COMPARAÇÃO DE ALGORITMOS DE BUSCA EM INTELIGÊNCIA ARTIFICIAL NO ROMANIA MAP PROBLEM

Aluno: Sávio Mendes de Souza

Resumo: Os algoritmos de busca são fundamentais na Inteligência Artificial para a resolução de problemas complexos por meio de processos racionais. Este trabalho tem como objetivo analisar e implementar diferentes algoritmos de busca aplicados ao Romania Map Problem – um problema clássico que consiste em encontrar o caminho mais curto entre duas cidades representadas como nós de um grafo, cujas arestas correspondem às estradas. A implementação, realizada em Java, envolve algoritmos sem informação (Busca em Largura, Busca de Custo Uniforme, Busca em Profundidade, Busca Iterativa em Profundidade, Busca Limitada em Profundidade e Busca Bidirecional) e algoritmos com informação (Busca Gulosa e A*). A avaliação baseou-se em duas métricas: a soma dos custos das arestas (distância total) e a porcentagem de nós explorados durante a busca. Os resultados indicam que os algoritmos informados, especialmente o A*, demonstram maior eficiência e qualidade das soluções, evidenciando a importância do uso de heurísticas na otimização dos processos de busca.

Palavras-chave: Inteligência Artificial; Algoritmos de Busca; Grafo; Busca sem Informação; Busca com Informação; A*; Heurística; Java.

Abstract: Search algorithms are fundamental in Artificial Intelligence for solving complex problems through rational processes. This work aims to analyze and implement various search algorithms applied to the Romania Map Problem—a classic problem involving finding the shortest path between two cities, represented as nodes in a graph where roads serve as edges. The implementation was carried out in Java. The study evaluates both uninformed search algorithms, such as Breadth-First Search (BFS), Uniform Cost Search (UCS), Depth-First Search (DFS), Iterative Deepening Depth-First Search (IDDFS), Depth-Limited Search (DLS), and Bidirectional Search, as well as informed search algorithms, including Greedy Search and A*. Performance metrics considered include the total distance traversed and the percentage of nodes explored during the execution of the algorithms. The results indicate that informed search algorithms—particularly A*—demonstrate superior performance in terms of efficiency and solution quality, underscoring the importance of heuristics in optimizing search processes.

Keywords: Artificial Intelligence; Search Algorithms; Graph; Uninformed Search; Informed Search; A*; Heuristic; Java

1. Introdução

A Inteligência Artificial (IA) é um campo da computação que visa simular processos cognitivos humanos, como aprendizado, raciocínio e tomada de decisão. Os algoritmos de busca são essenciais na Inteligência Artificial, pois permitem a resolução eficiente de problemas que exigem a exploração de grandes espaços de estados, como em planejamento, navegação e otimização. Aplicações práticas incluem desde sistemas de navegação por GPS até estratégias em jogos de tabuleiro e inteligência de agentes robóticos.

Um exemplo clássico utilizado para avaliar a eficiência desses algoritmos é o Romania Map Problem, que modela a busca pelo menor caminho entre duas cidades em um grafo onde os nós representam cidades e as arestas representam estradas. Esse problema não apenas fornece um cenário bem definido para comparação de algoritmos, mas também reflete desafios encontrados em sistemas reais de roteamento e logística. Assim, ao implementar e analisar diferentes estratégias de busca nesse contexto, podemos compreender melhor as vantagens e limitações de cada abordagem e seus impactos na eficiência computacional.

Esse tipo de problema é amplamente estudado em IA, pois envolve conceitos cruciais, como representação de estados, espaços de busca e estratégias de exploração. Neste trabalho, buscamos avaliar e implementar diferentes algoritmos de busca no contexto desse problema clássico, utilizando Java como linguagem de implementação.

O objetivo é comparar algoritmos de busca com e sem informação, como a Busca em Largura (Breadth-First Search - BFS), Busca de Custo Uniforme (Uniform Cost Search - UCS), Busca em Profundidade (Depth-First Search - DFS), Busca Gulosa (Greedy Search) e o Algoritmo A*.

A análise e a comparação dos algoritmos serão baseadas em métricas como a eficiência do tempo de execução, o número de nós explorados e a qualidade do caminho encontrado. Ao final, espera-se identificar as vantagens e limitações de cada algoritmo, além de demonstrar o impacto da utilização de heurísticas nas buscas com informação, como ocorre no caso do Algoritmo A*, que utiliza uma função heurística para guiar a busca de forma mais eficiente.

2. Metodologia

A metodologia adotada para este trabalho visa implementar, avaliar e comparar diferentes algoritmos de busca no contexto do problema do mapa da Romênia. O foco é

explorar tanto algoritmos de busca sem informação (como Busca em Largura, Busca em Profundidade e Busca de Custo Uniforme) quanto algoritmos com heurísticas (como a Busca Gulosa e o Algoritmo A*). A seguir, são apresentados os passos detalhados para a implementação e análise dos algoritmos:

2.1 Definição do Problema

O problema a ser resolvido consiste em encontrar o caminho mais curto entre duas cidades no mapa da Romênia. As cidades são representadas por nós de um grafo, e as estradas entre elas por arestas. A tarefa é determinar a sequência de cidades que conecta a cidade inicial à cidade final, minimizando a distância percorrida.

2.2 Estrutura de Dados

A implementação do problema é feita utilizando uma estrutura de dados de grafo, onde cada cidade é representada por um nó e as conexões entre as cidades por arestas. O grafo será representado por uma lista de adjacência, onde cada elemento da lista contém informações sobre as cidades vizinhas e as distâncias entre elas.

2.3 Algoritmos de Busca Implementados

2.3.1 Busca em Largura (Breadth-First Search - BFS):

Este algoritmo explora todos os nós em um nível de profundidade antes de passar para o próximo nível. No contexto do problema, BFS é utilizado para garantir que o caminho encontrado seja o mais curto, desde que todas as arestas tenham o mesmo custo.

Limitações: Consome muita memória em grafos extensos.

2.3.2 Busca de Custo Uniforme (Uniform Cost Search - UCS):

A UCS é uma versão do BFS que leva em consideração o custo das arestas, permitindo encontrar o caminho de menor custo mesmo quando as distâncias entre os nós são diferentes.

Limitações: Pode apresentar desempenho inferior em termos de tempo em casos de muitos caminhos de custo semelhante.

2.3.3 Busca em Profundidade (Depth-First Search - DFS):

Este algoritmo explora o grafo profundamente, indo o mais longe possível em cada ramificação antes de voltar atrás. Embora seja eficiente em termos de memória, o DFS não

garante encontrar o caminho mais curto.

Limitações: Não garante a solução ótima e pode resultar na exploração de muitos nós desnecessários.

2.3.4 Busca Gulosa (Greedy Search):

A busca gulosa utiliza uma heurística para direcionar a exploração do grafo em direção ao objetivo, sem se preocupar com o custo total do caminho. Ela tenta encontrar o caminho mais curto com base em uma estimativa da distância restante até o objetivo.

Limitações: Pode não encontrar a solução ótima, já que ignora o custo acumulado.

2.3.5 Algoritmo A*:

O A* é um algoritmo de busca informada que combina os conceitos de BFS e Busca Gulosa. Ele utiliza uma função de custo total, composta pelo custo já percorrido e uma estimativa heurística do custo restante, para guiar a busca de forma mais eficiente. A heurística utilizada é baseada na distância em linha reta (heurística de distância Euclidiana) entre as cidades.

Limitações: A eficácia depende fortemente da qualidade da heurística escolhida.

2.3.6 Busca Iterativa em Profundidade (IDDFS):

A busca iterativa em profundidade é uma variação do DFS que evita o problema de exploração infinita em grafos muito profundos. Ela realiza múltiplas buscas DFS limitadas à profundidade crescente até encontrar o objetivo. A principal vantagem desse algoritmo é que ele garante a solução ótima (como a BFS), mas usa menos memória (como o DFS), já que realiza iterações limitadas em profundidade. Este algoritmo será útil para comparar o desempenho com o DFS em situações onde a profundidade da solução não é conhecida.

Limitações: Repetição de buscas anteriores pode aumentar o tempo total de execução.

2.3.7 Busca Limita Profundidade (DLS):

Uma variação da Busca em Profundidade (DFS), mas com um limite máximo de profundidade. Ele evita explorar caminhos muito profundos, o que pode ser útil quando a profundidade do objetivo é conhecida ou limitada.

Limitações: Se o limite for muito baixo, pode não encontrar a solução; se for alto, as

vantagens em relação ao DFS são perdidas.

2.3.8 Busca Bidirecional (BIDIRECTIONAL):

Realiza duas buscas simultaneamente, uma a partir do nó inicial e outra a partir do nó objetivo. A ideia é que ambas as buscas se encontrem no meio, reduzindo significativamente o número de nós explorados. Esse algoritmo é muito eficaz em grafos grandes, já que as buscas convergem mais rapidamente.

Limitações: Requer uma implementação cuidadosa para sincronizar as duas buscas.

3 Implementação

Cada algoritmo foi implementado de forma modular, permitindo uma comparação direta dos resultados. A implementação seguiu os seguintes passos:

3.1 Inicialização do Grafo:

Leitura dos dados que representam as cidades e as estradas, com suas respectivas distâncias.

Construção do grafo utilizando uma lista de adjacência.

3.2 Execução dos Algoritmos:

Implementação modular de cada algoritmo de busca, utilizando estruturas de dados como filas, pilhas e filas de prioridade conforme necessário.

Execução dos algoritmos para determinar o caminho entre a cidade de origem e a cidade destino.

3.3 Registro dos Resultados:

Medição do tempo de execução;

Contabilização do número de nós explorados;

Cálculo do somatório dos custos das arestas do caminho encontrado;

4 Resultados

O objetivo principal de cada algoritmo de busca é gerar uma sequência de cidades a serem visitadas, representando o percurso entre a cidade de origem e a cidade destino. Esse

percurso é expresso por uma lista ordenada das cidades visitadas, do ponto de partida até o objetivo final. O desempenho dos algoritmos foi avaliado com base em duas métricas principais:

Somatório do tamanho das arestas: A soma dos custos das conexões entre os nós do caminho encontrado.

Porcentagem de nós explorados: A proporção dos nós visitados pelo algoritmo durante o processo de busca.

A seguir, apresentamos um exemplo de resultado obtido para o percurso entre as cidades Arad e Fagaras.

Exemplo de Resultado: De Arad a Fagaras

Algoritmo	Porcentagem de Nós Percorridos	Caminho Encontrado	Distância
Busca em extensão (amplitude) - BFS	35.71%	Arad -> Sibiu -> Fagaras	239
Busca de custo uniforme – UCS	35.71%	Arad -> Sibiu -> Fagaras	239
Busca em profundidade – DFS	64,29%	Fagaras -> Bucharest -> Pitesti -> Craiova -> Drobeta -> Mehadia -> Lugoj -> Timisoara -> Arad	944
Busca em profundidade limitada (Limite de 5) – DLS	35.71%	Arad -> Zerind -> Arad -> Sibiu -> Fagaras	389
Busca em profundidade limitada (Limite de 10) – DLS	64,29%	Fagaras -> Bucharest -> Pitesti -> Craiova -> Drobeta -> Mehadia -> Lugoj -> Timisoara -> Arad	944

Busca de aprofundamento Iterativo – IDDFS	21.43%	Arad -> Sibiu -> Fagaras	239
Busca direcional - BIDIRECTIONAL	35.71%	Arad -> Sibiu -> Fagaras	239
Busca gulosa – GFS	35.71%	Arad -> Sibiu -> Fagaras	239
Algoritmo A* - ASTAR	35.71%	Arad -> Sibiu -> Fagaras	239

Observação: Os caminhos representados de forma resumida indicam as principais sequências encontradas; alguns algoritmos, como o DFS, podem gerar caminhos muito mais longos, evidenciando sua ineficiência na exploração do espaço de busca.

5 Análise dos Resultados

A análise dos resultados permitiu identificar pontos fortes e limitações dos algoritmos testados:

5.1 Algoritmos Informados (Busca Gulosa e A):*

Os algoritmos que utilizam heurísticas demonstraram desempenho superior, explorando menos nós e encontrando soluções de menor custo. Em particular, o A* mostrou uma combinação eficiente entre custo acumulado e estimativa heurística, resultando em caminhos ótimos com um menor esforço computacional.

5.2 Algoritmos Não Informados:

BFS e UCS: Ambos apresentaram desempenho consistente, encontrando o caminho mais curto quando os custos são uniformes ou quando se trata de pesos diferentes, respectivamente.

DFS: Embora encontre uma solução válida, o DFS explorou um número significativamente maior de nós, resultando em um caminho subótimo e demonstrando sua limitação em problemas onde a eficiência é crucial.

IDDFS e DLS: Essas variações do DFS procuram equilibrar o consumo de memória e a completude da busca, mas os resultados indicam que a escolha do limite é crítica para seu desempenho.

Busca Bidirecional: Mostrou-se eficaz em reduzir o número de nós explorados, porém sua implementação requer cuidado especial na sincronização das buscas a partir dos nós inicial e final.

6 Conclusões

O Os experimentos realizados demonstraram que algoritmos de busca informados, como o A*, são mais eficientes na resolução do Romania Map Problem, pois conseguem encontrar caminhos ótimos com uma exploração reduzida do espaço de busca. Essa eficiência se deve ao uso de heurísticas que direcionam a busca para as regiões mais promissoras do grafo.

Por outro lado, os algoritmos não informados, embora garantam a completude da busca, apresentam limitações em termos de tempo de execução e número de nós explorados, especialmente em grafos de grande escala.

Esses resultados têm impacto direto em diversas aplicações, como navegação por GPS, otimização de rotas e desenvolvimento de estratégias em jogos e robótica. Como trabalhos futuros, recomenda-se a investigação de novas heurísticas e a aplicação dos algoritmos testados em cenários mais complexos, de modo a validar sua eficácia em ambientes com variáveis mais dinâmicas.

7 Referências

RUSSELL, Stuart; NORVIG, Peter. Inteligência Artificial: uma abordagem moderna. 3. ed. Rio de Janeiro: Elsevier, 2013.

NORVIG, P.; RUSSELL, S. aima-python: Python implementation of algorithms from "Artificial Intelligence: A Modern Approach". 2025. Disponível em: https://github.com/aimacode/aima-python. Acesso em: 31 mar. 2025.

FEOFILOFF, Paulo: Algoritmos para Grafos. Disponível em https://www.ime.usp.br/~pf/algoritmos_para_grafos/index.html#contents. Acesso em 30 mar. 2025.