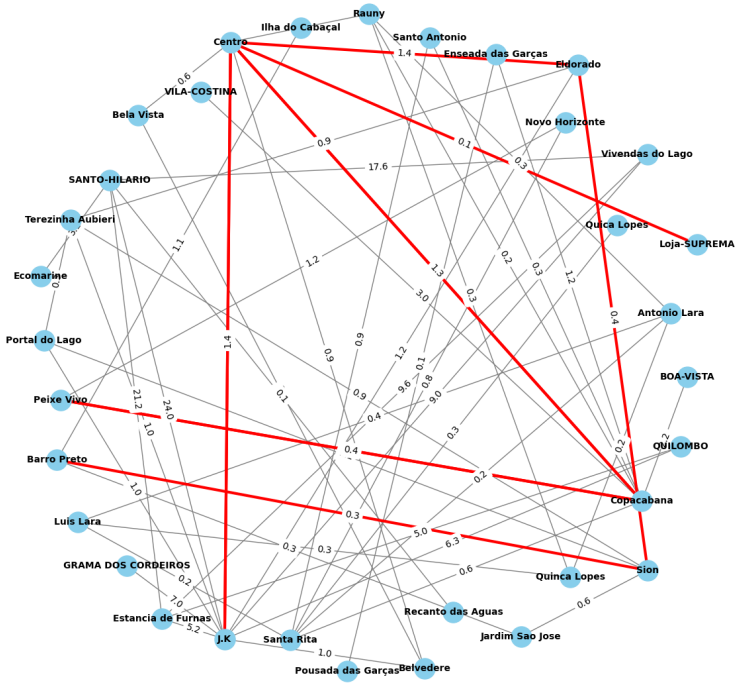

Figura 5: BFS

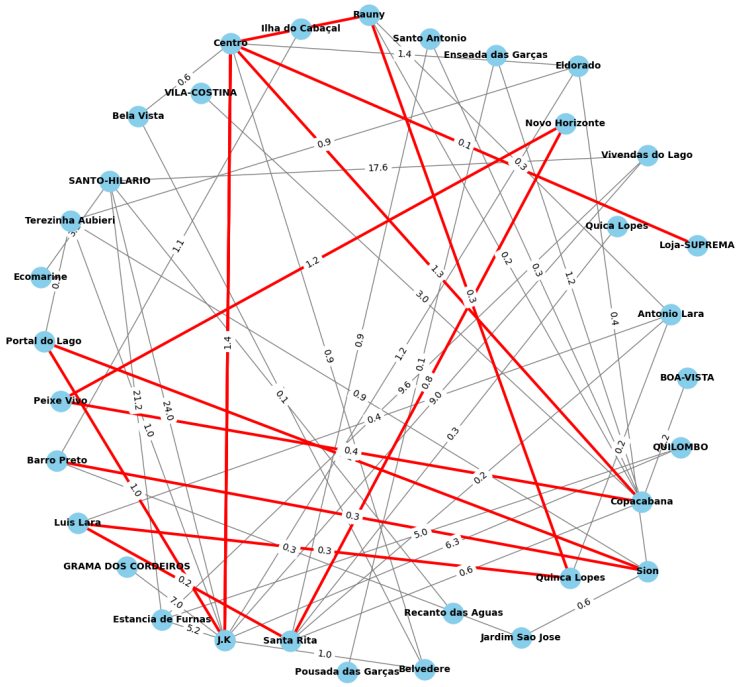

Figura 6: DFS

Na figura 3 a quantidade de nós explorados foi 45 e o custo total do caminho foi de 10.4. Na figura 4 demonstra que o algoritmo de Dijkstra encontrou o menor

caminho com custo total igual a 8.5, ao explorar 46 nós. O BFS, também conseguiu encontrar o menor caminho, entretanto, foi necessário explorar 53 nós. O DFS teve o pior desempenho, ao explorar 65 nós e conseguir um caminho de custo total igual a 12.5.

4 Conclusão

Conclui-se que a heurística utilizada no algoritmo A* não foi a ideal, embora tenha sido a única possível de ser aplicada neste caso. Apesar de ser mais eficiente em termos computacionais, explorando menos nós que o DFS e o BFS, ela não conseguiu gerar o caminho mínimo. O próprio A* sem heurística, que se comporta como o algoritmo de Dijkstra, foi mais eficaz ao encontrar o caminho mínimo, mesmo explorando uma quantidade média maior de nós.

Embora a análise tenha considerado três algoritmos, a melhor abordagem para otimizar as entregas no problema proposto seria obter as distâncias reais, modelar um grafo com esses dados e aplicar o algoritmo de Floyd-Warshall. Executado uma única vez, esse algoritmo geraria uma matriz de distâncias de custo mínimo entre todos os pares de nós. Apesar de seu custo elevado para grafos densos, ele garantiria o menor caminho entre qualquer par de destinos, eliminando a necessidade de novas verificações a cada transição entre destinos.