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

Relatório do Trabalho 01:
Busca no Labirinto (Não Informada e Informada)

Matheus Emanuel da Silva

Divinópolis - MG
Outubro de 2025

Sumário

Fundamentos Teóricos

Problemas de Busca e Seus Componentes

No campo da Inteligência Artificial, um problema de busca é um modelo formal para a resolução de problemas onde o objetivo é encontrar uma sequência de ações que leve de um estado inicial a um estado final desejado. Formalmente, um problema é definido por um conjunto de componentes essenciais: o espaço de estados, o estado inicial, as ações (ou operadores) e a função de teste de objetivo. A solução para tal problema é um caminho, ou uma sequência de ações, através do espaço de estados, do início ao fim.

Estratégias de Busca

Algoritmos de busca são os métodos empregados para explorar o espaço de estados e encontrar uma solução. Eles podem ser classificados em duas categorias principais, com base na informação que utilizam para guiar a busca: busca não informada e busca informada.

Busca Não Informada (Busca Cega)

Os algoritmos de busca não informada, também conhecidos como busca cega, operam sem qualquer conhecimento adicional sobre o problema além da sua própria definição. Eles não possuem uma estimativa da distância ou do custo para atingir o objetivo. A busca é, portanto, sistemática, explorando o espaço de estados de maneira predeterminada.

Para este trabalho, os algoritmos de busca não informada selecionados para análise foram o *Breadth-First Search (BFS)* e o *Depth-First Search (DFS)*.

Busca Informada (Busca Heurística)

Em contraste, os algoritmos de busca informada utilizam conhecimento específico do domínio do problema para otimizar o processo de busca. Essa informação é encapsulada em uma **função heurística**, denotada como $h(n)$, que estima o custo de se chegar do estado n até o estado objetivo. Ao utilizar essa heurística, o algoritmo pode tomar decisões mais inteligentes sobre qual nó expandir em seguida, geralmente resultando em uma busca significativamente mais eficiente.

Métricas de Avaliação de Desempenho

Para comparar a eficiência dos diferentes algoritmos de busca, este trabalho utilizará um conjunto de métricas de avaliação consolidadas na literatura. A análise será conduzida sobre uma

mesma instância de problema para garantir uma comparação justa. As métricas são:

- **Completude:** Avalia se o algoritmo garante encontrar uma solução, caso uma exista.
- **Optimalidade:** Verifica se a solução encontrada pelo algoritmo possui o menor custo de caminho possível.
- **Tempo de Execução:** Embora dependente do hardware, esta métrica será registrada para fornecer uma referência do custo computacional direto.
- **Complexidade Espacial (Memória):** Para uma análise independente de hardware, a memória será medida como o número máximo de nós mantidos simultaneamente nas estruturas de dados da busca. Isso inclui os nós na *fronteira* de exploração (fila, pilha ou heap) somados aos nós no conjunto de *explorados*[cite: 40, 261, 268].
- **Nós Expandidos:** Corresponde ao número total de nós que foram removidos da fronteira para que seus sucessores fossem gerados[cite: 41]. Esta métrica serve como um bom indicador do esforço computacional total do algoritmo.

Além dessas métricas, uma consideração metodológica importante foi adotada para a medição do tempo de execução. Todas as coletas de tempo foram realizadas a partir da média de 10 execuções com **cache frio** (”cold cache”). Esta abordagem busca garantir uma avaliação mais estável e representativa[cite: 52], minimizando a influência de dados ou instruções de execuções anteriores que possam estar armazenados na memória cache, o que poderia beneficiar artificialmente medições subsequentes (conhecidas como ”warm cache”).

Escopo do Estudo

O objetivo deste trabalho é implementar e comparar o desempenho de algoritmos representativos de cada categoria de busca. Para a busca não informada, serão analisados o *Breadth-First Search (BFS)* e o *Depth-First Search (DFS)*. Para a busca informada, os algoritmos estudados serão o Greedy Best-First Search e o A* Search. A comparação será realizada com base nas métricas definidas na Seção Métricas .

Parte I - Busca não informada

Nesta seção, são analisados e comparados os algoritmos de Busca em Largura (BFS) e Busca em Profundidade (DFS) para a resolução de um problema de busca em um grafo, partindo de um nó inicial 'S' até um nó objetivo 'G'.

Análise de Desempenho dos Algoritmos

As métricas de desempenho para cada algoritmo foram coletadas a partir da média de 10 execuções com cache frio ("cold cache") para garantir uma avaliação mais estável e representativa.

Algoritmo BFS (Busca em Largura)

O BFS explora o grafo em camadas, garantindo a descoberta do caminho mais curto em grafos com arestas de custo uniforme. Os resultados médios obtidos foram:

- **Tempo de execução:** Desprezível (< 0.001s).
- **Memória utilizada (média):** 1.20 KB (Desvio Padrão: 1.83 KB).
- **Máx. elementos nas estruturas (média):** 67. (Refere-se ao pico de $|fronteira| + |visitados|$).
- **Completude:** Sim. O algoritmo foi capaz de encontrar uma solução.
- **Optimalidade:** Sim. O caminho encontrado, com 16 passos (17 nós), é o mais curto possível.
- **Nós expandidos (média):** 64.

O caminho ótimo encontrado pelo BFS foi:

```
(8,0) -> (7,0) -> (6,0) -> (5,0) -> (5,1) -> (5,2) -> (4,2) ->
(4,3) -> (3,3) -> (2,3) -> (1,3) -> (1,4) -> (0,4) -> (0,5) ->
(0,6) -> (0,7) -> (0,8)
```

Algoritmo DFS (Busca em Profundidade)

O DFS explora o grafo aprofundando-se em um único ramo até atingir um nó terminal ou o objetivo. Os resultados médios coletados foram:

- **Tempo de execução (média):** 0.000072s (Desvio Padrão: 0.000146s).

- **Memória utilizada (média):** 1.60 KB (Desvio Padrão: 1.96 KB).
- **Máx. elementos nas estruturas (média):** 71. (Refere-se ao pico de $|fronteira| + |visitados|$).
- **Completude:** Sim. O algoritmo encontrou uma solução.
- **Optimalidade:** Não. O caminho encontrado, com 26 passos (27 nós), não é o mais curto.
- **Nós expandidos (média):** 42.

O caminho encontrado pelo DFS foi:

```
(8,0) -> (8,1) -> (8,2) -> (8,3) -> (8,4) -> (8,5) -> (8,6) ->
(8,7) -> (8,8) -> (7,8) -> (6,8) -> (6,7) -> (6,6) -> (6,5) ->
(6,4) -> (5,4) -> (5,3) -> (4,3) -> (3,3) -> (2,3) -> (2,4) ->
(2,5) -> (1,5) -> (0,5) -> (0,6) -> (0,7) -> (0,8)
```

Comparativo entre BFS e DFS

Para facilitar a análise, os resultados médios são consolidados na Tabela 1.

Tabela 1 – Comparativo de Métricas Médias (10 execuções) entre BFS e DFS.

Métrica	**BFS**	**DFS**
Tempo de Execução (média)	Desprezível	0.000072s
Memória Utilizada (média)	1.20 KB	1.60 KB
Máx. Elementos Estruturas	**67**	71
Nós Expandidos (média)	64	**42**
Comprimento do Caminho	**16 passos**	26 passos
Ótimo?	**Sim**	Não

A análise comparativa dos dados médios revela as características de cada algoritmo. O tempo de execução para ambos foi irrisório, embora o BFS tenha registrado uma média efetivamente nula (0.000000s) contra a pequena média mensurável do DFS (0.000072s).

O **DFS** novamente se mostrou mais eficiente na exploração, expandindo, em média, apenas 42 nós, contra 64 do BFS. Isso se deve à sua natureza de aprofundar em um único ramo, o que o levou rapidamente a *uma* solução.

Contudo, essa eficiência na exploração teve o custo da **optimalidade**. O DFS encontrou um caminho longo e subótimo de 26 passos. O **BFS**, por sua vez, ao explorar em camadas, garantiu a descoberta do caminho mais curto possível (16 passos), confirmando sua optimalidade.

A análise da **complexidade espacial** é a mais relevante. O **BFS** foi mais eficiente em *ambas* as métricas de memória:

- **Métrica Teórica (Elementos):** Alinhado à definição de complexidade espacial do Capítulo , o BFS manteve um pico de 67 elementos em suas estruturas, contra 71 do DFS.
- **Métrica de Hardware (KB):** O BFS também utilizou, em média, menos memória física (1.20 KB) que o DFS (1.60 KB).

Este resultado experimental contradiz a expectativa teórica comum, onde o BFS ($O(b^d)$) é frequentemente citado como mais intensivo em memória que o DFS ($O(bm)$). Neste problema, a profundidade do caminho longo encontrado pelo DFS (armazenando o caminho na pilha, somado aos visitados) exigiu mais espaço do que a largura máxima da fronteira do BFS.

Em suma, os resultados confirmam que o BFS é a escolha ideal para garantir a solução ótima e, neste cenário, foi superior em todas as métricas (tempo, memória e optimalidade), exceto no número de nós expandidos.

Parte II - Busca Informada

Neste capítulo, são analisados os algoritmos de busca informada, que utilizam uma função heurística para estimar o custo até o objetivo e guiar a busca de forma mais eficiente. Foram implementados dois algoritmos principais, *Greedy Best-First Search* (Busca Gulosa) e $A*$, cada um avaliado com duas heurísticas distintas: Distância de Manhattan e Distância Euclidiana.

As métricas, assim como no capítulo anterior, são oriundas da média de 10 execuções com cache frio.

Busca Gulosa (Greedy Best-First Search)

A Busca Gulosa é um algoritmo que expande o nó que parece estar mais próximo do objetivo, de acordo com a função heurística $h(n)$. Sua função de avaliação é $f(n) = h(n)$. Embora seja rápido, ele não é completo nem ótimo.

Greedy Search com Heurística Manhattan

- **Tempo de execução (média):** 0.000051s (Desvio Padrão: 0.000152s).
- **Memória utilizada (média):** 1.60 KB (Desvio Padrão: 1.96 KB).
- **Máx. elementos nas estruturas (média):** 37.
- **Nós expandidos (média):** 18.
- **Optimalidade:** Não garantida (mas encontrou o caminho ótimo de 16 passos).

Greedy Search com Heurística Euclidiana

- **Tempo de execução (média):** Desprezível (< 0.001s).
- **Memória utilizada (média):** 1.20 KB (Desvio Padrão: 1.83 KB).
- **Máx. elementos nas estruturas (média):** 49.
- **Nós expandidos (média):** 18.
- **Optimalidade:** Não garantida (mas encontrou um caminho ótimo de 16 passos).

Tabela 2 – Comparativo de Métricas Médias (Greedy Search).

Métrica	**Greedy (Manhattan)**	**Greedy (Euclidean)**
Tempo (média)	0.000051s	**Desprezível**
Memória (média)	1.60 KB	**1.20 KB**
Máx. Elementos Estruturas	**37**	49
Nós Expandidos (média)	**18**	**18**
Comp. Caminho	16 passos	16 passos

Comparativo das Heurísticas (Greedy)

A **Distância Euclidiana** foi marginalmente superior, apresentando tempo de execução nulo e menor consumo médio de memória (KB). Curiosamente, a **Distância de Manhattan** exigiu um pico menor de elementos nas estruturas (37 vs 49). Para a análise subsequente, a versão com **Heurística Euclidiana** será considerada a "campeã"do Greedy, devido ao seu tempo nulo e menor uso de memória (KB).

Busca A*

O algoritmo A* é uma estratégia de busca completa e ótima, desde que a heurística $h(n)$ seja admissível. Sua função de avaliação é $f(n) = g(n) + h(n)$.

A* com Heurística Manhattan

- **Tempo de execução (média):** Desprezível (< 0.001s).
- **Memória utilizada (média):** 0.80 KB (Desvio Padrão: 1.60 KB).
- **Máx. elementos nas estruturas (média):** 60.
- **Nós expandidos (média):** 53.
- **Optimalidade:** Sim (garantida).

A* com Heurística Euclidiana

- **Tempo de execução (média):** 0.000060s (Desvio Padrão: 0.000180s).
- **Memória utilizada (média):** 1.60 KB (Desvio Padrão: 1.96 KB).
- **Máx. elementos nas estruturas (média):** 61.
- **Nós expandidos (média):** 54.
- **Optimalidade:** Sim (garantida).

Comparativo das Heurísticas (A*)

Tabela 3 – Comparativo de Métricas Médias (A* Search).

Métrica	**A* (Manhattan)**	**A* (Euclidean)**
Tempo (média)	**Desprezível**	0.000060s
Memória (média)	**0.80 KB**	1.60 KB
Máx. Elementos Estruturas	**60**	61
Nós Expandidos (média)	**53**	54
Comp. Caminho	16 passos	16 passos

Para o algoritmo A*, a **Distância de Manhattan** provou ser a heurística superior em todas as métricas, guiando a busca de forma mais eficaz. Ela será considerada a versão "campeã"do A*.

Comparativo Greedy e A*)

Após identificar a melhor heurística para cada algoritmo informado, podemos comparar diretamente o desempenho da Busca Gulosa (com heurística Euclidiana) contra o A* (com heurística Manhattan).

Tabela 4 – Comparativo de Métricas Médias (Greedy vs A*).

Métrica	**Greedy (Euclidean)**	**A* (Manhattan)**
Nós Expandidos (média)	**18**	53
Máx. Elementos Estruturas	**37**	60
Tempo (média)	**Desprezível**	**Desprezível**
Memória (média)	1.20 KB	**0.80 KB**
Optimalidade	Sim (Neste caso)	**Sim (Garantido)**

A análise revela uma clara troca (trade-off) entre eficiência e garantia:

- **Eficiência de Exploração (Greedy):** A Busca Gulosa foi drasticamente mais eficiente, expandindo apenas **18 nós** contra 53 do A*. Isso se refletiu no pico de uso das estruturas (37 vs 60), mostrando que o Greedy foi muito mais direto ao seu objetivo.
- **Consumo de Memória (A*):** Curiosamente, o A* (Manhattan) teve um consumo médio de memória física (0.80 KB) inferior ao Greedy (1.20 KB).
- **Optimalidade:** O ponto crucial da comparação. O A* (Manhattan) tem a *garantia* teórica de que encontrou o caminho ótimo. O Greedy, por sua vez, *apenas coincidentemente* encontrou o caminho ótimo neste cenário. Em um mapa mais complexo, o Greedy poderia facilmente ter sido atraído para um caminho subótimo.

Em resumo, dentro das buscas informadas, o Greedy foi o mais rápido e eficiente em termos de exploração, mas o A* é a estratégia mais robusta e confiável por garantir a melhor solução.

Análise Comparativa e Conclusão

Após a apresentação individual dos resultados das buscas não informada (Capítulo) e informada (Capítulo), este capítulo realiza a análise comparativa entre todas as estratégias implementadas, culminando nas conclusões finais do estudo.

Comparativo Geral dos Algoritmos

A Tabela 5 consolida os resultados de todos os algoritmos testados, permitindo uma análise direta da eficácia de cada abordagem.

Tabela 5 – Comparativo de Métricas Médias de Todos os Algoritmos.

Tipo	Algoritmo (Heurística)	Nós Exp. (média)	Máx. Estruturas (média)	Tempo (s)	Memória (KB)	Comp. Caminho (Ótimo?)
Não Informada	BFS	64	67	0.000000	1.20 KB	16 (**Sim**)
Não Informada	DFS	42	71	0.000072	1.60 KB	26 (Não)
Informada	Greedy (Euclidean)	**18**	**37**	0.000000	1.20 KB	16 (**Sim**)
Informada	Greedy (Manhattan)	**18**	49	0.000051	1.60 KB	16 (**Sim**)
Informada	A* (Manhattan)	53	60	0.000000	**0.80 KB**	16 (**Sim, Garantido**)
Informada	A* (Euclidean)	54	61	0.000060	1.60 KB	16 (**Sim, Garantido**)

Conclusões Finais

A análise dos dados consolidados permite extrair as seguintes conclusões sobre o comportamento dos algoritmos neste problema:

1. **Eficácia da Busca Informada:** A introdução de heurísticas (Busca Gulosa e A*) resultou em uma eficiência de exploração drasticamente superior. A Busca Gulosa (ambas heurísticas) foi a campeã em eficiência de exploração, expandindo apenas **18 nós** — menos da metade do DFS (42) e quase quatro vezes menos que o BFS (64).

2. **Greedy vs. A*:** Neste cenário particular, a Busca Gulosa não só foi a mais eficiente, como também encontrou o caminho ótimo. Isso indica que, para este mapa, a heurística é um guia muito preciso que não leva o algoritmo a "becos sem saída"que exigiriam retrocesso. É importante notar, contudo, que esta optimalidade não é uma garantia teórica do Greedy, mas sim um resultado favorável deste experimento.

3. **A* vs. BFS (Batalha dos Ótimos):** O algoritmo A* (com heurística Manhattan) provou ser superior ao BFS. Ambos garantem a optimalidade, mas o A* foi mais inteligente, encontrando o mesmo caminho ótimo (16 passos) com menos esforço: expandiu menos nós (53 vs 64) e manteve menos elementos em suas estruturas (60 vs 67). Além disso, o A* (Manhattan) foi o algoritmo com o menor consumo de memória física (0.80 KB).

4. **O Pior Desempenho:** O DFS, embora eficiente em expandir poucos nós (42), falhou no critério de optimalidade (caminho de 26 passos) e, surpreendentemente, registrou o maior pico de uso das estruturas de dados (71), superando até mesmo o BFS.

Em suma, para este problema específico, a **Busca Gulosa (Greedy Search)** demonstrou ser a estratégia mais rápida e com menor custo de exploração, enquanto o **A* (com heurística Manhattan)** representou o melhor equilíbrio entre eficiência e a garantia teórica de optimalidade, superando o BFS em todas as métricas relevantes.

Créditos e Declaração de Autoria

Autores e Papéis

- **Matheus Emanuel da Silva:** Responsável pela implementação de todos os algoritmos (BFS, DFS, Greedy Search e A*), planejamento e execução dos experimentos, coleta de métricas, análise comparativa dos resultados e redação integral deste relatório.

Uso de Inteligência Artificial

Conforme a política da disciplina, ferramentas de IA foram utilizadas como assistentes para tarefas auxiliares, e não para a geração da implementação central dos algoritmos. As ferramentas utilizadas foram:

- **GitHub Copilot:** Utilizado para auxiliar na refatoração e organização do código no repositório, especificamente na separação do código monolítico original (um único arquivo `.py`) em módulos distintos.
- **Google NotebookLM:** Empregado para facilitar a busca de informações e conceitos dentro dos slides da disciplina e para auxiliar na descoberta de fontes externas.
- **Google Gemini:** Utilizado para realizar uma verificação de conformidade, comparando os requisitos listados nas instruções do trabalho com o relatório finalizado, garantindo que todos os itens foram atendidos.

Reitera-se que nenhuma parte do código-fonte dos algoritmos de busca (BFS, DFS, Greedy, A*) ou das funções de heurística foi gerada por IA.

Recursos Externos

Para o desenvolvimento do trabalho, foram consultadas as seguintes fontes e documentações:

- **Slides da Disciplina:** Material de aula sobre Busca Não Informada e Busca Informada (Prof. Tiago Alves de Oliveira).
- **GeeksforGeeks:** Para consulta de informações gerais e exemplos de implementação dos algoritmos de busca.
- **Stack Overflow:** Para resolução de dúvidas pontuais sobre implementações em Python e manipulação de estruturas de dados.

- **Documentação Oficial do Python:** Para consulta sobre o uso de bibliotecas padrão.

Declaração

Eu, Matheus Emanuel da Silva, confirmo que o código entregue foi desenvolvido por mim, respeitando as políticas da disciplina.

Referências

[1] GEEKSFORGEEKS. **A* Search Algorithm in Python**. GeeksforGeeks, 23 jul. 2025. Disponível em: <https://www.geeksforgeeks.org/python/a-search-algorithm-in-python/>.

[2] GEEKSFORGEEKS. **Greedy Best-First Search in AI**. GeeksforGeeks, 23 jul. 2025. Disponível em: <https://www.geeksforgeeks.org/artificial-intelligence/greedy-best-first-search-in-ai/>.

[3] GEEKSFORGEEKS. **Monitoring Memory Usage of a Running Python Program**. GeeksforGeeks, 23 jul. 2025. Disponível em: <https://www.geeksforgeeks.org/python/monitoring-memory-usage-of-a-running-python-program/>.

[4] DESIGN GURUS. **What are the practical factors to consider when choosing between Depth-First Search (DFS) and Breadth-First Search (BFS)**. Design Gurus, 6 jun. 2025. Disponível em: <https://www.designgurus.io/answers/detail/what-are-the-practical-factors-to-consider-when-choosing-between-depth-first-search-dfs-and-breadth-first-search-bfs>.

[5] STACK OVERFLOW. **What does it mean by cold cache and warm cache concept**. Stack Overflow, 31 mar. 2014. Disponível em: <https://stackoverflow.com/questions/22756092/what-does-it-mean-by-cold-cache-and-warm-cache-concept>.

[6] PYTHON SOFTWARE FOUNDATION. **The Python Language Reference**. Versão 3.14.0. [S.l.: s.n.], [2025?]. Disponível em: <https://docs.python.org/3/reference/index.html>.

[7] OLIVEIRA, T. A. de. **EC_IA_004_Busca**. [S.d.]. 1. slides (Apresentação de aula). Divinópolis: CEFET-MG. Disponível em: [Mídias do Professor Tiago Alves de Oliveira].

[8] OLIVEIRA, T. A. de. **EC_IA_004_Busca-Parte2**. [S.d.]. 1. slides (Apresentação de aula). Divinópolis: CEFET-MG. Disponível em: [Mídias do Professor Tiago Alves de Oliveira].

[9] MEU NOTEBOOKLM. **Notebook sobre Busca, Cache e Algoritmos de IA**. [S.l.]: NotebookLM, [2025?]. Disponível em: <https://notebooklm.google.com/notebook/d800d6be-a086-463a-8aac-87a3ff869196>.