Centro Federal de Educação Tecnológica de Minas Gerais
Engenharia de Computação - Inteligência Artificial

Trabalho 1 de IA: Busca no Labirinto

Pedro Auguto Gontijo Moura
Samuel Silva Gomes

Prof. Thiago Alves de Oliveira

Divinópolis/MG
Outubro de 2025

Introdução

O problema da busca no labirinto proposto envolve encontrar uma solução para o labirinto representado na Figura 1. Isso significa que, partindo de um dado ponto inicial (Start), o objetivo é encontrar um caminho (solução) que leve ao ponto final (Goal). Isso envolve tanto encontrar o melhor caminho dentro do labirinto quanto utilizar o algoritmo mais eficiente para encontrar esse caminho.

Os algoritmos de busca podem ser classificados em não informados, que exploram o espaço de estados de forma sistemática sem conhecimento adicional, e informados, que utilizam heurísticas para guiar a exploração em direção ao objetivo. Essa distinção é fundamental, pois impacta diretamente a eficiência da busca [3].

O presente trabalho tem como objetivo implementar e comparar algoritmos de busca informada (Busca Gulosa e A*) e não informada (BFS e DFS), analisando seu desempenho em métricas como tempo de execução, número de nós expandidos e gerados, consumo de memória e custo do caminho.

Figura 1: Labirinto do problema

O labirinto apresentado na Figura 1 foi representado computacionalmente por meio de uma matriz de adjacência. Cada posição da matriz corresponde a uma célula do labirinto, identificada por coordenadas na forma [linha, coluna]. Para cada célula, foram definidos quatro bits que indicam a presença ou ausência de paredes nas direções Norte (N), Sul (S), Leste (L) e Oeste (O). Dessa forma, a estrutura permite verificar de maneira eficiente quais movimentos são possíveis a partir de cada posição, servindo como base para a implementação dos algoritmos de busca.

Objetivos

- Implementar e consolidar consolidar conceitos de busca em espaço de estados em algoritmos de busca informada (Busca Gulosa e A*) e não informada (BFS e DFS);
- Aplicar métricas de desempenho para comparação e análise de métodos.
- Analisar a influência das heurísticas utilizadas (distância de Manhattan e distância Euclidiana) e dos parâmetros do problema nos resultados finais obtidos.

Metodologia

Materiais Utilizados

- Python 3.11.9 (linguagem de programação)
- Github (repositório de código) [1]
- Visual Studio Code (editor de texto)

Algoritmos de busca não informada utilizados:

Os algoritmos de busca não informados, também chamados de buscas cegas, exploram o espaço de estados sem utilizar conhecimento adicional sobre a localização do objetivo. Sua estratégia depende apenas da ordem de expansão dos nós, como no caso da busca em largura (BFS), que percorre camada por camada, e da busca em profundidade (DFS), que segue cada ramo até o fim antes de retroceder.

Ambos foram implementados com base no pseudocódigo apresentado em aula [2].

Algoritmo de busca em largura (BFS)

O algoritmo de busca em largura (do inglês *breadth-first search*, ou BFS) é usado principalmente para realizar buscas em grafos e árvores. A sua lógica é simples: com base em um ponto inicial P, todos os vizinhos de P são explorados. Para cada um desses vizinhos V, seus vizinhos são explorados, e isso se repete para cada "camada"de vizinhos até que a solução seja encontrada.

No código [1], esse algoritmo foi implementado com base em uma fila (deque) com os "nós"a serem explorados na ordem que corresponde à uma busca em largura. Com isso, também existe um conjunto (set) que representa os nós já visitados, a fim de se evitar revisitas desnecessárias, além de um dicionário usado para o mapeamento do labirinto, permitindo que o trajeto final possa ser reconstruído.

Algoritmo de busca em profundidade (DFS)

O algoritmo de busca em profundidade (do inglês *depth-first search*, ou DFS) também é usado na busca em árvores e grafos. Sua lógica consiste em explorar por completo cada ramo de uma estrutura a partir de um ponto inicial P antes de reteoceder.

A implementação em código foi semelhante à feita para o BFS, mas aqui foi utilizada uma pilha para armazenar os "nós"a serem explorados, já que a exploração é feita ramo a ramo nesse caso. Também existe um conjunto (set) de lugares já visitados e um dicionário (dict) que permite a reconstrução do camminho final. [1]

Algoritmos de busca informada utilizados

Os algoritmos de busca informados utilizam heurísticas para guiar a exploração em direção ao objetivo, tornando a busca mais eficiente. A busca gulosa avalia apenas a estimativa heurística $h(n)$, enquanto o algoritmo A* combina o custo real acumulado $g(n)$ com a estimativa $h(n)$, garantindo soluções ótimas quando a heurística é admissível e consistente.

Esses algoritmos também foram implementados com base nos pseudocódigos apresentados em aula [2].

Algoritmo de Busca Gulosa (*greedy-search*)

O algoritmo de busca gulosa (do inglês *greedy-search*) funciona de forma semelhante ao algoritmo de busca em profundidade (BFS), no sentido de que ele cada vez mais se aprofunda na estrutura de dados em busca da solução a partir de um ponto inicial n. Porém, a principal diferença vem do fato de que, como se trata de um algoritmo de busca informada, utiliza de uma heurística que estima uma distância ideal $h(n)$, até eventualmente chegar à solução do problema. Com isso, a função de avaliação desse algoritmo pode ser representada por $f(n)$, ta que:

$$f(n) = h(n)$$

A implementação parte de uma escolha entre uma das heurísticas escolhidas para guiar a resolução do problema (distância de Manhattan ou distância euclidiana). bAssim, o algoritmo explora os "nós"ordenado em um heap com base na heurística, com um conjunto que impede revisitas, um dicionário que permite reconstruir o caminho final e um variável que mapea o melhor "nó"para continuar o caminho com base apenas na menor distância (heurística) disponível. [1]

Algoritmo A*

O algoritmo A* busca o melhor caminho a partir de um nó n com base em uma heurística que estima uma distância $h(n)$ junto de uma distância real $g(n)$. A avaliação do algoritmo pode ser representada por uma função $f(n)$, tal que:

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

A diferença aqui em relação a busca gulosa é o uso adicional de um custo real acumulado $g(n)$ somado com a heurística $h(n)$. Logo, para cada "nó"explorado, a melhor opção em seguida envolve também obter o custo $f(n)$ para cada opção e armazená-lo. [1]

Heurísticas utilizadas

As heurísticas desempenham um papel fundamental nos algoritmos de busca informada, pois orientam a exploração do espaço de estados em direção ao objetivo. Neste trabalho, foram utilizadas duas heurísticas clássicas: a distância de Manhattan e a distância Euclidiana.

Distância de Manhattan

A distância de Manhattan diz que a distância entre dois pontos pode ser dada soma das diferenças absolutas de suas coordenadas. Isso quer dizer que, para dois pontos bidimensionais $P = (p_x, p_y)$ e $Q = (q_x, q_y)$, a distância entre eles pode ser dada por $D(P, Q)$, tal que:

$$D(P, Q) = |p_x - q_x| + |p_y - q_y|$$

Essa heurística é admissível, pois nunca superestima o custo real do caminho em um ambiente onde os movimentos são restritos a quatro direções. Além disso, é consistente e representa o número mínimo de passos necessários se fosse possível andar em linha reta apenas em direções ortogonais [3].

Distância Euclideana

A distância Euclideana pode ser entendida como a distância matemática entre dois pontos. Para medir a distância entre dois pontos bidimensionais, temos um ponto $P = (p_x, p_y)$ e um ponto $Q = (q_x, q_y)$, tal que a distância $D(P, Q)$ é dada por:

$$D(P, Q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$$

Essa heurística é mais próxima da distância real em ambientes contínuos, mas em um labirinto com movimentos restritos a quatro direções, pode subestimar o custo real. Ainda assim, continua sendo admissível e consistente. Sua principal vantagem é fornecer uma estimativa mais precisa da proximidade entre estados, o que pode reduzir o tempo de busca [3].

Resultados e Conclusões

Com base nas implementações feitas em código [1], foi possível obter os resultados com base nos parâmetros de execução utilizados no trabalho. Eles podem ser vistos nas tabelas a seguir:

Algoritmo	**Heurística**	**Tempo (ms)**	**Expandidos**	**Gerados**	**Pico de Memória**
BFS	-	0.070	22	22	25
DFS	-	0.057	18	20	25
A*	Manhattan	0.108	20	20	20
A*	Euclideana	0.084	20	20	20
Greedy	Manhattan	0.052	14	16	16
Greedy	Euclideana	0.049	14	16	16

Tabela 1: Tabela dos principais resultados

Algoritmo	Heurística	Completo	Ótimo	Custo	Caminho
BFS	-	sim	sim	10	U(S) ->V ->Q ->L ->M ->N ->I ->H ->C ->D ->E(G)
DFS	-	sim	sim	10	U(S) ->V ->Q ->L ->M ->N ->I ->H ->C ->D ->E(G)
A*	Manhattan	sim	sim	10	U(S) ->V ->Q ->L ->M ->N ->I ->H ->C ->D ->E(G)
A*	Euclideana	sim	sim	10	U(S) ->V ->Q ->L ->M ->N ->I ->H ->C ->D ->E(G)
Greedy	Manhattan	sim	sim	10	U(S) ->V ->Q ->L ->M ->N ->I ->H ->C ->D ->E(G)
Greedy	Euclideana	sim	sim	10	U(S) ->V ->Q ->L ->M ->N ->I ->H ->C ->D ->E(G)

Tabela 2: Tabela dos caminhos por algoritmo

Vale destacar que, conforme apresentado na Tabela 2, todos os algoritmos chegaram à mesma solução, o que pode ser observado nas colunas de **Custo** e **Caminho**. O percurso é descrito por uma sequência de letras que correspondem às posições do labirinto ilustrado na Figura 1.

A seguir, são apresentados os gráficos que mostram de forma organizada as métricas coletadas, facilitando a comparação entre os diferentes algoritmos.

Análise dos tempos dos algoritmos

A seguir, observa-se, na Figura 2, a comparação do tempo de execução dos métodos. Essa métrica representa o intervalo total entre o início e o fim da busca, sendo medido em milisegundos, indicando a eficiência temporal do algoritmo.

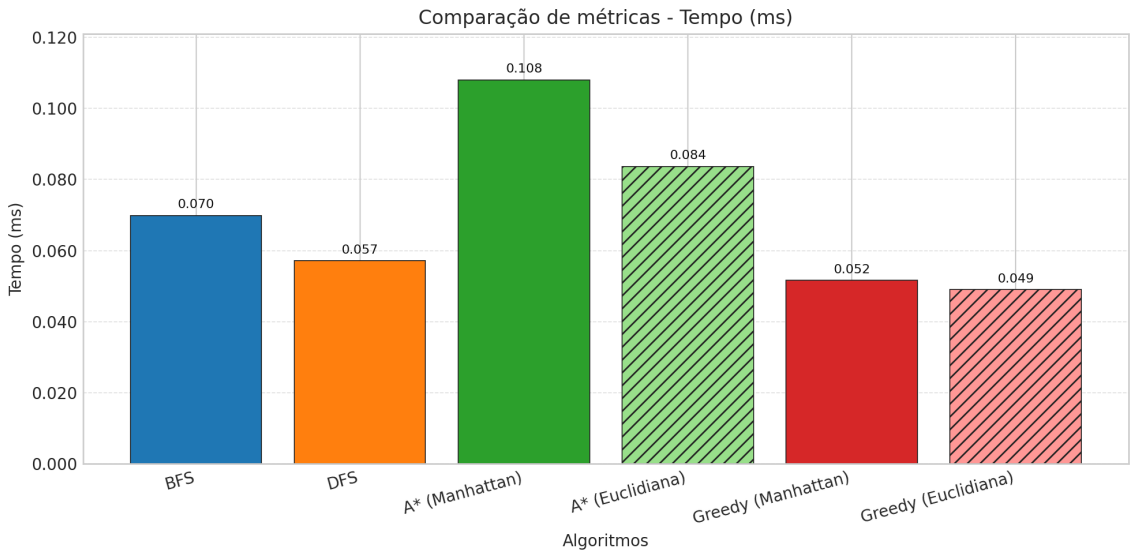

Figura 2: Gráfico comparativo dos tempos em ms

Não informados: O algoritmo DFS apresentou menor tempo de execução. Isso se deve ao fato de que, no labirinto em questão, o objetivo estava posicionado de forma a favorecer a exploração em profundidade: ao seguir um ramo até o fim, o DFS encontrou a solução mais rapidamente. Já o BFS, por sua natureza sistemática, percorreu mais níveis antes de alcançar o objetivo, o que aumentou o tempo.

Informados: A busca gulosa foi mais rápida que o A*, pois toma decisões locais baseadas apenas na heurística, sem acumular custos de caminho. Essa simplicidade reduz o tempo de processamento. Dentro desse grupo, a heurística euclidiana foi ligeiramente mais eficiente que a Manhattan, já que fornece uma estimativa mais próxima da distância real em um espaço bidimensional.

Análise do número de nós expandidos

A Figura 3 mostra a comparação dos nós expandidos de cada algoritmo. Essa métrica correspode à quantidade de nós cujo conteúdo foi processado, refletindo o esforço computacional efetivo usado durante a busca.

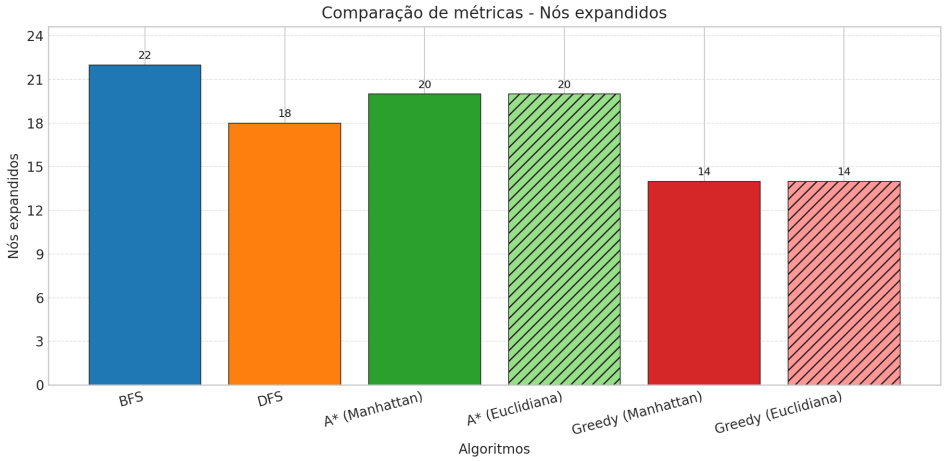

Figura 3: Gráfico comparativo das expansões

Não informados: O BFS expandiu mais nós que o DFS, o que é esperado, pois precisa explorar todos os vizinhos de cada camada antes de avançar. O DFS, ao seguir um único ramo até o fim, expandiu menos nós, embora em outros cenários isso pudesse levá-lo a caminhos muito longos sem solução.

Informados: O Greedy expandiu menos nós que o A*, justamente por não considerar o custo acumulado. Ele segue diretamente em direção ao objetivo, minimizando expansões. O A*, por outro lado, precisa equilibrar custo real e heurística, o que aumenta o número de expansões. Aqui, a escolha da heurística não alterou o número de expansões.

Análise do número de nós gerados

Observa-se na Figura 4 o gráfico comparativo em relação aos nós gerados por cada método. Está métrica representa o número total de novos nós descobertos e inseridos na fronteira de busca, indicando a quantidade de possibilidades consideradas.

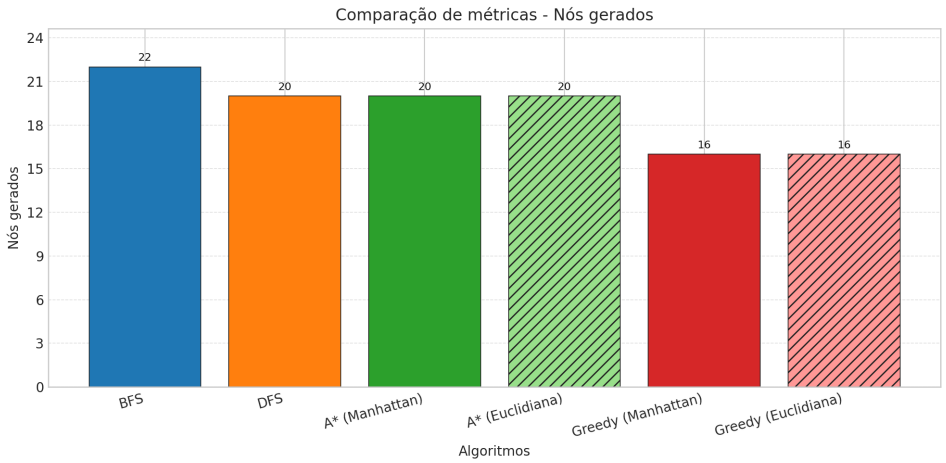

Figura 4: Gráfico comparativo das gerações

Não informados: O BFS gerou mais nós que o DFS, reflexo direto de sua estratégia de explorar camada por camada. Isso significa que, mesmo que não expanda todos esses nós imediatamente, ele precisa armazená-los na fronteira, aumentando o esforço computacional. O DFS, ao focar em um único caminho por vez, gera menos nós no processo.

Informados: O Greedy também gerou menos nós que o A*, confirmando sua eficiência em termos de custo computacional. O A*, ao considerar tanto $g(n)$ quanto $h(n)$, precisa gerar e avaliar mais possibilidades antes de decidir o próximo passo. Assim como nas expansões, a heurística não teve impacto relevante no número de nós gerados.

Análise dos picos de memória

Na Figura 5 tem-se o gráfico com o pico de memória de cada método. Essa métrica indica o maior número de elementos simultaneamente armazenados nas estruturas de controle da busca, servindo para avaliar o consumo de espaço durante a execução.

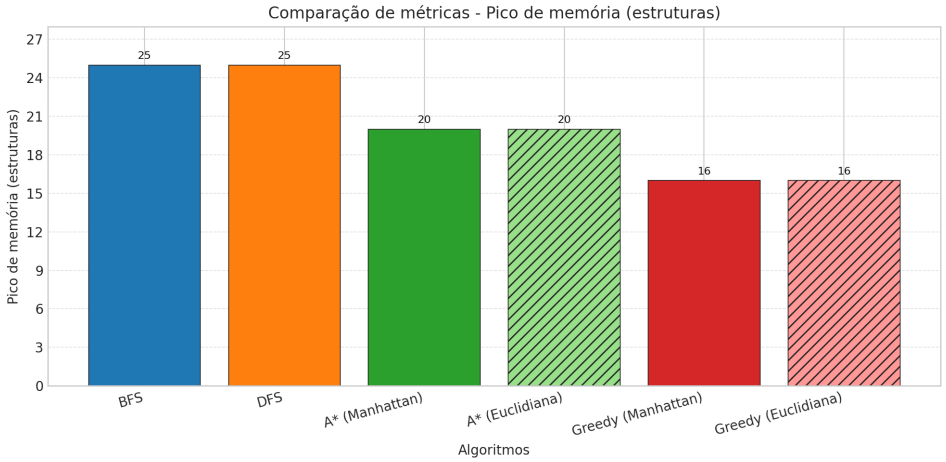

Figura 5: Gráfico comparativo dos picos de memória

Não informados: O BFS apresentou o maior pico de memória, já que precisa manter em sua fila todos os nós de uma camada antes de avançar para a próxima. O DFS, embora também tenha um consumo elevado, tende a ser mais econômico, pois armazena apenas o caminho atual e alguns ramos alternativos.

Informados: O Greedy foi o mais econômico em memória, pois mantém apenas os nós mais promissores segundo a heurística. O A*, por sua vez, exige mais espaço, já que precisa armazenar tanto os custos acumulados quanto as estimativas heurísticas, além de manter múltiplos caminhos possíveis para garantir a solução ótima. Ainda assim, ambos os informados foram mais econômicos que os não informados.

Custo de Caminho

Na Figura 6 observa-se o gráfico com o custo do caminho dos métodos. Esse custo representa o número de passos necessários para alcançar o objetivo.

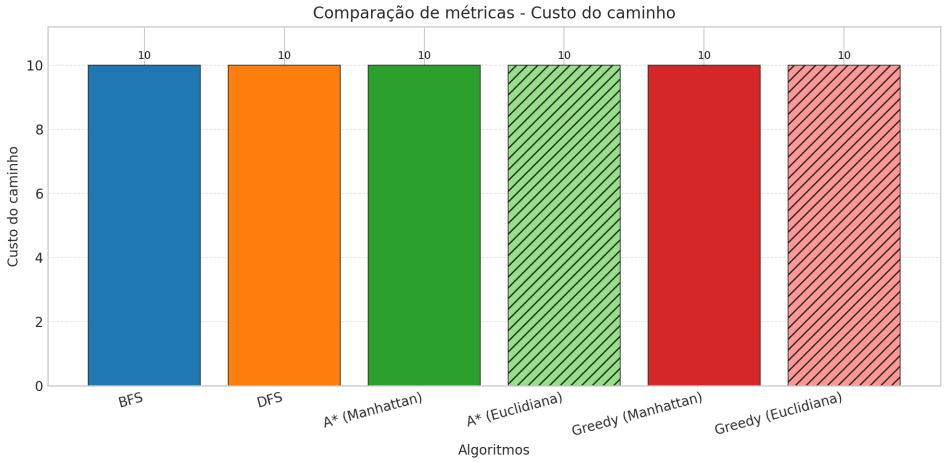

Figura 6: Gráfico comparativo dos custos de caminhos

Em relação ao custo dos caminhos, todos os algoritmos analisados — tanto os informados quanto os não informados — encontraram soluções com o mesmo valor total, igual a 10. Esse resultado indica que, independentemente da estratégia de busca utilizada, o labirinto em questão possui um caminho ótimo bem definido, que foi identificado por todos os métodos.

Conclusão da Análise

Entre os algoritmos não informados, o DFS se mostrou mais eficiente em tempo e memória que o BFS, embora ambos tenham encontrado a solução ótima. Entre os informados, o Greedy foi mais rápido e econômico, mas o A* manteve a eficiência, sendo mais robusto e garantido em cenários gerais. A heurística euclidiana trouxe pequenas vantagens de tempo, mas não alterou significativamente expansões, gerações ou memória.

No que diz respeito ao custo dos caminhos, como dito anteriormente, todos os algoritmos encontraram a mesma solução com mesmo custo. Isso evidencia que, embora as estratégias de busca apresentem desempenhos distintos em termos de eficiência, todas foram capazes de convergir para o mesmo resultado final. Assim, a principal diferença entre os métodos não esteve na qualidade da solução encontrada, mas sim na forma como cada um percorreu o espaço de busca até alcançá-la.

Créditos e Declaração de Autoria

Autores e Papéis

O trabalho foi desenvolvido pelos seguintes integrantes da equipe:

- **Pedro Augusto Gontijo Moura**: responsável pela implementação dos algoritmos de busca, coleta de métricas e desenvolvimento do relatório.
- **Samuel Silva Gomes**: responsável pela coleta de métricas, análise dos resultados, elaboração de gráficos e desenvolvimento do relatório.

Uso de Inteligência Artificial

Ferramentas de Inteligência Artificial, como o ChatGPT (OpenAI), foram utilizadas apenas para auxílio básico na redação do texto, revisão de formatação e comentários, e esclarecimento de dúvidas conceituais. Nenhum trecho de código foi gerado automaticamente ou copiado de fontes externas.

Declaração de Autoria

Declara-se que este trabalho foi integralmente desenvolvido pelos autores listados, respeitando as diretrizes acadêmicas da disciplina e as políticas de integridade do curso. Todos os resultados, análises e códigos aqui apresentados refletem o entendimento e o esforço dos autores.

Referências Bibliográficas

[1] GOMES, S. S; MOURA, P. A. **IA-Busca-8Rainhas**. Disponível em: https://github.com/PedroAugusto08/IA-Busca-8Rainhas.git. Acesso em out. 2025

[2] A. Thiago, **Capítulo 4: Estruturas e Estratégias de Busca**, Sistema Integrado de Gestão de Atividades Acadêmicas. CEFET-MG. Acesso em out. 2025.

[3] RUSSELL, Stuart J.; NORVIG, Peter. Artificial Intelligence: A Modern Approach. 4. ed. Hoboken: Pearson, 2021.