Uma imagem com texto

Descrição gerada automaticamente

**Instituto Politécnico do Cávado e do Ave**

**Escola Superior de Tecnologia**

**Uma imagem com arte, Gráficos, clipart, design

Descrição gerada automaticamente**

**Licenciatura**

**em**

**Engenharia Informática Médica**

**Inteligência Artificial**

Bruno Rafael Mendes Oliveira – a15566

Diogo Mário Sá Fernandes – a24017

**Dezembro de 2023**

Esta página foi deixada em branco propositadamente.

Uma imagem com texto

Descrição gerada automaticamente

**Instituto Politécnico do Cávado e do Ave**

**Escola Superior de Tecnologia**

**Licenciatura**

**em**

**Engenharia Informática Médica**

**Relatório do Projeto Engenharia de Software**

**Gestão de Armazéns por Inteligência Artificial**

**Unidade Curricular**

Inteligência Artificial

**Nome dos Alunos**

Bruno Oliveira

Diogo Fernandes

**Docente da Unidade Curricular:**

Profª. Joaquim Gonçalves

**Dezembro de 2023**

Esta página foi deixada em branco propositadamente.

**Resumo**

Este relatório descreve o trabalho prático realizado na UC de Inteligência Artificial, envolvendo a análise e desenvolvimento de pesquisa num determinado armazém automático, onde este armazém tem um ponto de início onde existe um robot para coletar um produto e depois de coletar este produto o robot tem de ir para a saída. O objetivo foi aplicar os conceitos teóricos e desenvolver habilidades práticas lecionada na UC.

O trabalho prático permitiu a aplicação dos conhecimentos teóricos e destacou a importância da Inteligência Artificial bem como a implementação de vários algoritmos como a perceção de uso. Assim como consolidar os conhecimentos adquiridos ao longo da UC.

**Palavras-Chaves:** Inteligência Artificial, Agentes, Armazéns Automáticos, *Python*, Algoritmos de pesquisa

**Abstract**

*This report describes the practical work carried out in the AI course, involving analysis and research development in a certain imaginary warehouse, where this warehouse has a starting point where there is a robot to collect a product and after collecting this product the robot has to go to the exit. The objective was to apply theoretical concepts and develop practical skills taught in the course. The practical work allowed the application of theoretical knowledge and highlighted the importance of Artificial Intelligence as well as the implementation of various algorithms as a perception of use. As well as consolidating the knowledge acquired throughout the course.*

***Keywords****: Artificial Intelligence, Agents, Automated Warehouses, Python, Search Algorithms*

**Índice**

[Índice de Figuras 9](#_Toc152432138)

[Índice de Tabelas 10](#_Toc152432139)

[Lista de siglas e acrónimos 11](#_Toc152432140)

[1. Introdução 12](#_Toc152432141)

[1.1. Enquadramento 12](#_Toc152432142)

[1.2. Objetivos 12](#_Toc152432143)

[2. Algoritmos 13](#_Toc152432144)

[2.1. Algoritmo de Busca em Largura (*BFS*) 13](#_Toc152432145)

[2.1.1. Definição 13](#_Toc152432146)

[2.1.2. Propriedades: 13](#_Toc152432147)

[2.2. A Estrela 14](#_Toc152432148)

[2.2.1. Propriedades do Algoritmo A\*: 14](#_Toc152432149)

[*2.3.* *Greedy Search* 15](#_Toc152432150)

[*2.3.1.* Propriedades do Algoritmo *Greedy Search* 15](#_Toc152432151)

[3. Armazéns 16](#_Toc152432152)

[4. Desenvolvimento 17](#_Toc152432153)

[4.1. Matriz para simular Armazém 17](#_Toc152432154)

[4.2. Algoritmo BFS 18](#_Toc152432155)

[4.3. Algoritmo A Estrela 20](#_Toc152432156)

[4.4. Algoritmo *Gridy Search* 22](#_Toc152432157)

[4.5. Interface Gráfica 23](#_Toc152432158)

[4.5.1. Interface Gráfica A\* 24](#_Toc152432159)

[4.5.2. Interface Greedy Search 25](#_Toc152432160)

[4.5.3. Interface Gráfica *BFS* 26](#_Toc152432161)

[5. Resultados 27](#_Toc152432162)

[5.1. Resultados Matrizes 10x10 29](#_Toc152432163)

[5.2. Resultados Matrizes 25x25 32](#_Toc152432164)

[5.3. Resultados Matrizes 48x52 35](#_Toc152432165)

[5.4. Resultados Matrizes 50x50 38](#_Toc152432166)

[5.5. Resultados Matrizes 75x75 41](#_Toc152432167)

[6. Analise dos Resultados 44](#_Toc152432168)

[6.1. Avaliação Geral 44](#_Toc152432169)

[6.2. Análise Específica por Dimensões 45](#_Toc152432170)

[7. Pros e Contras 46](#_Toc152432171)

[8. Conclusão 47](#_Toc152432172)

[9. Bibliografia 48](#_Toc152432173)

# Índice de Figuras

[Figura 1 - Interface Gráfica A\*- Robô para Produto 24](#_Toc152432121)

[Figura 2 - Interface Gráfica A\*- Produto para Output 24](#_Toc152432122)

[Figura 3 - Interface Gráfica *Greedy Search* \*- Robô para Produto 25](#_Toc152432123)

[Figura 4 - Interface Gráfica *Greedy Search* \*- Produto para Output 25](#_Toc152432124)

[Figura 5 - Interface Gráfica *BFS* - Robô para Produto 26](#_Toc152432125)

[Figura 6 - Interface Gráfica *BFS* - Produto para Output 26](#_Toc152432126)

[Figura 7 - Resultados Matrizes 10x10 29](#_Toc152432127)

[Figura 8 - Resultados Matrizes 25x25 32](#_Toc152432128)

[Figura 9 - Resultados Matrizes 48x52 35](#_Toc152432129)

[Figura 10 - Resultados Matrizes 50x50 38](#_Toc152432130)

[Figura 11 - Resultados Matrizes 75x75 41](#_Toc152432131)

# Índice de Tabelas

[Tabela 1 - Armazéns Simulados 27](#_Toc152432132)

[Tabela 2 – Tempo Total Médio de cada Algoritmo em Matrizes 10x10 31](#_Toc152432133)

[Tabela 3 - Tempo Total Médio de cada Algoritmo em Matrizes 25x25 34](#_Toc152432134)

[Tabela 4 - Tempo Total Médio de cada Algoritmo em Matrizes 48x52 37](#_Toc152432135)

[Tabela 5 - Tempo Total Médio de cada Algoritmo em Matrizes 50x50 40](#_Toc152432136)

[Tabela 6 - Tempo Total Médio de cada Algoritmo em Matrizes 75x75 43](#_Toc152432137)

# Lista de siglas e acrónimos

* AI: *Artificial Inteligence*;
* UC: Unidade Curricular
* *BFS*: *Breadth-First Search*

# Introdução

## Enquadramento

No âmbito da UC de Inteligência Artificial, aprender sobre algoritmos de busca é crucial para entender como construir sistemas inteligentes. Estes algoritmos são fundamentais para a compreensão de como as máquinas podem imitar processos de raciocínio humano, especialmente em tarefas que envolvem resolução de problemas e tomada de decisões.

Os algoritmos de busca, como o Busca em Largura (*BFS*), A Estrela (A\*) e *Greedy Search*, exemplificam a aplicação de conceitos de Inteligência Artificial na solução de problemas complexos de navegação e otimização. No contexto de um armazém automatizado, estes algoritmos desempenham um papel crucial na orientação de robôs para a execução eficiente de tarefas de coleta e entrega, uma aplicação prática e altamente relevante dos princípios de IA.

Este estudo contribui para o entendimento teórico de algoritmos de busca dentro do campo da Inteligência Artificial, mas também demonstra como esses conceitos são aplicados em cenários do mundo real. A análise do desempenho desses algoritmos em simulações de ambientes de armazém oferece insights valiosos sobre sua eficácia, limitações e potencial para futuras inovações na indústria de automação e logística.

Ao integrar teoria e prática, este trabalho alinha-se com os objetivos centrais da cadeira de Inteligência Artificial, proporcionando uma compreensão profunda tanto dos fundamentos teóricos quanto das aplicações práticas dos algoritmos de busca.

## 1.2. Objetivos

Compreender a aplicação dos algoritmos de busca em largura (*BFS*), A Estrela (A\*), e *Greedy Search* em ambientes de armazém automatizado.

Analisar a eficácia destes algoritmos em diferentes cenários de simulação de armazém, com variações no tamanho da matriz e nos níveis de obstáculos.

Comparar o desempenho desses algoritmos em termos de eficiência, tempo de execução, e precisão na navegação de um robô automatizado.

Explorar as limitações e potenciais melhorias desses métodos em ambientes de armazéns reais.

# Algoritmos

## Algoritmo de Busca em Largura (*BFS*)

### Definição

A busca em largura (*BFS*), é um algoritmo para percorrer ou pesquisar estruturas de dados em árvore ou grafo. O algoritmo começa no nó raiz (ou qualquer nó específico em um grafo) e explora todos os vizinhos desse nó antes de passar para os nós no próximo nível de profundidade. Ele visita os nós em camadas ou níveis, garantindo que todos os eles de um determinado nível sejam explorados antes de passar para o próximo. A BFS é útil para encontrar o caminho mais curto em grafos não ponderados e para travessias que precisam processar nós em ordem de proximidade ao nó inicial.

### Propriedades:

* **Ótimo para Grafos Não Ponderados:** Em grafos não ponderados, o BFS garante encontrar o caminho mais curto entre o nó inicial e qualquer outro nó no grafo, pois ele explora todos os nós igualmente em cada nível antes de passar para o próximo.
* **Completo:** O BFS é completo, o que significa que se houver uma solução, o algoritmo a encontrará. Ele explora sistematicamente todos os nós e caminhos possíveis em um grafo finito.
* **Complexidade de Espaço:** A complexidade de espaço do BFS é proporcional à largura máxima do grafo. Ele precisa armazenar todos os nós de um nível para acessar os do próximo, o que é feito através de um vetor. Portanto, em grafos largos, o BFS pode consumir uma quantidade significativa de memória.

## A Estrela

O Algoritmo A\* é uma técnica de busca amplamente usada em algoritmia para encontrar o caminho mais curto entre um ponto inicial e um ponto final em um grafo.

### Propriedades do Algoritmo A\*:

* **Ótimo e Completo (com a Heurística Adequada):** O algoritmo A\* é ótimo e completo se a função heurística usada for admissível, ou seja, nunca superestima o custo real para alcançar o objetivo. Isso garante que o A\* encontre o caminho mais curto em um espaço de busca finito.
* **Heurística:** O coração do A\* é sua função heurística, que estima o custo do caminho mais barato do nó atual para o objetivo.
* **Complexidade de Espaço:** A\* mantém todos os caminhos não explorados na memória. Em espaços de busca grandes, isso pode levar a um alto consumo de memória. A complexidade de espaço é, na pior das hipóteses, exponencial em relação ao comprimento do caminho.
* **Complexidade de Tempo:** A complexidade de tempo depende fortemente da heurística. Na pior das hipóteses, pode ser exponencial, especialmente se a heurística não for eficaz. Com uma boa heurística, no entanto, A\* é significativamente mais rápido do que algoritmos de busca sem informação.
* **Balanceamento entre Exploração e Eficiência:** A\* equilibra a busca que se move rapidamente em direção ao objetivo, e o método de menor custo, que explora caminhos de baixo custo. Este balanceamento é alcançado através da heurística.

## *Greedy Search*

Um algoritmo *Greedy Search* é uma abordagem para resolver um problema selecionando a melhor opção disponível no momento. Não se preocupa se o melhor resultado atual trará o resultado global ótimo.

### Propriedades do Algoritmo *Greedy Search*

**Não Ótimo:** O *Greedy Search* não garante encontrar a solução mais ótima para um problema. Ele faz escolhas que parecem ser as melhores no momento, sem considerar o custo acumulado ou o impacto futuro dessas escolhas.

**Não Completo:** Este algoritmo não é completo, o que significa que não há garantia de que ele sempre encontrará uma solução, se ela existir. Em cenários com múltiplos caminhos ou obstáculos, pode falhar em encontrar um caminho até o objetivo.

**Dependência da Heurística:** A eficácia do *Greedy Search* é altamente dependente da qualidade da função heurística utilizada. Uma heurística apropriada pode conduzir rapidamente ao objetivo, enquanto uma heurística inadequada pode levar a resultados não atinge a mais alta qualidade ou falhas.

**Eficiência em Termos de Espaço e Tempo:** O *Greedy Search* é geralmente eficiente em termos de uso de memória e tempo de execução.

**Aplicabilidade Prática Limitada:** Embora o *Greedy Search* seja útil em alguns cenários específicos, sua aplicabilidade é limitada em problemas complexos, onde a escolha mais promissora a curto prazo não necessariamente leva ao melhor resultado global.

# Armazéns

Neste relatório, exploramos a aplicação de algoritmos de busca em um ambiente de armazém automatizado, onde um robô é utilizado para realizar tarefas de coleta e entrega de produtos. A eficiência e eficácia do robô em navegar pelo armazém são fundamentais para otimizar as operações logísticas. Para isso, modelamos o ambiente do armazém como uma matriz, onde cada célula representa uma área específica, variando de espaços transitáveis a obstáculos.

**Componentes do Armazém:**

* **Robô (R):** Localizado inicialmente em uma posição específica da matriz, o robô é a unidade móvel encarregada de coletar e entregar produtos.
* **Espaço Livre (.):** Representa as áreas livres do armazém por onde o robô pode mover-se, permitindo-lhe traçar rotas para alcançar diferentes pontos.
* **Obstáculos (#):** Simulam paredes ou pilares no armazém, criando barreiras físicas. Estes obstáculos são intransponíveis para o robô, requerendo que ele planeje rotas alternativas para alcançar seu destino.
* **Produto (P):** Localização de um produto que o robô deve coletar. Este ponto é o destino inicial do robô, onde ele deve chegar para realizar a coleta.
* **Ponto de Entrega (O):** Após a coleta do produto, o robô deve levar o item até este ponto para completar a entrega.

**Aplicação dos Algoritmos de Busca:**

Os algoritmos de busca como DFS, BFS e A\* são implementados para guiar o robô pelo armazém.

O objetivo é encontrar o caminho mais eficiente do ponto inicial (R) até o produto (P), e em seguida, do produto até o ponto de entrega (O), considerando os obstáculos presentes.

# Desenvolvimento

## Matriz para simular Armazém

A matriz serve como uma representação simplificada de um armazém, onde cada célula corresponde a uma área específica dentro do espaço físico do armazém. Esta representação é fundamental para simular e analisar o movimento do robô dentro deste ambiente.

* **Criação da Matriz:**

A matriz é inicialmente criada como uma matriz de células vazias, representadas por pontos ('.'). Cada ponto simboliza um espaço livre por onde o robô pode se mover.

* **Definição de Percentagem de Obstáculos**:

A quantidade de obstáculos é determinada por uma percentagem definida, assegurando que um certo número de células na matriz seja convertido em obstáculos.

* **Adição de Obstáculos**

Obstáculos são distribuídos aleatoriamente pela matriz para simular obstáculos no armazém. Estes são representados por '#' e indicam áreas pelas quais o robô não pode passar.

* **Posicionamento de Elementos-Chave:**

**Robô (R):** Uma célula é designada como o ponto de partida do robô.

**Produto (P):** Um local na matriz é escolhido aleatoriamente para representar onde o produto está localizado.

**Ponto de Entrega (O):** Um ponto é estabelecido como o destino final do robô, após a coleta do produto.

* **Aleatoriedade e Realismo:**

A posição dos obstáculos e do produto é aleatória, imitando a natureza imprevisível e variável de um armazém real.

* **Processo de Geração:**

A matriz é gerada automaticamente com estes critérios, utilizando algoritmos para assegurar a distribuição aleatória dos obstáculos e a colocação dos elementos-chave.

## Algoritmo BFS

O Algoritmo de Busca em Largura (BFS) é utilizado para orientar o movimento de um robô em um armazém simulado. O objetivo é que o robô encontre e siga o caminho mais eficiente para coletar um produto (P) e, em seguida, entregá-lo em um ponto de saída (O), navegando por um espaço repleto de obstáculos (#).

1. **Implementação e Funcionamento:**

Uma interface gráfica mostra o armazém, com cores distintas marcando o robô, o produto, o ponto de saída, os caminhos explorados e os obstáculos. Isso proporciona uma visualização clara do ambiente e da trajetória do robô.

1. **Ponto de Partida:**

O *BFS* começa em um nó específico do grafo, que é geralmente referido como o nó raiz ou inicial. Este nó serve como o ponto de partida para a busca.

1. **Inicialização de Estruturas de Dados:**

Para rastrear a progressão da busca, o BFS utiliza um vetor. Todos os nós visitados são armazenados nesse vetor.

1. **Exploração do Primeiro Nível:**

O nó inicial é marcado como visitado e adicionado ao vetor. Em seguida, o algoritmo explora todos os vizinhos diretos desse nó.

1. **Passagem para Níveis Subsequentes:**

Uma vez que todos os vizinhos do nó inicial são explorados e adicionados ao vetor, o algoritmo remove o nó inicial da fila e passa para o próximo nó no vetor.

Este processo é repetido para explorar todos os vizinhos não visitados do nó atual e adicioná-los ao vetor.

A busca continua seguindo a ordem dos nós na fila, garantindo que os eles sejam explorados nível por nível.

1. **Continuação até Esgotar os Nós ou chegar ao Objetivo:**

O processo continua até que a vetor esteja vazio ou chegue ao objetivo.

Durante a busca, cada nó é visitado exatamente uma vez, assegurando uma procura completa e sistemática.

1. **Reconstrução do Caminho:**

Uma vez que o objetivo é alcançado, o algoritmo reconstrói o caminho do nó objetivo de volta ao nó inicial.

## Algoritmo A Estrela

No cenário do armazém automatizado, o algoritmo A Estrela é utilizado para guiar um robô na coleta e entrega de produtos. O armazém é simulado por uma matriz, onde o robô deve encontrar o caminho mais eficiente até um produto (P), e depois até um ponto de saída (O), superando obstáculos representados por '#' na matriz.

1. **Implementação e Funcionamento:**

Uma interface gráfica mostra o armazém, com cores distintas marcando o robô, o produto, o ponto de saída, os caminhos explorados e os obstáculos. Isso proporciona uma visualização clara do ambiente e da trajetória do robô.

1. **Inicialização:**

O algoritmo começa com um conjunto aberto que contém apenas o nó inicial e um conjunto fechado vazio.

Cada nó mantém três valores importantes: o custo real do caminho do nó inicial até ele (g), a estimativa heurística do custo para alcançar o objetivo, onde é calculado o caminho de *Manhattan* (h) e a soma destes dois valores (f = g + h).

1. **Exploração de Nós:**

Em cada etapa, o algoritmo escolhe o nó com o menor valor de f no conjunto aberto.

Este nó é removido do conjunto aberto e adicionado ao conjunto fechado.

1. **Expansão do Nó Atual:**

O algoritmo expande o nó selecionado, examinando todos os seus vizinhos.

Para cada vizinho, o algoritmo calcula o valor de g (custo do caminho do nó inicial até o vizinho), h (caminho de *Manhattan)* e atualiza o valor de f.

1. **Atualização do Conjunto Aberto:**

Se um vizinho não está nem no conjunto aberto nem no fechado, ele é adicionado ao conjunto aberto.

Se o vizinho já está em um dos conjuntos, mas o novo caminho é melhor (menor g), o caminho até esse vizinho é atualizado.

1. **Repetição do Processo:**

O algoritmo repete esse processo de seleção, expansão e atualização até que o conjunto aberto esteja vazio ou até que o objetivo seja alcançado.

1. **Verificação de Objetivo:**

Se o nó selecionado for o nó objetivo, o algoritmo reconstrói o caminho até o nó inicial e termina.

1. **Reconstrução do Caminho:**

Uma vez que o objetivo é alcançado, o algoritmo reconstrói o caminho do nó objetivo de volta ao nó inicial, seguindo os caminhos de menor custo registados.

## Algoritmo *Gridy Search*

No ambiente simulado de um armazém, o algoritmo *Greedy Search* é empregue para guiar um robô na tarefa de localizar e coletar um produto, e posteriormente transportá-lo para um ponto de saída. O armazém é mapeado por uma matriz, com obstáculos, caminhos e locais de interesse claramente definidos.

1. **Implementação e Funcionamento:**

Uma interface gráfica mostra o armazém, com cores distintas marcando o robô, o produto, o ponto de saída, os caminhos explorados e os obstáculos. Isso proporciona uma visualização clara do ambiente e da trajetória do robô.

1. **Ponto de Partida:**

O algoritmo inicia a partir de um nó específico, considerado o ponto de partida da procura.

1. **Decisão Baseada em Heurística:**

Em cada passo, o algoritmo seleciona o próximo nó para explorar com base em uma função heurística. Essa função avalia qual opção parece levar mais rapidamente ao objetivo, sem preocupar-se com o caminho percorrido até então.

1. **Avanço Direcionado:**

O algoritmo avança para o nó escolhido, baseando-se unicamente que ele oferece em direção ao objetivo. Ele não reconsidera as escolhas anteriores, mantendo-se focado no caminho à frente.

1. **Procura pelo Objetivo:**

A busca continua seguindo a orientação da heurística, até que o objetivo seja encontrado ou até que fique claro que não há caminho viável para o objetivo a partir do trajeto escolhido.

1. **Conclusão da Busca:**

Uma vez que o objetivo é alcançado, o algoritmo reconstrói o caminho do nó objetivo de volta ao nó inicial, seguindo os caminhos de menor custo registados.

## Interface Gráfica

A trajetória encontrada pelos algoritmos é claramente visível em verde, mostrando a rota do ponto inicial (azul) ao ponto final (rosa). Esta representação visual é útil para confirmar a validade e eficiência do caminho encontrado.

A eficiência dos algoritmos também é evidente na forma como a trajetória parece tomar decisões direcionais ótimas, evitando obstáculos (pretos) e minimizando a distância total percorrida.

As áreas que foram exploradas durante a busca, mas que não fazem parte do caminho final, estão marcadas em vermelho. Isso demonstra o processo de busca dos algoritmos, onde várias rotas potenciais são consideradas antes de se chegar à solução final.

A densidade de áreas vermelhas pode dar uma ideia da complexidade do espaço de busca e da eficiência do algoritmo em descartar caminhos não ótimos rapidamente.

A ausência de áreas vermelhas excessivas ao longo do caminho verde indica que os algoritmos não desperdiçaram recursos a explorar as áreas desnecessárias do espaço de busca.

Esta representação visual é extremamente útil na otimização de rotas, simulando assim aplicações do mundo real.

### Interface Gráfica A\*

Uma imagem com texto, coser, puzzle de palavras-cruzadas, padrão

Descrição gerada automaticamente

Figura - Interface Gráfica A\*- Robô para Produto

Uma imagem com texto, captura de ecrã, Saturação de cores, diagrama

Descrição gerada automaticamente

Figura - Interface Gráfica A\*- Produto para Output

### Interface Greedy Search

Uma imagem com puzzle de palavras-cruzadas, texto, coser

Descrição gerada automaticamente

Figura - Interface Gráfica *Greedy Search* \*- Robô para Produto

Uma imagem com texto, puzzle de palavras-cruzadas, diagrama, file

Descrição gerada automaticamente

Figura - Interface Gráfica *Greedy Search* \*- Produto para Output

### Interface Gráfica *BFS*

Uma imagem com coser, texto, puzzle de palavras-cruzadas, padrão

Descrição gerada automaticamente

Figura - Interface Gráfica *BFS* - Robô para Produto

Uma imagem com Saturação de cores, captura de ecrã, padrão, encarnado

Descrição gerada automaticamente

Figura - Interface Gráfica *BFS* - Produto para Output

# Resultados

Foram criadas 72 matrizes para simular os armazéns e avaliar o desempenho de cada algoritmo. As matrizes criadas são do tipo:

|  |  |  |
| --- | --- | --- |
| Número de Matrizes | Tamanho da Matriz | Tipo de Matriz |
| 5 | 10x10 | Nível Baixo de Obstáculos |
| 25x25 | Nível Baixo de Obstáculos |
| 48x52 | Nível Baixo de Obstáculos |
| 50x50 | Nível Baixo de Obstáculos |
| 75x75 | Nível Baixo de Obstáculos |
| 10x10 | Nível Médio de Obstáculos |
| 25x25 | Nível Médio de Obstáculos |
| 48x52 | Nível Médio de Obstáculos |
| 50x50 | Nível Médio de Obstáculos |
| 75x75 | Nível Médio de Obstáculos |
| 10x10 | Nível Alto de Obstáculos |
| 25x25 | Nível Alto de Obstáculos |
| 48x52 | Nível Alto de Obstáculos |
| 50x50 | Nível Alto de Obstáculos |
| 75x75 | Nível Alto de Obstáculos |

Tabela - Armazéns Simulados

Foi criado um procedimento para validar o desempenho de cada algoritmo em cada um destes armazéns. Para validar estes procedimentos foram retiradas várias métricas como, media, mediana, desvio padrão, máximo e mínimo, para verificar qual a performance de todos os algoritmos.

Os dados foram separados da seguinte forma:

* + Custo de Robô para Produto (onde é calculado em unidades de tempo);
  + Custo de Produto para Output (onde é calculado em unidades de tempo);
  + Custo de Robô para Produto em segundos (estimando que cada casa o robô demora 0.1segundos a percorrer);
  + Custo de Produto para Output em segundos (estimando que cada casa o robô demora 0.1segundos a percorrer);
  + Tempo de procura do caminho do Robô para Produto em segundos;
  + Tempo de procura do caminho do Produto para Output em segundos;

## Resultados Matrizes 10x10



Figura - Resultados Matrizes 10x10

|  |  |  |  |
| --- | --- | --- | --- |
|  | BFS | *Greedy Search* | A\* |
| Tempo Total Médio | 2,537442446 | 2,49467907 | 2,387797281 |

Tabela – Tempo Total Médio de cada Algoritmo em Matrizes 10x10

## Resultados Matrizes 25x25



Figura - Resultados Matrizes 25x25

|  |  |  |  |
| --- | --- | --- | --- |
|  | BFS | *Greedy Search* | A\* |
| Tempo Total Médio | 6,302886602 | 6,780658817 | 6,070904078 |

Tabela - Tempo Total Médio de cada Algoritmo em Matrizes 25x25

## Resultados Matrizes 48x52



Figura - Resultados Matrizes 48x52

|  |  |  |  |
| --- | --- | --- | --- |
|  | BFS | *Greedy Search* | A\* |
| Tempo Total Médio | 13,08884541 | 15,04398451 | 12,37767164 |

Tabela - Tempo Total Médio de cada Algoritmo em Matrizes 48x52

## Resultados Matrizes 50x50



Figura - Resultados Matrizes 50x50

|  |  |  |  |
| --- | --- | --- | --- |
|  | BFS | *Greedy Search* | A\* |
| Tempo Total Médio | 13,24809088 | 15,08395625 | 12,23072351 |

Tabela - Tempo Total Médio de cada Algoritmo em Matrizes 50x50

## Resultados Matrizes 75x75



Figura - Resultados Matrizes 75x75

|  |  |  |  |
| --- | --- | --- | --- |
|  | BFS | *Greedy Search* | A\* |
| Tempo Total Médio | 18,65401435 | 20,93695713 | 17,26915387 |

Tabela - Tempo Total Médio de cada Algoritmo em Matrizes 75x75

# Analise dos Resultados

A análise comparativa permitirá entender a eficácia e a eficiência de cada algoritmo em termos de custo do caminho, tempo de execução e adequação a diferentes densidades de obstáculos.

## Avaliação Geral

* **BFS:**
  + Mostrou-se bastante consistente em todas as dimensões, com tempos de procura geralmente baixos.
  + No entanto, os custos dos caminhos tendem a ser mais altos, indicando que o BFS pode não ser o melhor em termos de otimização de caminho, mas sim em termos de confiabilidade e velocidade.
* ***Greedy Search:***
  + Demonstrou maior variabilidade em custos de caminho e tempo de execução, o que sugere que pode ser sensível à configuração específica da matriz.
  + Em matrizes maiores, o *Greedy Search* apresentou um aumento significativo tanto no custo do caminho quanto no tempo total de execução, possivelmente devido à sua natureza de optar pela escolha localmente ótima sem considerar o custo total do caminho.
* **A\*:**
  + Apresentou um equilíbrio notável entre o custo do caminho e o tempo de execução.
  + Foi particularmente eficiente em matrizes de maior densidade, sugerindo que a sua abordagem heurística está bem ajustada para lidar com complexidades adicionais no ambiente.

## Análise Específica por Dimensões

* **Matrizes 10x10:**
  + O A\* teve o menor tempo total de execução e apresentou consistentemente bons resultados tanto no custo quanto no tempo, o que sugere uma alta eficiência em espaços pequenos.
  + O BFS mostrou tempos de execução rápidos, mas com custos de caminho ligeiramente maiores.
* **Matrizes 25x25 e 48x52:**
  + A diferença de desempenho entre o BFS e o A\* começa a ser mais evidente, com o A\* mantendo custos de caminho mais baixos e tempos de execução competitivos mesmo à medida que o tamanho da matriz aumenta.
  + O *Greedy Search* começa a mostrar uma degradação no desempenho, com tempos de execução e custos de caminho cada vez mais altos.
* **Matrizes 50x50 e 75x75:**
  + O A\* demonstra uma capacidade impressionante de escalar com o tamanho da matriz, mantendo a eficiência no custo do caminho e no tempo de execução.
  + O *BFS*, apesar de manter a confiabilidade, começa a sofrer com o aumento do custo do caminho.
  + O *Greedy Search* revela sua maior desvantagem no aumento do tamanho das matrizes, com tempos de execução baixos, mas custos de caminho mais altos.

# Pros e Contras

O algoritmo A\* mostrou-se o mais versátil e eficiente em todas as dimensões, equilibrando com sucesso o custo do caminho e o tempo de execução. A consistência nos custos de caminho e nos tempos de procura pode indicar uma robustez em diferentes configurações do armazém.

O algoritmo *BFS* tende a ter um desempenho uniforme e previsível, mas pode não ser o mais eficiente em termos de custo do caminho. Este algoritmo é recomendável para ambientes onde a velocidade é mais crítica do que a otimização do caminho e onde os recursos computacionais são limitados.

O algoritmo *Greedy Search* pode ser útil em cenários onde as soluções aproximadas são aceitáveis. Apresenta uma maior variabilidade nos resultados, indicando uma sensibilidade a configurações específicas.

# Conclusão

A realização do trabalho mostrou-se, especialmente importante, na medida em que, nos foi possível consolidar os conhecimentos adquiridos na unidade curricular lecionada ao longo do 1º semestre, e adquirir outros, durante o desenvolvimento do trabalho prático acima detalhado.

Relativamente aos pontos propostos para o trabalho prático, consideramos que todos eles foram concretizados com sucesso, com isto referimo-nos ao cumprimento e resolução de todos os pressupostos para a realização do mesmo, incluindo os respetivos testes pedidos assim como o uso das ferramentas necessárias para comprovar os mesmos.

# Bibliografia

1. <https://github.com/Zav04/AI_PROJECT.git>

<consultado a 2-12-2023>

1. <https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/>

<consultado a 12-10-2023>

1. <https://www.geeksforgeeks.org/a-search-algorithm/>

<consultado a 20-10-2023>

1. <https://www.geeksforgeeks.org/greedy-algorithms/>

<consultado a 15-01-2023>