INSTITUTO FEDERAL DO ESPÍRITO SANTO
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA
PPCOMP

FRANCO MARCHIORI LOUZADA

INTELIGÊNCIA ARTIFICIAL
TRABALHO 1

SERRA
2023

SUMÁRIO

1 FUNDAMENTAÇÃO TEÓRICA

Os algoritmos de busca desempenham um papel crucial na resolução de problemas em inteligência artificial. Eles são desenvolvidos para explorar e analisar espaços de busca complexos, com o objetivo de encontrar soluções adequadas. Ao explorar de forma sistemática o espaço de busca, esses algoritmos se tornam ferramentas indispensáveis para resolver problemas desafiadores na área de inteligência artificial, contribuindo para avanços significativos nesse campo de estudo.

Além disso, para executar um algoritmo de busca, é necessário identificar um **objetivo** e formular um **problema** bem definido. Um problema é composto por cinco partes essenciais: o **estado inicial**, que representa o ponto de partida; um conjunto de **ações** possíveis que podem ser tomadas a partir de cada estado; um **modelo de transição**, que descreve como as ações levam a transições de estado; uma função **teste de objetivo**, que verifica se um estado é o estado objetivo desejado; e uma função c**usto de caminho**, que atribui um valor de custo a cada ação ou transição de estado. É importante ressaltar que a **solução** para um problema de busca é definida como um **caminho** através do espaço de estados, indo do estado inicial ao estado objetivo. Uma solução é **ótima** se, entre todas as soluções, possui o menor custo de caminho.

Alguns métodos de busca mais simples têm acesso apenas à definição do problema e do objetivo, mas não têm informações sobre uma estimativa de custo para encontrar a solução, por exemplo. Por isso são chamados de métodos de **busca não informada**, ou busca cega. Há também estratégias onde é dada uma estimativa do custo do caminho de um estado até o estado objetivo. Por isso são chamados de métodos de **busca informada** ou **busca heurística**.

Para avaliar o desempenho de um algoritmo é necessário considerar alguns critérios para escolher a melhor solução. Pode-se fazer isso em quatro aspectos:

- **Completeza:** O algoritmo oferece a garantia de encontrar uma solução quando ela existir?
- **Otimização:** A estratégia encontra a solução ótima?
- **Complexidade de tempo:** Quanto tempo ele leva para encontrar uma solução?
- **Complexidade de espaço:** Quanta memória é necessária para executar a busca?

1.1 Busca em Largura

O algoritmo busca em largura (**BFS**, *Breadth-First Search*) é uma estratégia onde o nó inicial é expandido, em seguida todos os seu sucessores são expandidos, depois os sucessores desses nós, e assim por diante. A cada nó expandido, seus vizinho são adicionados a uma fila FIFO (*first in, first out*), onde o primeiro a entrar na fila é o primeiro a sair. Assim, o nó mais raso não expandido é o escolhido para a próxima expansão.

O BFS garante que a solução encontrada seja a mais rasa possível, pois explora os nós em largura. Ele é ideal para problemas em que a solução mais rasa é preferível ou quando todos os caminhos possíveis devem ser explorados. Portando, o BFS é completo; ótimo quando os passos têm custo unitário e tem complexidade de tempo exponencial.

1.2 Busca em Profundidade

O método de busca em profundidade (**DFS**, *Depth-First Search*) sempre expande o nó mais profundo na borda atual. A cada nó expandido, seus vizinhos são adicionados à uma fila LIFO (*last in, first out*), assim a busca prossegue até o nível mais profundo da árvore de busca, onde os nós não têm sucessores.

Comparado ao BFS, o DFS pode encontrar soluções profundas mais rapidamente. Este método é recomendado para problemas em que o espaço de busca é muito grande e não é necessário explorar todos os caminhos possíveis. O DFS não é completo nem ótimo e tem complexidade espacial linear.

1.3 Busca de Custo Uniforme

O algoritmo de busca de custo uniforme (**UCS**, *Uniform-Cost Search*) expande o nó com o menor custo de caminho até o momento. O custo do caminho é dado pela função *g(n)*, onde para cada transição entre nós é atribuído um custo. A cada nó expandido os seus vizinhos são armazenados em uma fila de prioridade ordenada por *g*. Dessa forma, o UCS prioriza os nós com menor custo e continua expandindo-os até encontrar o objetivo ou explorar todo o espaço de busca.

O UCS não se importa com a quantidade de passos de um caminho, mas apenas com o seu custo total, por isso ficará preso em um laço infinito se existir um caminho com sequência infinita de ações a custo zero. Logo, não é completo para este caso. O UCS é completo quando todos os custos de transição são maiores que zero. Em relação à otimalidade, este método é ótimo se todos os custos de ações são não negativos.

1.4 Busca A*

O algoritmo de busca A estrela (**A***, *A-Star*) é um método de busca informada que avalia os nós pela função *f(n) = g(n) + h(n)*, onde *g(n)* é o custo acumulado do caminho e *h(n)* é uma estimativa (**heurística**) do do custo para ir do nó *n* ao objetivo. A cada nó expandido os seus vizinhos são adicionados a uma fila de prioridade ordenada por *f*, assim, tenta encontrar a solução de menor custo.

Para que o A* seja completa a função heurística deve ser admissível, ou seja, *h(n)* não pode ser maior que o custo real para atingir o objetivo. Para garantir otimalidade, *h(n)* deve ser exatamente igual ao custo real para chegar ao estado ou nó objetivo.

1.5 Comparação de complexidade

A Tabela 1 apresenta uma comparação de complexidade de tempo e espaço entre os algoritmos desenvolvidos.

Tabela 1: Complexidade dos métodos BFS, DFS, UCS e A*

	BFS	DFS	UCS	A*
Complexidade de Tempo	$O(b^d)$	$O(b^m)$	$O\left(b^{1+[C^*/\epsilon]}\right)$	$O(b^d)$
Complexidade de Espaço	$O(b^d)$	O(bm)	$O\left(b^{1+[C^*/\epsilon]}\right)$	$O(b^d)$

Onde,

- b é o fator de ramificação;
- d é a profundidade da solução mais rasa;
- m é a profundidade máxima da árvore de busca;
- C^* é o custo da solução ótima;
- $\in$ é uma constante positiva pequena.

2 EXPERIMENTOS

2.1 Questão 1

O procedimento experimental consistiu em testar a execução dos quatro algoritmos de busca BFS, DFS, UCS e A*, em um labirinto com tamanho 300x300 com percentual de bloqueio 50% gerado aleatoriamente. O experimento foi repetido 10 vezes para obter uma média mais precisa dos resultados e reduzir a influência de variações aleatórias. O custo de transição de um nó para o outro foi calculado como a distância euclidiana entre os centros dos quadrados vizinhos.

A Figura 1 mostra um exemplo de caminhos encontrados pelos algoritmos testados em um labirinto de tamanho 10x15.

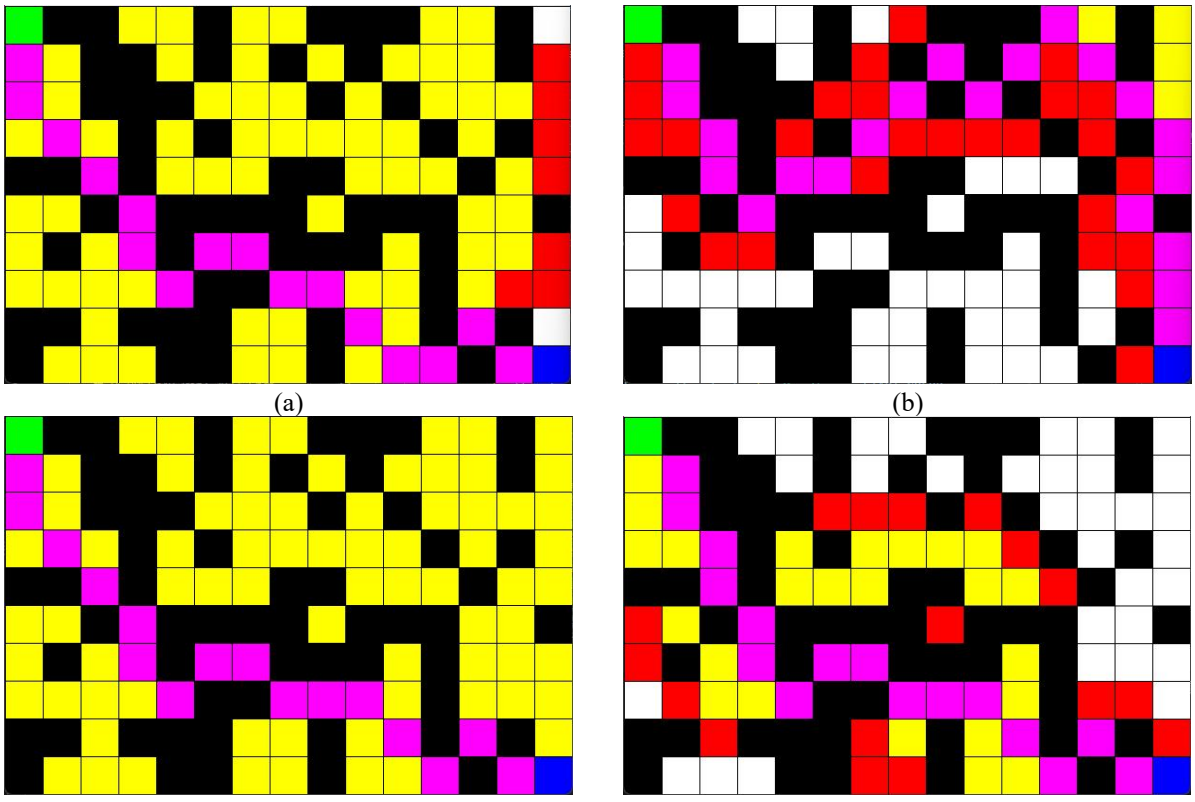

Figura 1: Exemplos de caminhos encontrados em labirinto um de tamanho 10x15 usando método: (a) BFS; (b) DFS; (c) UCS; e (d) A*.

Os quadrados verdes são o estado inicial, os azuis o estado objetivo, os brancos são os nós liberados, os pretos são os obstáculos que não podem ser visitados, os rosas indicam o caminho encontrado, os amarelos são os nós expandidos, os vermelhos são os nós adicionados à fila de vizinhos mas que não foram visitados.

2.2 Questão 2

Para este problema o grafo dado era fixo, logo cada algoritmo precisou ser executado apenas uma vez para avaliar os tempos e custos de execução. A Figura 2 apresenta o grafo utilizado para os testes.

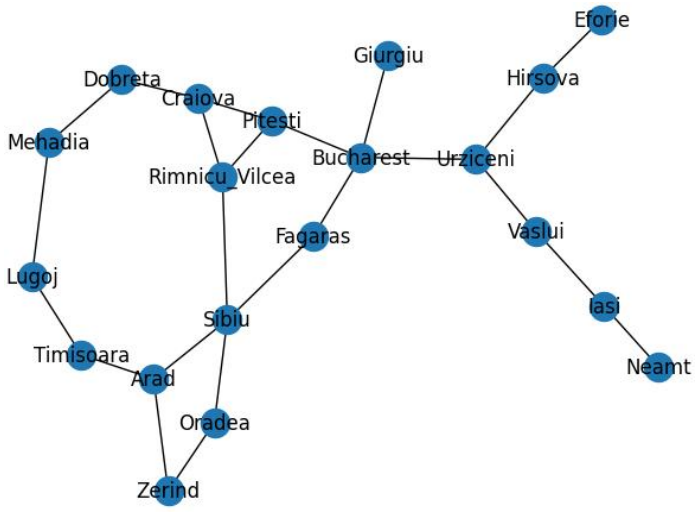

Figura 2: Grafo da questão 2

2.3 Ambiente de Trabalho:

Para o desenvolvimento e testes realizados utilizou-se um notebook com a configuração apresentada na Tabela 2:

Tabela 2: Configurações do ambiente de trabalho

Processador	Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz
Placa Gráfica	Intel(R) UHD Graphics 620
Memória RAM	16,0 GB (utilizável: 15,9 GB)
Sistema Operacional	Windows 11 Home Versão 22H2 de 64 bits

3 RESULTADOS

As métricas utilizadas para comparar o desempenho dos algoritmos foram as seguintes:

- **Tempo de execução:** Mediu-se o tempo total necessário para cada algoritmo encontrar o caminho no labirinto. Isso permitiu avaliar a eficiência temporal de cada algoritmo, ou seja, o quão rápido ele foi capaz de encontrar a solução.
- **Número de nós expandidos:** Contou-se o número de nós do labirinto que foram visitados e explorados por cada algoritmo durante a busca pelo caminho. Essa métrica deu uma indicação da eficiência espacial de cada algoritmo, ou seja, a quantidade de memória necessária para executar o algoritmo.
- **Custo do caminho:** Avaliou-se o custo total do caminho encontrado por cada algoritmo. O custo pode ser definido como a soma dos custos associados às transições entre os nós no caminho. Essa métrica permitiu comparar os diferentes algoritmos quanto a sua otimalidade para encontrar caminhos.
- **Tamanho do caminho:** Mediu-se o número de passos necessários para percorrer o caminho encontrado por cada algoritmo.

3.1 Questão 1

Tabela 3: Resultados dos testes da Questão 1

	BFS	DFS	UCS	A*
Tempo de execução (segundos)	37,0238	1,7619	656,3412	291,3711
Número de nós expandidos	544890	9458	544933	214222
Custo do caminho	4748,9279	9316,6473	4620,012	4633,8699
Tamanho do caminho	3597	7456	3623	3641

3.2 Questão 2

Tabela 4: Resultados dos testes da Questão 2

	BFS	UCS	A*
Tempo de execução (segundos)	1,123	0,0713	0,0744
Número de nós expandidos	20	11	9
Custo total do caminho	366	366	455
Tamanho do caminho	3	3	4
Caminho	Arad, Sibiu, Rimnicu_Vilcea, Craiova	Arad, Sibiu, Rimnicu_Vilcea, Craiova	Arad, Sibiu, Rimnicu_Vilcea, Pitesti, Craiova