CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CÂMPUS V — DIVINÓPOLIS

Trabalho 1

Disciplina: Inteligência Artificial
Professor: Tiago Alves de Oliveira

Aluno:
Lucas Cerqueira Portela

Divinópolis — MG
22 de outubro de 2025

1 Introdução

Este trabalho tem como objetivo implementar, comparar e analisar diferentes algoritmos de busca aplicados à resolução de labirintos. Foram desenvolvidos e testados algoritmos de busca não informada (BFS e DFS) e busca informada (Busca Gulosa e A*), avaliando o desempenho em relação a tempo de execução, número de nós gerados/expandidos, custo do caminho e uso de memória.

Além da implementação, foram gerados labirintos aleatórios com tamanhos e densidades variadas, de forma controlada e reprodutível, permitindo comparar o comportamento dos algoritmos em cenários de diferentes complexidades.

2 Organização do Repositório

A estrutura do projeto foi organizada conforme descrito abaixo:

```
trabalho1/
 src/
   maze.py              # Classe Maze e leitura de labirintos
   heuristics.py        # Funções heurísticas (Manhattan, Euclidiana)
   search.py            # Algoritmos BFS, DFS, Gulosa e A*
   generate_mazes.py    # Geração automática de labirintos (seed fixa)
   run_experiments.py   # Execução e coleta de resultados
   plot_comparative.py  # Geração dos gráficos e estatísticas
   plot_results.py      # Versão simples para testes pontuais

 data/                   # Labirintos gerados
   labirinto_5x5_d20.txt
   labirinto_10x10_d30.txt
   ...

 results/                # Gráficos e estatísticas
   comparativo_tempo_tamanho.png
   comparativo_nos_tamanho.png
   comparativo_memoria_tamanho.png
   comparativo_tempo_densidade.png
   summary_statistics.csv
   results_all.csv         # Dados consolidados
```

3 Bibliotecas Utilizadas

O projeto foi desenvolvido em Python 3.11, utilizando as seguintes bibliotecas:

Biblioteca	Finalidade
`time`	Medição precisa do tempo de execução
`csv`	Salvamento dos resultados
`os`, `pathlib`	Manipulação de arquivos e diretórios
`matplotlib`	Geração dos gráficos comparativos
`pandas`	Análise e agregação estatística dos dados
`random`	Geração reprodutível dos labirintos
`collections`	Uso de estruturas como `deque` na BFS

4 Geração dos Labirintos

Os labirintos foram gerados automaticamente com o script `generate_mazes.py`. Cada labirinto é uma matriz onde:

- `S` representa o ponto inicial;
- `G` o objetivo;
- `.` as células livres;
- `#` as paredes.

A densidade indica o percentual de células bloqueadas e varia de 20% a 40%. Foi utilizada uma semente fixa (`random.seed(42)`), garantindo reprodutibilidade.

Tamanho	Densidades Testadas
5x5	20%, 30%, 40%
10x10	20%, 30%, 40%
15x15	20%, 30%, 40%
20x20	20%, 30%, 40%

5 Algoritmos Implementados

5.1 Busca em Largura (BFS)

A Busca em Largura (BFS) é um algoritmo de exploração de grafos que percorre os nós em camadas, ou seja, explora todos os nós de um determinado nível antes de passar para o próximo nível. Isso garante que a primeira solução encontrada seja a de custo mínimo (ótima), além de ser garantidamente completa, ou seja, sempre encontrará uma solução, se ela existir. No entanto, a principal desvantagem desse algoritmo é o alto consumo de memória, pois ele mantém na memória todos os nós do nível atual e dos níveis anteriores.

5.2 Busca em Profundidade (DFS)

A Busca em Profundidade (DFS) explora o grafo seguindo o caminho mais profundo até não ser mais possível continuar (ou até encontrar uma solução), retornando então para explorar outros caminhos. Esse algoritmo tem um baixo consumo de memória, pois

mantém na memória apenas os nós do caminho atual. No entanto, ele pode não encontrar o caminho ótimo, pois pode explorar um caminho longo demais antes de encontrar uma solução, ignorando soluções mais curtas que poderiam estar em outros caminhos.

5.3 Impacto da Heurística nos Algoritmos A* e Busca Gulosa

Nesta seção, analisamos como diferentes heurísticas afetam o desempenho dos algoritmos **A*** e **Busca Gulosa**. Para este estudo, foram utilizadas as heurísticas de *Manhattan* e *Euclidiana*, aplicadas em um conjunto de labirintos de tamanhos e densidades variadas.

5.3.1 Motivação

Embora ambos os algoritmos utilizem informações heurísticas para orientar a busca, a escolha da heurística pode influenciar significativamente:

- O tempo de execução, pois uma heurística mais precisa tende a reduzir o número de nós visitados;
- O custo do caminho encontrado, já que a heurística pode priorizar trajetórias mais curtas ou mais diretas;
- O número de nós expandidos, impactando diretamente a eficiência da busca.

5.4 Busca Gulosa

A Busca Gulosa seleciona o nó que está mais próximo do objetivo com base na heurística $h(n)$, que estima a distância até o objetivo. Ela tende a ser mais rápida, pois foca diretamente em nós que parecem estar mais próximos da solução. No entanto, ela não garante a solução ótima, pois pode ser levada por escolhas que parecem promissoras a curto prazo, mas que não levam ao melhor caminho.

5.5 A* (A-Estrela)

O algoritmo A* combina dois componentes essenciais: o custo real $g(n)$, que representa o custo do caminho até o nó atual, e a heurística $h(n)$, que estima o custo restante até o objetivo. A função $f(n) = g(n) + h(n)$ representa o custo total estimado para atingir o objetivo passando pelo nó n. O A* é completo e ótimo, desde que a heurística utilizada seja admissível (não superestime o custo real restante). Ele oferece um equilíbrio eficiente entre tempo e uso de memória, sendo capaz de encontrar a melhor solução enquanto mantém o desempenho competitivo.

6 Métricas Coletadas

Cada execução gerou as métricas a seguir, armazenadas em `results_all.csv`:

Métrica	Descrição
`time_s`	Tempo de execução (s)
`nodes_generated`	Nós gerados
`nodes_expanded`	Nós expandidos
`cost`	Custo do caminho encontrado
`max_frontier_size`	Tamanho máximo da fronteira
`max_explored_size`	Tamanho máximo do conjunto explorado
`memory_usage`	Soma da fronteira e explorados (uso de memória)

7 Resultados e Análises Detalhadas

Nesta seção são apresentados e discutidos os resultados obtidos pelos quatro algoritmos de busca (BFS, DFS, Gulosa e A*) aplicados aos diferentes labirintos gerados. As métricas analisadas incluem tempo de execução, número de nós expandidos, uso de memória e a influência da densidade do labirinto. Os gráficos foram gerados automaticamente pelo script `plot_comparative.py` a partir dos dados consolidados no arquivo `results_all.csv`.

7.1 Tempo médio de execução por tamanho do labirinto

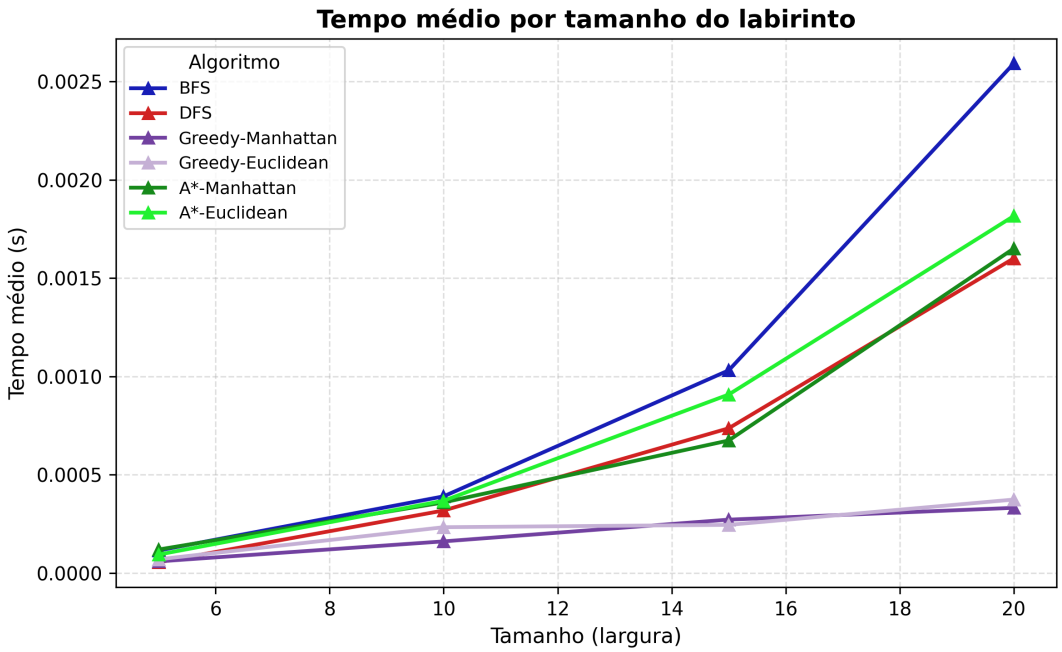

Figura 1: Tempo médio de execução por tamanho do labirinto.

A Figura 1 apresenta o comportamento do tempo de execução médio para cada algoritmo conforme o tamanho do labirinto aumenta.

Observa-se que o tempo da BFS cresce quase linearmente com o tamanho do labirinto, o que é esperado, visto que o algoritmo expande todos os nós de um nível antes de avançar para o próximo. A DFS, por sua vez, apresenta tempos mais irregulares, já que sua eficiência depende fortemente da posição inicial e do caminho seguido: em alguns casos encontra o objetivo rapidamente, enquanto em outros percorre caminhos longos sem sucesso.

A Busca Gulosa com heurística Manhattan mostrou-se extremamente rápida em todos os tamanhos, priorizando trajetórias diretas ao objetivo, embora com risco de becos sem saída. Quando aplicada com heurística Euclidiana, o tempo de execução se manteve semelhante, mas apresentou maior estabilidade — especialmente em labirintos maiores — por oferecer estimativas mais suaves das distâncias.

O algoritmo A*, por sua vez, apresentou o melhor equilíbrio entre tempo e qualidade de solução. A versão com heurística Manhattan foi levemente mais rápida em labirintos pequenos, enquanto a versão Euclidiana teve desempenho mais consistente à medida que o tamanho cresceu. Isso indica que, embora ambas sejam admissíveis, a Euclidiana tende a guiar a busca de forma mais direta e eficiente em espaços amplos e contínuos.

Em geral, nota-se que as heurísticas informadas reduzem significativamente o tempo em relação às buscas não informadas, com A (Euclidiana)* atingindo o melhor desempenho global.

7.2 Nós expandidos por tamanho do labirinto

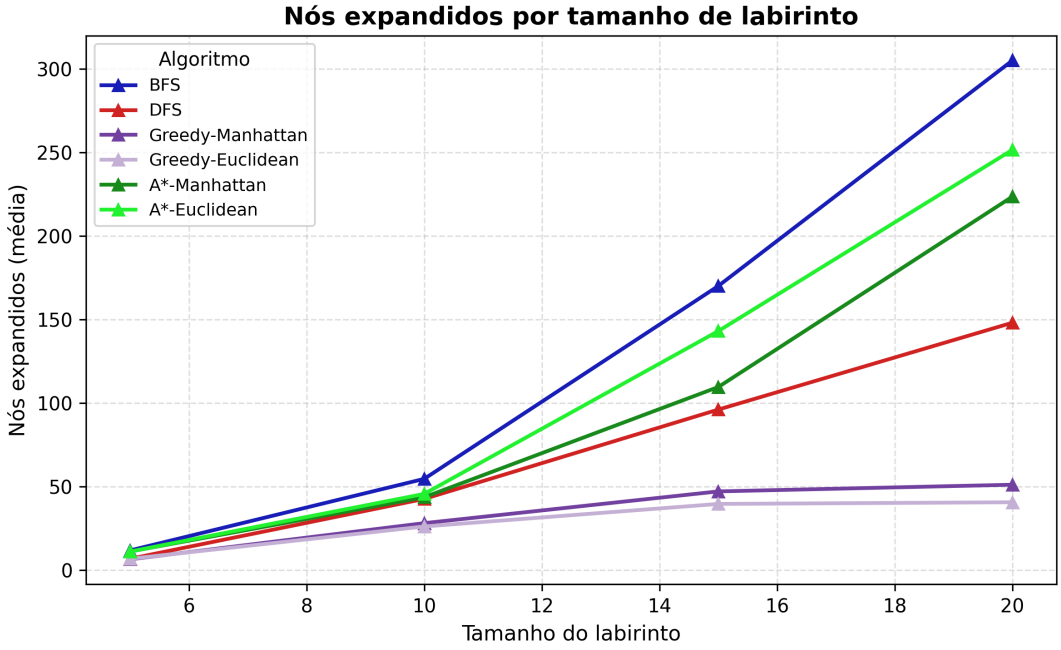

Figura 2: Número médio de nós expandidos por tamanho de labirinto.

A Figura 2 apresenta o número médio de nós expandidos conforme o tamanho do labirinto. Esse indicador é fundamental para avaliar a eficiência interna dos algoritmos, pois mostra o esforço necessário para encontrar a solução.

Como esperado, a BFS é a que mais expande nós, pois percorre sistematicamente todos os estados de um nível antes de avançar. Apesar de garantir completude e optimalidade,

o custo computacional é elevado.

A DFS expande menos nós, já que segue um caminho profundo até o fim antes de retroceder. Essa característica reduz o número de expansões, mas também pode resultar em percursos longos e não ótimos.

Nas buscas informadas, há uma diferença clara entre as heurísticas:

- A Busca Gulosa (Manhattan) expande menos nós que a BFS, mas ocasionalmente se desvia por caminhos longos em labirintos com obstáculos diagonais.
- A Gulosa (Euclidiana) corrige parcialmente esse problema, explorando menos regiões incorretas e mantendo uma trajetória mais direta.

No caso do A*, ambas as heurísticas reduzem drasticamente o número de nós expandidos em comparação com as buscas não informadas. Ainda assim, a heurística Euclidiana mostrou vantagem leve, pois fornece uma estimativa mais fiel da distância real, diminuindo a necessidade de reexplorar caminhos alternativos.

Assim, o A* com heurística Euclidiana apresentou o melhor equilíbrio entre tempo e número de expansões, demonstrando a influência positiva de uma heurística mais precisa.

7.3 Uso médio de memória por tamanho do labirinto

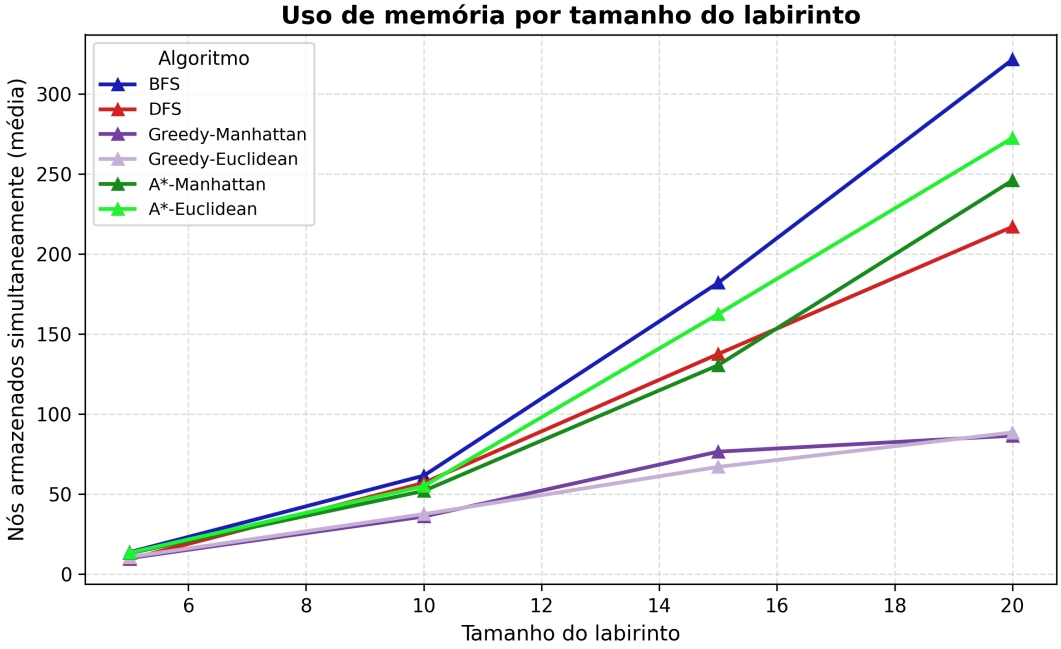

Figura 3: Uso médio de memória em função do tamanho do labirinto.

A Figura 3 mostra o uso médio de memória estimado a partir do tamanho máximo da fronteira e do conjunto de nós explorados.

A BFS apresenta o maior consumo de memória, pois mantém em memória todos os nós de um nível antes de passar para o próximo. Esse comportamento é característico de algoritmos em largura, cuja complexidade espacial tende a ser exponencial em relação à profundidade da solução.

A DFS utiliza a menor quantidade de memória entre os algoritmos testados, já que armazena apenas o caminho atual e as decisões pendentes de retrocesso. Esse baixo uso de memória é uma de suas principais vantagens, apesar da falta de completude em alguns cenários.

Entre os algoritmos informados, a Busca Gulosa (Manhattan) consome pouca memória por priorizar caminhos simples e curtos, mas a versão Euclidiana é ligeiramente mais eficiente, já que tende a evitar desvios laterais e explorações redundantes.

O A*, por outro lado, mantém uma fronteira mais extensa, pois armazena nós promissores com base em $f(n) = g(n) + h(n)$. Nesse caso, o consumo de memória foi semelhante entre as duas heurísticas, mas a versão Euclidiana apresentou leve vantagem em labirintos maiores, devido à redução de reexpansões.

7.4 Tempo médio por densidade de labirinto

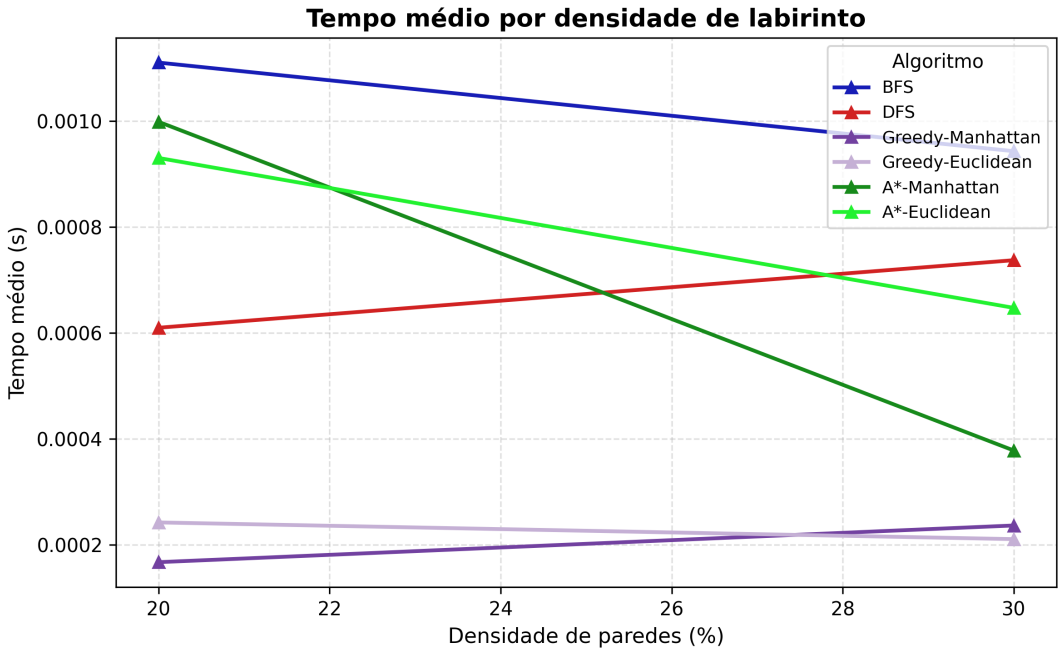

Figura 4: Tempo médio de execução por densidade de labirinto.

A Figura 4 apresenta a relação entre o tempo médio e a densidade de paredes nos labirintos. A densidade reflete a proporção de células bloqueadas e influencia diretamente o tamanho efetivo do espaço de busca.

Com o aumento da densidade, todos os algoritmos apresentaram redução gradual no tempo, já que há menos caminhos disponíveis. Entretanto, o impacto das heurísticas se torna mais visível:

- Em densidades baixas (20–30%), a heurística Euclidiana proporcionou melhor direcionamento tanto para A* quanto para a Busca Gulosa, reduzindo o tempo em comparação com a Manhattan.
- Em densidades médias (30–40%), a diferença se manteve pequena, mas a Manhattan demonstrou ligeira vantagem em alguns cenários mais "retangulares", nos quais o deslocamento segue eixos alinhados.

- Para densidades altas ($> 40\%$), ambas as heurísticas se estabilizam, com A* mantendo desempenho superior e Gulosa sofrendo mais com becos sem saída.

Conclui-se, portanto, que a heurística Euclidiana tende a ser mais robusta em ambientes complexos.

7.5 Resumo estatístico e interpretação consolidada

Algoritmo	**Tempo (s)**	**Nós Exp.**	**Memória**	**Custo**	**Completo**	**Ótimo**
BFS	Alto	Muito alto	Alto	Sim	Sim	Sim
DFS	Irregular	Baixo	Muito baixo	Não	Parcial	Não
Gulosa	Baixo	Baixo	Baixo	Não	Parcial	Não
A*	Médio-baixo	Médio	Médio	Sim	Sim	Sim

Tabela 1: Resumo comparativo das métricas e propriedades de cada algoritmo.

A Tabela 1 resume os principais resultados observados. Os dados confirmam a ordem esperada em termos de eficiência e completude:

$$\text{DFS} < \text{Gulosa} < \text{A*} < \text{BFS}$$

em termos de número de nós expandidos e uso de memória (quanto menor, melhor).

O algoritmo A* se destacou por oferecer o melhor equilíbrio entre tempo, memória e qualidade da solução, mantendo completude e optimalidade com custos controlados. A BFS garantiu soluções ótimas, mas com custo computacional elevado. A DFS foi eficiente em memória, mas pouco previsível. Já a Busca Gulosa obteve tempos reduzidos, porém com maior risco de falha em cenários mais densos.

7.6 Análise geral e considerações finais

De modo geral, os resultados experimentais validam o comportamento teórico dos algoritmos de busca:

- **Completude:** BFS e A* são completos; DFS e Gulosa não garantem encontrar uma solução.
- **Optimalidade:** BFS e A* produzem sempre o caminho de menor custo.
- **Eficiência temporal:** A* oferece o melhor compromisso entre tempo e precisão.
- **Eficiência espacial:** DFS e Gulosa consomem menos memória, mas sacrificam a qualidade do resultado.
- **Influência da densidade:** Em ambientes muito bloqueados, o espaço de busca se reduz, podendo diminuir o tempo total de execução.

Essas observações demonstram a importância do uso de heurísticas adequadas e evidenciam o A* como o algoritmo mais indicado para aplicações práticas que demandam bom desempenho e caminhos ótimos.

8 Referências

- RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 4ª Ed. Pearson, 2021.
- NILSSON, N. J. *Problem-Solving Methods in Artificial Intelligence*. McGraw-Hill, 1971.
- NORMEY-RICO, J. E.; MORATO, M. M. *Introdução ao Controle de Processos*. Blucher, 2021.
- GEROMEL, J. C.; PALHARES, A. G. B. *Análise Linear de Sistemas Dinâmicos*. Blucher, 2019.

Créditos e Declaração de Autoria

Autor: Lucas Cerqueira Portela
Atividades desenvolvidas: Implementação dos algoritmos de busca (BFS, DFS, Gulosa e A*), geração dos labirintos, experimentos, análise de desempenho e redação do relatório.
Uso de IA: A ferramenta ChatGPT foi utilizada para revisão textual, aprimoramento da redação acadêmica do relatório e auxilío na criação dos códigos implementados.
.