UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DE COMPUTAÇÃO
CURSO DE INFORMÁTICA/PPGI - MESTRADO ACADÊMICO

RELATÓRIO DE INTELIGÊNCIA ARTIFICIAL
ATIVIDADE AVALIATIVA - PROBLEM SOLVING

William Gabriel da Paz Rosendo: 2025100304

Maceió - AL
2025

William Gabriel da Paz Rosendo

ATIVIDADE AVALIATIVA - PROBLEM SOLVING

Relatório apresentado à disciplina de Inteligência Artificial, correspondente à avaliação do semestre 2025.1 do curso de Pós-Graduação em Informática/PPGI da Universidade Federal de Alagoas, sob orientação de **Prof. Evandro de Barros Costa e Prof. Glauber Rodrigues Leite.**

Maceió - AL
2025

LISTA DE SÍMBOLOS

A*	Algoritmo A-Estrela.
r	Raio.
G	Instância de um grafo.
V	Conjunto de vértices do grafo G.
E	Conjunto de arestas do grafo G.
n	Quantidade de vértices do grafo.

Lista de Figuras

Sumário

1 Introdução

Este relatório apresenta a atividade prática baseada no conteúdo *Problem Solving*, com o objetivo de resolver o problema de determinação da menor rota entre duas cidades. Para isso, utiliza-se a distância euclidiana como medida entre os pontos geográficos das cidades.

A instância do problema é fornecida por um arquivo JSON que contém informações geográficas e populacionais das cidades mais populosas dos Estados Unidos. A partir dessas informações, foram aplicadas técnicas de Inteligência Artificial, fundamentadas nos conceitos do Capítulo 3 do livro, para encontrar o caminho de menor distância acumulada entre duas cidades arbitrárias.

Um parâmetro importante do problema é o valor r, que representa a distância máxima para considerar que existe uma rota direta entre duas cidades. Ou seja, se a distância entre duas cidades for menor ou igual a r, considera-se que há uma estrada conectando as duas cidades diretamente. Com base nessas conexões, o desafio consiste em encontrar o caminho de menor custo (menor distância acumulada) entre a cidade de origem e a cidade de destino, considerando apenas as rotas válidas definidas pela restrição de distância r. Além disso, em caso de empate entre cidades com o mesmo custo, deve-se priorizar a estrada que leva à cidade com menor população.

2 Modelagem

Esta seção apresenta a modelagem do problema, descrevendo o modelo PEAS (Performance, Environment, Actuators, Sensors) e fornecendo uma representação abstrata do ambiente. A modelagem é feita com base nos conceitos de espaço de estados, conjunto de ações, modelo de transição e função ação-custo.

2.1 PEAS

O modelo PEAS é uma estrutura usada para definir e analisar sistemas de inteligência artificial. Ou seja, esse modelo ajuda a organizar e entender os elementos fundamentais de um sistema inteligente. A Tabela 1 apresenta o modelo PEAS do problema abordado.

Tabela 1: Componentes PEAS do problema

Componente	**Descrição**
Performance	Minimizar a distância total da rota; em caso de empate, priorizar cidades com menor população.
Environment	Grafo onde: - As cidades são representadas como os vértices do grafo. - As estradas entre cidades (arestas) são definidas se a distância euclidiana entre elas for $\leq r$. - Cada cidade possui atributos como: nome, latitude, longitude e população.
Actuators	Algoritmos de busca (A* e Bellman-Ford) que selecionam o próximo vértice a ser visitado.
Sensors	Leitura do arquivo JSON contendo os dados das cidades.

2.2 Representação Abstrata

Para resolver o problema, foi construído um grafo, onde:

- Os vértices representam as cidades listadas no arquivo JSON. Cada vértice contém informações como o nome da cidade, latitude, longitude e população.
- As arestas representam conexões diretas entre duas cidades, criadas sempre que a distância euclidiana entre elas é menor ou igual a um raio r. O peso de cada aresta corresponde a essa distância.

Assim, os vértices do grafo representam as cidades, enquanto as arestas indicam as estradas que conectam diretamente as cidades associadas aos respectivos vértices. A Figura 1 apresenta o mapa das cidades fornecidas no arquivo JSON, destacando a cidade inicial (Suffolk, Virginia) e a cidade final (El Paso, Texas), que foram fornecidas como parâmetros. Já a Figura 2 mostra o grafo gerado a partir do mapa das cidades, considerando um valor de raio $r = 300km$ para definir a existência de uma estrada direta entre duas cidades.

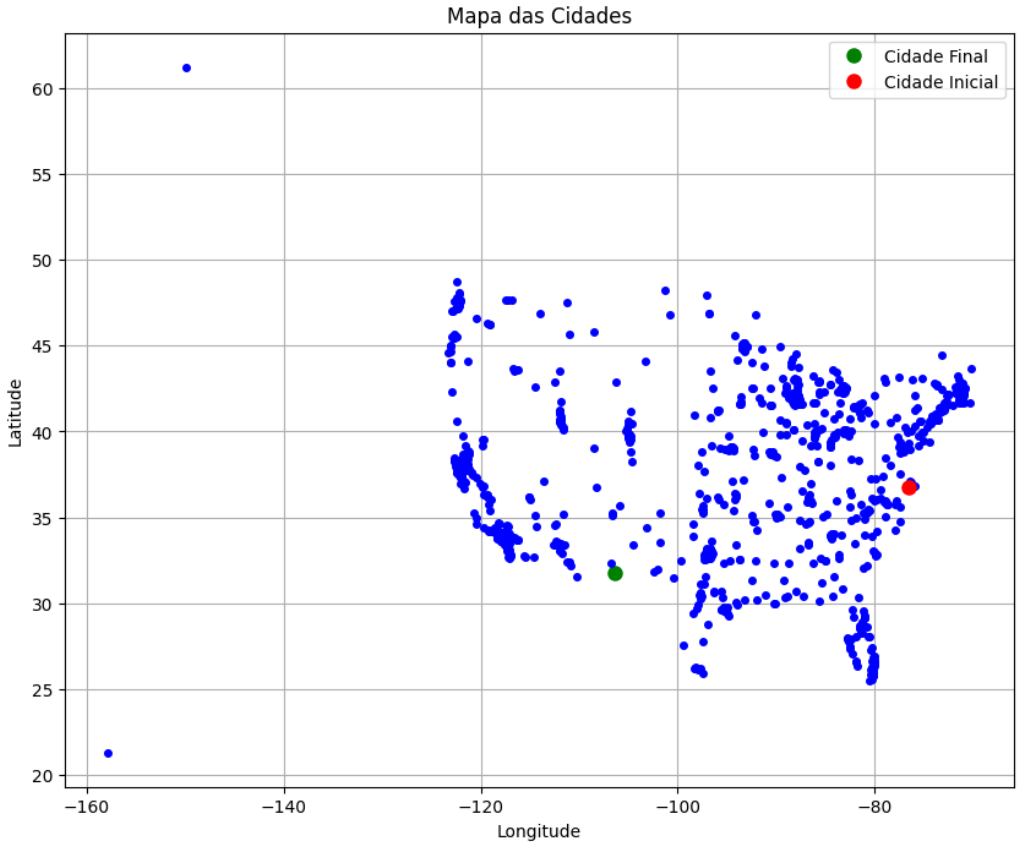

Figura 1: Mapa das cidades.

A partir da representação em formato de grafo, é possível modelar o ambiente de maneira abstrata. Com base nessa representação, temos:

- **Espaço de estados:** Conjunto de todas as cidades do grafo, onde cada cidade é um estado.
- **Estado inicial:** Qualquer estado pode ser designado como o estado inicial.

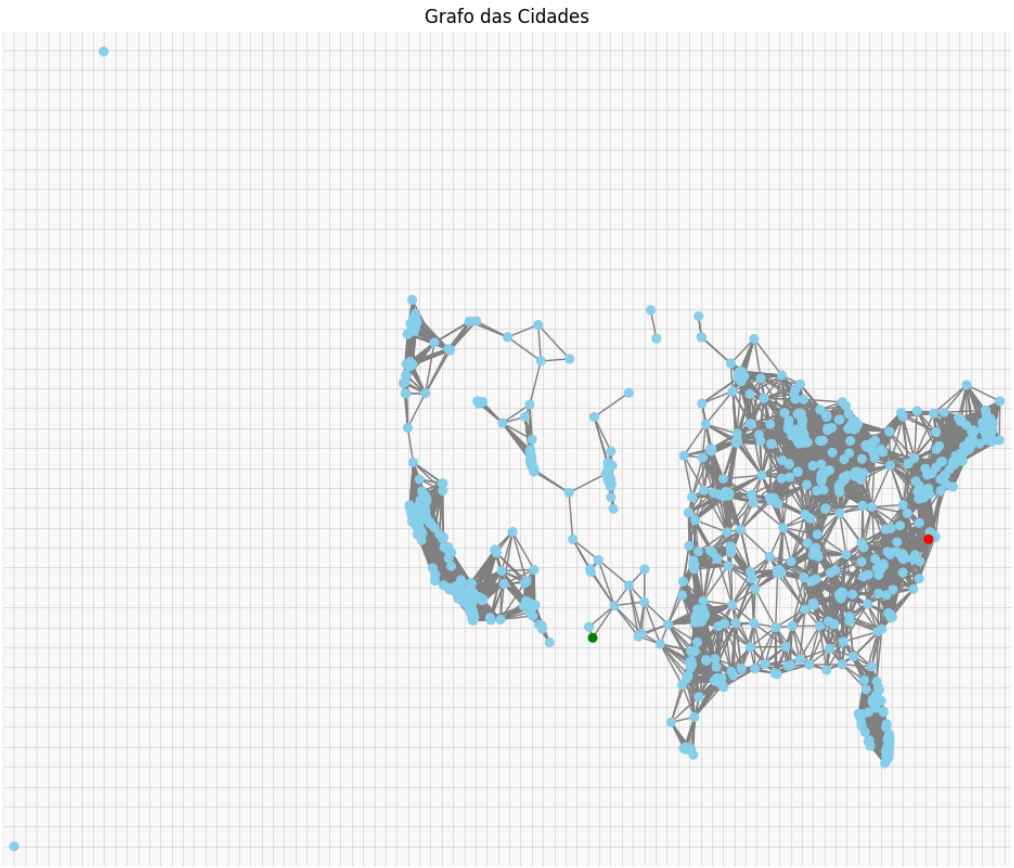

Figura 2: Grafo resultante para $r = 300km$.

- **Conjunto de ações:** Mover-se de uma cidade para uma cidade vizinha conectada por uma estrada ($distancia \leq r$). Ex: Se estou na cidade A, posso ir para B, C ou D, se elas estiverem a uma distância $\leq r$.
- **Modelo de transição:** A transição ocorre de um estado S (cidade atual) para um novo estado S' (cidade vizinha), por meio da ação de deslocar-se pela estrada que conecta as duas. Assim, temos: estado atual $\rightarrow$ ação $\rightarrow$ novo estado.
- **Função ação-custo:** O custo da ação de ir de uma *cidade_i* para uma *cidade_j* é a distância euclidiana entre as duas. Em caso de empate de custo, a escolha favorece a cidade com menor população. Logo, *Custo* = *Distância Euclidiana*, e em caso de empate entre cidades, *Custo* = *Distância Euclidiana* + *penalização para cidades com maior população*.

3 Implementação

Para resolver o problema de caminho com menor distância acumulada entre duas cidades, foram implementados dois algoritmos de busca. O algoritmo A*, que é abordado no capítulo 3 do Livro usado na disciplina, e o algoritmo de Bellman-Ford.

3.1 A*

O algoritmo A* é uma técnica de busca eficiente amplamente utilizada para encontrar o caminho mais curto entre dois vértices em um grafo. Ele é muito utilizado em várias áreas, como navegação, inteligência artificial e jogos. O A* combina duas abordagens de busca para otimizar o processo:

- Dijkstra: Este algoritmo encontra o caminho mais curto de um vértice inicial para todos os outros vértices, explorando todos os caminhos de maneira ótima.
- Busca Gulosa: Esse algoritmo explora os vértices mais próximos do objetivo, com base em uma estimativa heurística.

O A* combina esses dois métodos ao considerar, para cada vértice do grafo:

- $g(n)$: O custo real de percorrer o caminho até o vértice atual.
- $h(n)$: A estimativa do custo de chegar até o objetivo a partir do vértice atual, também chamada de heurística.
- $f(n)$: O custo total estimado para alcançar o objetivo a partir do vértice atual, que é dado por $g(n) + h(n)$.

Essa combinação de abordagens torna o A* tanto eficiente quanto preciso, pois ele leva em conta o que já foi percorrido e uma estimativa inteligente do que ainda falta percorrer.

O algoritmo A* começa a busca a partir do vértice inicial. Ele utiliza uma fila de prioridade para explorar, a cada iteração, o nó com o menor valor de $f(n)$, ou seja, o caminho mais promissor. Durante o processo, ele mantém uma lista de visitados, que contém os vértices que já foram avaliados, para evitar revisitas e redundâncias. O algoritmo continua explorando os vértices que não foram visitados até que o vértice final seja alcançado ou até que determine que não há caminho possível. Quando o vértice final é encontrado, o caminho mais curto é reconstruído com base nos vértices visitados.

Para problemas em um espaço de busca baseado em grades ou mapas, as funções de heurística mais comuns são:

- **Distância Manhattan**: É uma medida que calcula a soma das diferenças absolutas entre as coordenadas x e y dos dois pontos. Essa função é útil para problemas em que os movimentos só podem ser feitos em linhas retas ou em quadrados (como em grades).
- **Distância Euclidiana**: Mede a distância em linha reta entre dois pontos, usando o teorema de Pitágoras. Essa heurística é mais precisa em espaços contínuos e em mapas reais, onde o movimento não se limita a direções específicas.

Para resolver o problema apresentado, foi considerada a Distância Euclidiana para representar a distância entre duas cidades.

O A* é muito eficaz, pois encontra o caminho ótimo, desde que a heurística utilizada seja admissível, ou seja, nunca superestime o custo real do caminho. Além disso, o A* é eficiente, pois ele evita explorar caminhos desnecessários ao avaliar apenas os vértices mais promissores.

3.2 Bellman-Ford

O algoritmo de Bellman-Ford é uma técnica fundamental em teoria dos grafos utilizada para encontrar os caminhos mais curtos de um vértice inicial até todos os outros vértices em um grafo. Sua principal vantagem é a capacidade de lidar com arestas de peso negativo, algo que outros algoritmos, como o de Dijkstra, não conseguem fazer. Além disso, o Bellman-Ford pode identificar a existência de ciclos negativos, ou seja, ciclos cujo somatório dos pesos das arestas é menor que zero.

O funcionamento do algoritmo se baseia no conceito de relaxamento de arestas. O algoritmo monitora a distância de cada vértice em relação ao vértice inicial. Inicialmente, a distância até o vértice inicial é definida como zero, e a distância até qualquer outro vértice é considerada infinita. O algoritmo então encurta essas distâncias encontrando caminhos mais curtos até que não consiga reduzir mais a distância. Quando não houver mais arestas capazes de reduzir distâncias, significa que as distâncias são finais.

Os principais componentes do algoritmo Bellman-Ford são:

- Vetor de Distâncias: O algoritmo usa um vetor de distância para rastrear as distâncias provisórias do vértice inicial até cada vértice do gráfico. Inicialmente, todas as distâncias são definidas como infinitas, exceto a distância até o vértice inicial, que é definida como zero.

- Relaxamento: Para cada aresta do gráfico, ele verifica e ajusta repetidamente os valores de distância. Se a distância total até um vértice no caminho atual for menor do que a registrada, ele atualiza a distância até esse vértice. Esse procedimento é repetido diversas vezes ao longo da execução.

- Abordagem Iterativa: O algoritmo realiza o processo de relaxamento de forma iterativa. Ele percorre todas as arestas do grafo durante um número de iterações igual ao número de vértices menos um. Dessa forma, ele garante chances suficientes para atualizar todas as distâncias.

- Detecção de Ciclos Negativos: Uma característica importante do algoritmo Bellman-Ford é sua capacidade de detectar ciclos negativos. Um ciclo é considerado negativo quando a soma dos pesos de suas arestas resulta em um valor negativo. Se, após as iterações esperadas, o algoritmo ainda conseguir atualizar alguma distância, isso indica a existência de um ciclo negativo no grafo.

A repetição do processo garante que todas as rotas possíveis sejam consideradas. A complexidade do algoritmo é de $O(nm)$, onde n é o número de vértices e m é o número de arestas.

As aplicações do Bellman-Ford são diversas. Em redes de computadores, ele é utilizado em protocolos de roteamento, ajudando a encontrar os melhores caminhos para o tráfego de dados. Também é útil na verificação da conectividade de redes e na detecção de ciclos negativos, que podem comprometer o desempenho do sistema. Além disso, o algoritmo é aplicado em jogos digitais para determinar caminhos ótimos entre pontos no mapa, facilitando a movimentação de personagens.

4 Resultados

Esta Seção apresenta os resultados obtidos a partir das implementações dos métodos de busca.

4.1 Cenários de Utilização

Tabela 2: Cenários de Utilização

Cenário	r (km)	Cidade Inicial	Cidade Final	Solução Esperada
1	200	Philadelphia, Pennsylvania	Baltimore, Maryland	Solução ótima (estrada direta entre as cidades).
2	100	Los Angeles, California	Riverside, California	Solução ótima (estrada direta ou via 1 cidade intermediária).
3	1000	Honolulu, Hawaii	New York, New York	Sem solução (a distância da cidade mais próxima de Honolulu é maior que r).

- **Cenário 1:** Philadelphia e Baltimore estão a uma distância de $143.904km$, que é $\leq 200km$, portanto há uma estrada direta entre essas cidades fazendo com que seja possivel chegar a uma solução.
- **Cenário 2:** Los Angeles e Riverside estão a uma distância de $78.758km$, que é $\leq 100km$, portanto há uma estrada direta entre essas cidades fazendo com que seja possivel chegar a uma solução.

  Cenário 3: Honolulo está muito afastada das outras cidades e a cidade mais próxima dela é Pacifica, que está a 3868.180km de distância. Logo não existe arestas de Honolulo para outras cidades considerando o valor de $r = 1000km$, portanto não é possivel encontrar uma solução para este cenário.

4.2 Análise por Cenário

4.2.1 Cenário 1 - Solução Esperada

- $r = 200km$
- *Cidade Inicial: "Philadelphia, Pennsylvania"*
- *Cidade Final: "Baltimore, Maryland"*

A solução foi encontrada usando o algoritmo A* e o algoritmo Bellman-Ford.

- A* encontra o caminho seguindo a estrada direta entre *Philadelphia* e *Baltimore*.
  - Caminho: $Philadelphia \rightarrow Baltimore$

  - Custo: $\approx 143.9km$

- Bellman-Ford encontra a mesma rota.

O modelo proposto é dado por:

- **Espaço de estados**: Cada cidade é um estado.
- **Estado inicial (s):** *Philadelphia, Pennsylvania*
- **Estado final (g):** *Baltimore, Maryland*
- **Ações:** Mover-se de uma cidade para uma cidade vizinha conectada
- **Transição:** De um estado atual s, a ação a leva a um novo estado s' se $distancia(s,s') \leq r$
- **Função ação-custo:** $custo(s,s') =$ *Distância Euclidiana* entre s e s'
- **Critério de desempate (A):** Cidade com menor população

4.2.2 Cenário 2 - Solução Esperada

- $r = 100km$
- *Cidade Inicial: "Los Angeles, California"*
- *Cidade Final: "Riverside, California"*

A solução foi encontrada usando o algoritmo A* e o algoritmo Bellman-Ford.

- A* encontra o caminho seguindo a estrada direta entre *Los Angeles* e *Riverside*.
  - Caminho: *Los Angeles* $\rightarrow$ *Riverside*
  - Custo: $\approx 78.7583km$
- Bellman-Ford encontra a mesma rota.

O modelo proposto é dado por:

- **Espaço de estados**: Cada cidade é um estado.
- **Estado inicial (s):** *Los Angeles, California*
- **Estado final (g):** *Riverside, California*
- **Ações:** Mover-se de uma cidade para uma cidade vizinha conectada
- **Transição:** De um estado atual s, a ação a leva a um novo estado s' se $distancia(s,s') \leq r$
- **Função ação-custo:** $custo(s,s') =$ *Distância Euclidiana* entre s e s'

- **Critério de desempate (A):** Cidade com menor população

Neste cenário, considerando um valor menor para o raio, ou seja, $r = 70km$, a solução encontrada foi dada por:

- A* encontra o caminho seguindo uma rota por uma cidade intermediária entre *Los Angeles* e *Riverside*.
  - Caminho: *Los Angeles* $\rightarrow$ *La Puente* $\rightarrow$ *Riverside*
  - Custo: $\approx 78.7588km$
- Bellman-Ford encontra a mesma rota.

4.2.3 Cenário 3 - Sem Solução

- $r = 1000km$
- *Cidade Inicial: "Honolulu, Hawaii"*
- *Cidade Final: "New York, New York"*

Não exite solução para este cenário.

- O A* e o Bellman-Ford retornam "Nenhum caminho encontrado."
- Não há conexões suficientes no grafo com raio de $1000km$ para gerar um caminho a partir de Honolulu até New York

O modelo proposto é dado por:

- **Espaço de estados**: Cada cidade é um estado.
- **Estado inicial (s):** *Honolulu, Hawaii*
- **Estado final (g):** *New York, New York*
- **Ações:** Nenhuma viável
- **Transição:** Nenhuma transição possível
- **Função ação-custo:** Irrelevante

4.3 Comparação Entre os Algoritmos

No contexto geral:

Atributo	**A***	**Bellman-Ford**
Tipo de busca	Heurística e informada	Baseada em relaxamentos sucessivos
Heurística	Sim, precisa de uma função $h(n)$	Não usa heurística
Facilidade de implementar	Moderada, pois usa filas de prioridade e precisa de heurística boa	Simples em estrutura, mas requer várias iterações
Explicabilidade	Alta, pois pode justificar escolhas pela heurística	Média, segue relaxamentos até a convergência
Tempo de computação	$O(m)$ no melhor caso	$O(nm)$
Uso de memória	Baixo a moderado (fila de prioridade)	Alto (armazena distâncias para todos os vértices)

Tabela 3: Comparação entre os algoritmos A* e Bellman-Ford

No contexto específico do problema, as diferenças entre os algoritmos A* e Bellman-Ford tornam-se mais evidentes com base nas características do grafo e nos objetivos da busca. Para realizar uma melhor comparação entre os algoritmos, foi calculado o tempo(s) e a quantidade de iterações que cada algoritmo executou para chega na melhor solução. Toda a implementação está disponível no *link* anexado na Seção de anexos.

Considerando o problema de menor distância acumulada entre cidades, temos que:

A*

- Se beneficiou de uma heurística baseada na distância euclidiana.
- Efetivo para grafos esparsos com distâncias físicas razoáveis entre as cidades.
- Mais rápido para encontrar o caminho ótimo entre pares específicos de cidades.

Bellman-Ford

- Mais adequado para calcular caminhos de uma cidade para todas as outras, e não apenas entre um par específico.
- Mais lento e mais custoso computacionalmente no contexto de grandes conjuntos de dados.
- Pode ser menos eficiente, mas garante encontrar o melhor caminho mesmo com pesos negativos (não aplicável neste caso, mas importante para comparação).

Considerando algumas métricas observadas, temos que:

Tempo de Execução

- O A* foi mais rápido nas execuções práticas, especialmente ao buscar caminhos entre pares únicos de cidades.
- O Bellman-Ford demorou mais, devido à necessidade de iterar sobre todas as arestas múltiplas vezes.

Iterações

- O Bellman-Ford realizou significativamente mais iterações.
- O A* limitou-se às cidades promissoras graças à sua heurística.

Consumo de Memória

- O Bellman-Ford manteve mais tabelas de distâncias.
- O A* foi mais leve, utilizando apenas a fila de prioridade e alguns dicionários auxiliares.

5 Conclusão

5.1 Escalabilidade e Limitações

A implementação funciona bem para o número atual de cidades ($n = 1000$). Porém, conforme o número de cidades cresce exponencialmente (ex: 50.000 ou mais), ocorrem problemas sérios de desempenho fazendo com que os algoritmos possam não escalar bem.

5.2 Limitações dos Algoritmos de Busca em Grafos

Na aplicação de algoritmos de busca em grafos para mapas urbanos, diversas limitações devem ser consideradas, pois impactam diretamente a eficiência e a precisão dos resultados. Primeiramente, o tempo de execução é uma preocupação fundamental. À medida que o número de vértices e arestas cresce, especialmente em grafos densos, algoritmos como o Bellman-Ford, cuja complexidade é $O(nm)$, tornam-se impraticáveis em contextos com grande volume de dados.

O uso de memória também é um fator limitante. Estruturas como dicionários, listas de adjacência e tabelas de distâncias exigem um volume considerável de recursos, o que pode levar a lentidão ou até falhas em sistemas com pouca capacidade de processamento. Outro ponto importante é a precisão da heurística. Algoritmos como o A* dependem fortemente de uma boa função heurística. Quando baseada apenas na distância em linha reta, essa heurística pode não representar adequadamente obstáculos ou irregularidades do terreno, comprometendo a qualidade da solução encontrada.

Por fim, a conectividade do grafo pode ser um obstáculo. Se o parâmetro de conexão r for muito pequeno, há o risco de muitos pares de cidades não estarem conectados diretamente. Isso pode causar falhas na busca mesmo quando existe um caminho viável por vias indiretas.

5.3 Tornando o Problema Mais Realista ou Desafiador

Para tornar o problema mais realista ou desafiador, pode-se:

- **Incluir pesos reais:** Substituir a distância euclidiana por fatores mais realistas como tráfego, relevo, ou qualidade das estradas.
- **Modelar obstáculos:** Simular áreas intransitáveis, como rios, desertos ou zonas restritas, tornando o espaço de busca mais complexo.
- **Impor restrições nas rotas:** Considerar aspectos como horários de funcionamento das cidades, existência de pedágios ou proibições de trânsito.
- **Tornar o raio de conexão dinâmico:** Em vez de utilizar um valor fixo para r, adaptá-lo conforme características locais, como população ou importância da cidade.

ANEXOS

As implementações dos algoritmos A* e Bellman-Ford, bem como os resultados obtidos nas execuções, estão disponíveis no seguinte link:

https://colab.research.google.com/drive/1Max86hLewSLNMUm6ZGQc8mUHojNos30d?usp=sharing