Lista: **Implementação e Análise de Algoritmos de Busca**
Disciplina: Inteligência Artificial
Profª.: Cristiane Neri Nobre
Data: 05/12/2025
Aluna: Alessandra Faria Rodrigues
Matrícula: 828333

Código disponível em:
https://github.com/ale-faria/CC/tree/main/IA/ListaExtra-MetodosDeBusca

---

Implementação e Análise de Algoritmos de Busca

1 INTRODUÇÃO AO PROBLEMA ABORDADO

Em muitos cenários, um agente inteligente precisa encontrar uma sequência de ações que o leve de um estado inicial até um estado objetivo (meta). Para isso, utilizam-se algoritmos de busca que exploram o espaço de estados do problema, tentando encontrar a solução de forma eficiente, seja minimizando o custo do caminho ou o tempo de computação.

O problema abordado neste trabalho é o **Jogo do 15** (15-Puzzle), um quebra-cabeça clássico de deslizamento. O jogo consiste em uma moldura de 4x4 posições contendo 15 peças numeradas de 1 a 15 e um espaço vazio. O objetivo é ordenar as peças em sequência numérica (do 1 ao 15) realizando movimentos deslizantes que utilizam o espaço vazio.

Apesar de suas regras simples, o Jogo do 15 apresenta um desafio computacional significativo. O espaço de estados (número de configurações possíveis das peças) é vasto, chegando a trilhões de permutações. Tentar resolver o jogo através de força bruta ou aleatoriedade é inviável computacionalmente. Portanto, este problema serve como um excelente ambiente de teste para a aplicação e comparação de diferentes estratégias de busca em IA.

O objetivo principal deste trabalho é implementar e analisar comparativamente três algoritmos de busca distintos aplicados a este jogo: Busca em Largura (BFS), Busca Gulosa (Greedy Search) e A* (A-Star).

2 EXPLICAÇÃO DOS ALGORITMOS E LÓGICA DE FUNCIONAMENTO

2.1 Busca em Largura (Breadth-First Search - BFS)

A Busca em Largura é um algoritmo de busca cega (ou não informada). Sua lógica consiste em explorar o espaço de estados nível por nível.

- **Funcionamento**: O algoritmo inicia no estado raiz e explora todos os seus vizinhos (filhos) antes de avançar para os vizinhos do próximo nível de profundidade. Para gerenciar a ordem de visitação, utiliza-se uma estrutura de dados do tipo Fila.
- **Análise**: A principal vantagem do BFS neste contexto é a garantia de otimalidade: se uma solução existe, o BFS encontrará o caminho com o menor número de movimentos possível. No entanto, sua desvantagem é o alto consumo de memória e tempo, pois o número de nós a serem armazenados cresce exponencialmente com a profundidade da solução.

2.2 Busca Gulosa (*Greedy Search*)

A Busca Gulosa é um algoritmo de busca informada que utiliza uma função heurística $h(n)$ para guiar a exploração.

- **Funcionamento**: A cada passo, o algoritmo seleciona o nó que parece estar mais próximo do objetivo, baseando-se exclusivamente no valor da heurística. A função de avaliação é $f(n) = h(n)$. Ele ignora o custo do caminho já percorrido $g(n)$.
- **Análise**: Por escolher sempre o "melhor caminho local" sem considerar o custo total, a Busca Gulosa tende a ser muito rápida e consumir menos memória que o BFS em muitos casos. Contudo, ela não garante a otimalidade (pode encontrar um caminho mais longo que o necessário) e nem sempre é completa (pode ficar presa em loops se não houver tratamento de estados visitados).

2.3 Algoritmo A* (A-Star)

O A* é o algoritmo central deste estudo. Ele combina as características da busca de custo uniforme com a busca gulosa, garantindo tanto eficiência quanto otimalidade (desde que a heurística seja admissível).

- **Funcionamento**: O algoritmo avalia os nós através da função $f(n) = g(n) + h(n)$, onde:
  - $g(n)$: O custo real do caminho do estado inicial até o nó atual n.
  - $h(n)$: A estimativa heurística do custo do nó n até o objetivo.
- **Análise**: Ao considerar $g(n)$, o A* penaliza caminhos que já estão ficando muito longos. Ao considerar $h(n)$, ele prioriza caminhos que parecem promissores. Isso faz com que ele explore menos nós que o BFS, mas,

diferentemente da Gulosa, ele nunca descarta a possibilidade de revisar um caminho se ele se provar mais curto globalmente.

3 JUSTIFICATIVA DAS HEURÍSTICAS ESCOLHIDAS

Para os algoritmos A* e Gulosa, a eficiência da busca depende diretamente da qualidade da função heurística $h(n)$. Para este trabalho, foram selecionadas três heurísticas distintas para análise comparativa:

3.1 Peças Fora do Lugar (Misplaced Tiles)

Esta é a heurística mais simples para o problema.

- **Lógica**: Conta o número de peças que não estão na sua posição final correta.
- **Justificativa**: É uma heurística admissível (nunca superestima o custo, pois cada peça errada precisará de pelo menos um movimento), mas é considerada "fraca" porque não informa o quão longe a peça está do seu destino, apenas que ela não está lá. Espera-se que o A* com esta heurística visite mais nós do que com as outras.

3.2 Distância de Manhattan

Esta heurística calcula a soma das distâncias horizontais e verticais de cada peça até sua posição objetivo.

- **Lógica**: $h(n) = \sum\left(\left|x_{atual} - x_{obj}\right| + \left|y_{atual} - y_{obj}\right|\right)$.
- **Justificativa**: No 15-Puzzle, as peças só podem se mover na vertical e horizontal (não há movimento diagonal). Portanto, a geometria de Manhattan reflete a "física" real do jogo. É uma heurística admissível e muito mais precisa que a anterior, devendo guiar o algoritmo de forma mais eficiente ao objetivo.

3.3 Distância Euclidiana

Esta heurística calcula a distância em linha reta (hipotenusa) de cada peça até seu objetivo.

- **Lógica**: $h(n) = \sqrt{\left(x_{atual} - x_{obj}\right)^2 + \left(y_{atual} - y_{obj}\right)^2}$
- **Justificativa**: Escolhida para fins acadêmicos. Embora geometricamente válida, ela é geralmente menos informativa para este problema específico do que a Manhattan, pois subestima mais o custo real (já que as peças não podem "voar" em linha reta). A análise do seu desempenho em relação à Manhattan demonstrará a importância de escolher uma heurística alinhada às regras de movimento do problema.

4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS EXPERIMENTAIS

Os testes foram realizados utilizando duas configurações de tabuleiro distintas (definidas pelas Seeds 2 e 15) para garantir a consistência das observações. A Tabela 1 apresenta o comparativo de tempo de execução, número de nós visitados e profundidade da solução encontrada.

Tabela 1 - Desempenho dos Algoritmos de Busca no 15-Puzzle

Seed	Algoritmo	Heurística	Tempo (s)	Nós Visitados	Profundidade
2	A*	Manhattan	0.0021s	17	40
	A*	Euclidiana	0.0020s	17	40
	A*	Peças Fora (Misplaced)	0.0010s	22	40
	Gulosa	Manhattan	0.0010s	11	40
	Gulosa	Euclidiana	0.0030s	37	46
	Gulosa	Peças Fora (Misplaced)	0.0490s	610	48
	BFS (Largura)	-	0.2532s	4669	40
15	A*	Manhattan	0.0010s	17	44
	A*	Euclidiana	0.0010s	17	44
	A*	Peças Fora (Misplaced)	0.0040s	50	44
	Gulosa	Manhattan	0.0010s	15	44
	Gulosa	Euclidiana	0.0010s	18	44
	Gulosa	Peças Fora (Misplaced)	0.0020s	19	44
	BFS (Largura)	-	5.2270	82039	44

4.1 Análise de Desempenho: BFS vs. A* e Gulosa

O resultado mais impactante dos testes é a diferença no consumo de recursos entre a **Busca em Largura (BFS) e os algoritmos A* e Gulosa**.

Na Seed 15, para encontrar uma solução de profundidade 44, o BFS precisou visitar **82.039 nós**, levando cerca de **5,22 segundos**. Em comparação, o algoritmo A* (com heurística Manhattan) resolveu o mesmo problema visitando apenas **17 nós** em **0,001 segundos**.

Concluindo, isso confirma a teoria de que a busca cega (BFS) é inviável para problemas com fatores de ramificação e profundidade moderados devido ao crescimento exponencial do espaço de busca. Embora o BFS garanta a solução ótima (profundidade 44), o custo computacional é proibitivo comparado ao uso de heurísticas.

4.2 Análise da Busca Gulosa: Velocidade vs. Qualidade

A Busca Gulosa apresentou o comportamento esperado de sacrificar a solução ótima em favor da velocidade, o que ficou evidente na Seed 2.

Enquanto o BFS e o A* encontraram o caminho ótimo de **40 passos**, a Busca Gulosa com heurística Euclidiana encontrou um caminho de **46 passos**, e com Misplaced um caminho de **48 passos**. Isso ocorre porque a Gulosa não considera o custo acumulado $g(n)$, tomando decisões imediatistas que podem levar a caminhos mais longos.

Outro ponto relevante na Seed 2 é que a Gulosa com Misplaced visitou **610 nós**, muito mais que o A* (22 nós). Isso demonstra que, sem o contrapeso do custo $g(n)$, uma heurística fraca pode fazer a Busca Gulosa se "perder" em caminhos irrelevantes, tornando-a, neste caso específico, pior que o A* tanto em tempo quanto em qualidade da solução.

4.3 Análise do Algoritmo A* e Comparação de Heurísticas

O algoritmo A* demonstrou consistência absoluta, encontrando sempre a solução ótima (profundidades 40 e 44, idênticas ao BFS) em tempos extremamente baixos. Observa-se claramente que a qualidade da heurística afeta o número de nós visitados.

- **Manhattan e Euclidiana:** Tiveram desempenho excelente e idêntico nos testes (17 nós visitados em ambos os cenários). Sendo heurísticas mais informativas, elas guiaram a busca direto ao objetivo com desvios mínimos.
- **Peças Fora do Lugar (Misplaced):** Sendo uma heurística menos precisa, fez o A* explorar mais nós (22 na Seed 2 e 50 na Seed 15). Embora ainda muito superior ao BFS, ela é claramente menos eficiente que a Manhattan para limitar o espaço de busca.

A Tabela 2 apresenta o comparativo entre as heurísticas para o algoritmo A*.

Tabela 2: Comparação das Heurísticas no Algoritmo A*

Heurística	Média de Tempo (s)	Média de Nós Visitados	Qualidade da Solução (Profundidade)
Manhattan	0.0015s	17	Ótima
Euclidiana	0.0015s	17	Ótima
Peças Fora	0.0025s	36	Ótima

4.4 Influência da Configuração Inicial (Topologia do Espaço de Busca)

Os resultados evidenciam que a configuração inicial do tabuleiro impacta os algoritmos de formas distintas. Para a **Busca em Largura (BFS)**, o fator crítico é a profundidade da solução. Observou-se que um aumento de apenas 4 passos na profundidade ótima (da Seed 2 para a Seed 15) resultou em um aumento no número de nós visitados (de 4.669 para 82.039), confirmando a natureza exponencial do algoritmo.

Por outro lado, para a **Busca Gulosa**, a dificuldade não está necessariamente na profundidade, mas na topografia heurística do tabuleiro. A Seed 2, embora exija menos passos para ser resolvida (40), apresentou uma configuração enganosa para a heurística de Peças Fora do Lugar (Misplaced), fazendo o algoritmo visitar 610 nós. Isso ocorre quando a ação necessária para resolver o puzzle exige momentaneamente piorar o valor da heurística (afastar peças do objetivo para abrir passagem), algo que a abordagem gulosa tenta evitar, desviando-se do caminho ótimo. O algoritmo **A*** manteve-se estável e eficiente em ambas as configurações, demonstrando sua robustez.

O vídeo contendo o passo a passo do caminho de cada solução encontra-se disponível no seguinte link:
https://github.com/ale-faria/CC/tree/main/IA/ListaExtra-MetodosDeBusca/Testes

5 CONCLUSÃO

Este trabalho apresentou a implementação e a análise comparativa de três algoritmos de busca (BFS, Busca Gulosa e A*) aplicados ao problema do 15-Puzzle, utilizando uma interface gráfica desenvolvida em Python para visualização e coleta de métricas experimentais.

Os experimentos realizados confirmaram na prática os conceitos teóricos da Inteligência Artificial sobre busca em espaço de estados. Em relação à pergunta sobre qual algoritmo foi mais rápido e qual visitou menos nós, os resultados foram decisivos:

1. **Eficiência do A*:** O algoritmo A* demonstrou ser a abordagem mais equilibrada e robusta. Ele foi capaz de encontrar a solução ótima (caminho de menor custo) em tempos inferiores a 0,01 segundos, visitando um número mínimo de nós (apenas 17 nas Seeds testadas), superando drasticamente a Busca em Largura.
2. **Inviabilidade do BFS:** A Busca em Largura (BFS), embora garanta a solução ótima, provou-se inviável para instâncias mais profundas do problema devido ao crescimento exponencial do espaço de busca. A diferença de desempenho observada entre a Seed 2 e a Seed 15 ilustra como um pequeno aumento na profundidade da solução pode multiplicar o tempo de processamento e o consumo de memória em dezenas de vezes.
3. **Riscos da Busca Gulosa:** A Busca Gulosa mostrou-se rápida, mas instável. Em configurações específicas (como na Seed 2), ela produziu soluções sub-ótimas (caminhos mais longos que o necessário) e, quando utilizada com uma heurística fraca (Misplaced), chegou a visitar mais nós do que o A*, perdendo sua principal vantagem de velocidade.

Sobre o impacto das heurísticas no A*, conclui-se que a escolha da função $h(n)$ é determinante para o desempenho. A heurística de **Distância de Manhattan** e a **Euclidiana** apresentaram os melhores resultados, guiando a busca de forma muito mais direta do que a heurística de **Peças Fora do Lugar (Misplaced)**. Esta última, por ser menos informativa, fez com que o algoritmo explorasse uma quantidade maior de estados desnecessários.

Portanto, conclui-se que para problemas de caminho mínimo com ramificação moderada como o 15-Puzzle, o algoritmo **A* combinado com a heurística de Distância de Manhattan** é a melhor abordagem, pois une a garantia de solução ótima da busca em largura com a eficiência de direcionamento da busca gulosa, evitando a explosão combinatória sem sacrificar a qualidade da solução.