|  |
| --- |
| Instituto FEDERAL DO ESPÍRITO SANTO  PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA – PPCOMP  FRANCO MARCHIORI LOUZADA  Inteligência artificial trabalho 1 |
| serrA  2023 |

**LISTA DE FIGURAS**

##### **Erro! Nenhuma entrada de tabela de imagens encontrada.**

**SUMÁRIO**

[Erro! Nenhuma entrada de tabela de imagens encontrada. ii](#_Toc11313)

[1 Fundamentação Teórica 4](#_Toc286)

[1.1 Busca em Largura 4](#_Toc27693)

[1.2 Busca em profundidade 5](#_Toc4456)

[2 Experimentos 7](#_Toc17075)

[2.1 Questão 1 7](#_Toc3923)

[2.2 Questão 2 7](#_Toc32156)

[2.3 Ambiente de Trabalho: 7](#_Toc5425)

[3 Resultados 9](#_Toc19030)

[3.1 Questão 1 9](#_Toc3501)

[3.2 Questão 2 9](#_Toc22529)

[REFERÊNCIAS BIBLIOGRÁFICAS 10](#_Toc17897)

# 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, como definido na página 68?
* **Complexidade de tempo:** Quanto tempo ele leva para encontrar uma solução?
* **Complexidade de espaço:** Quanta memória é necessária para executar a busca?

## 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.

## 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.

2. Busca em Largura (BFS):

- Descrição geral do algoritmo BFS.

- Explicação do funcionamento do BFS, que explora todos os nós de um determinado nível antes de avançar para o próximo nível.

- Análise da complexidade temporal e espacial do BFS.

- Vantagens e limitações do BFS em relação a outros algoritmos de busca.

3. Busca em Profundidade (DFS):

- Descrição geral do algoritmo DFS.

- Explicação do funcionamento do DFS, que explora o máximo possível em um ramo antes de retroceder.

- Análise da complexidade temporal e espacial do DFS.

- Vantagens e limitações do DFS em relação a outros algoritmos de busca.

4. Busca de Custo Uniforme (UCS):

- Descrição geral do algoritmo UCS.

- Explicação do funcionamento do UCS, que seleciona o caminho com o menor custo acumulado até o momento.

- Discussão sobre a importância da definição correta dos custos associados às transições entre os nós para o funcionamento adequado do UCS.

- Análise da complexidade temporal e espacial do UCS.

- Vantagens e limitações do UCS em relação a outros algoritmos de busca.

5. A\* (A Estrela):

- Descrição geral do algoritmo A\*.

- Explicação do funcionamento do A\*, que combina a heurística (estimativa do custo restante) com o custo acumulado para selecionar o próximo nó a ser explorado.

- Discussão sobre a importância da heurística admissível e consistente para garantir a otimalidade do A\*.

- Análise da complexidade temporal e espacial do A\*.

- Vantagens e limitações do A\* em relação a outros algoritmos de busca.

6. Comparação dos algoritmos de busca:

- Discussão sobre as diferenças fundamentais entre BFS, DFS, UCS e A\* em termos de estratégias de busca, eficiência e otimalidade.

- Comparação das complexidades temporais e espaciais dos algoritmos.

- Considerações sobre os cenários em que cada algoritmo pode ser mais adequado, levando em conta as características específicas do problema.

Lembre-se de que esses tópicos são apenas sugestões para a fundamentação teórica. É importante pesquisar e adaptar o conteúdo de acordo com as referências acadêmicas disponíveis e os requisitos específicos do seu trabalho.

# Experimentos

## 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.

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.

## Questão 2

## Ambiente de Trabalho:

Para o desenvolvimento e testes realizados utilizou-se um notebook com a configuração apresentada na Tabela 1:

Tabela 1: Configurações do ambiente de trabalho

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

# Resultados

## Questão 1

Tabela 2: 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 |

## Questão 2

Tabela 3: 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 do caminho | 366.0 | 366.0 | 455.0 |
| Tamanho do caminho | 3 | 3 | 4 |
| Caminho | Arad, Sibiu, Rimnicu\_Vilcea, Craiova | Arad, Sibiu, Rimnicu\_Vilcea, Craiova | Arad, Sibiu, Rimnicu\_Vilcea, Pitesti, Craiova |

# REFERÊNCIAS BIBLIOGRÁFICAS

**Error! Indicador Não Definido .**