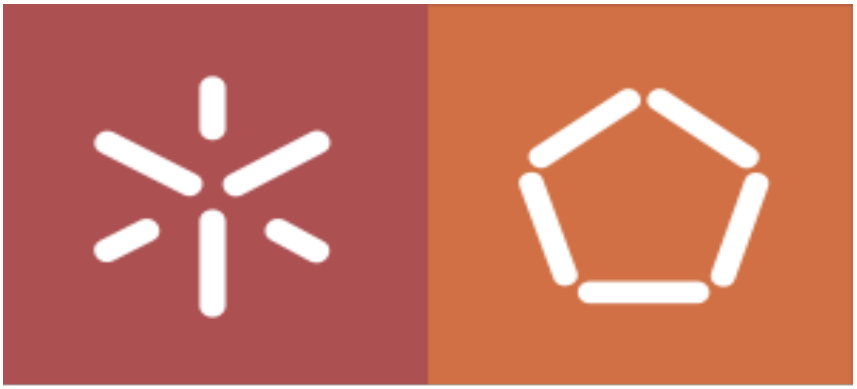

Universidade do Minho
Escola de Engenharia

Inteligência Artificial 2024/2025

Trabalho Prático – Resolução de Problemas - Algoritmos de procura

Trabalho realizado por:

Diogo Silva a104183 Miguel Barrocas a104272

Olavo Carreira a104526 João Pinto a104270

--- Grupo 44 ---

Avaliação Grupo

Olavo Carreira = 0

Diogo Silva = 0
João Pinto = 0

Miguel Barrocas = 0

Índice

1. Introdução

Este trabalho foi realizado no âmbito da unidade curricular de Inteligência Artificial da Licenciatura em Engenharia Informática da Universidade do Minho, e teve como objetivo a resolução de problemas através da conceção e implementação de algoritmos de procura abordados nas aulas. Neste trabalho foram implementadas estratégias para a resolução de problemas com o uso de algoritmos de procura, partindo da formulação do problema em questão.

2. Formulação do problema

Uma vez que o problema se encontra num ambiente determinístico e completamente observável, em que o agente "sabe" exatamente o estado em que está e as soluções pretendidas são sequências, podemos afirmar que este é um problema de estado único. Além disso, a sua formulação pode ser efetuada da seguinte forma:

- **Representação do estado:** Grafo não orientado, em que cada nodo representa uma posição no circuito e cada aresta representa o trajeto ente posições.
- **Estado inicial:** A posição inicial representa o posto de onde o veículo parte.
- **Estado objetivo:** As zonas de entrega identificadas como F1, F2 e F3.
- **Operadores:** Movimentos permitidos no labirinto, dependendo do tipo de terreno e capacidades do veículo.
- **Solução:** Um caminho válido (sequência de posições percorridas) que comece na posição inicial e termine numa das posições finais.
- **Custo da solução:** Distância total do percurso feito pelo agente.

3. Representação do Ambiente

```
# Define maze layout
labyrinth = [
    ['#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#'],
    ['I', ' ', '#', 'T', 'T', '#', ' ', 'T', 'T', ' ', ' ', ' ', ' ', 'F1'],
    [' ', 'T', 'T', 'T', '#', 'T', ' ', ' ', 'T', 'T', ' ', ' ', ' ', ' '],
    ['#', 'T', 'T', ' ', ' ', 'T', 'T', ' ', '#', 'A', 'A', 'A', 'A', ' '],
    ['#', ' ', 'T', 'A', 'A', ' ', ' ', ' ', '#', 'A', 'A', 'A', ' ', ' '],
    ['#', 'A', 'A', 'A', 'A', 'A', ' ', ' ', ' ', '#', 'A', ' ', ' ', ' '],
    ['#', 'A', 'A', '#', ' ', ' ', '#', ' ', ' ', ' ', '#', 'A', ' ', ' '],
    ['#', 'A', 'A', '#', ' ', ' ', '#', ' ', '#', ' ', 'A', 'A', 'A', ' '],
    ['#', 'A', '#', '#', ' ', 'T', 'T', ' ', ' ', 'A', 'A', ' ', '#', '#'],
    ['#', ' ', 'A', 'A', 'A', ' ', 'T', ' ', 'T', 'T', 'A', ' ', '#', '#'],
    ['#', ' ', 'A', 'A', 'A', ' ', 'T', ' ', 'T', 'T', 'A', 'T', '#', '#'],
    ['#', ' ', 'A', 'A', 'A', ' ', '#', ' ', ' ', 'A', 'A', 'T', '#', '#'],
    ['#', ' ', 'F3', '#', ' ', ' ', '#', ' ', 'A', ' ', ' ', 'T', 'T', 'F2'],
    ['#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#']
]
```

Figura 1 - Labirinto

3.1. O Labirinto

O ambiente é representado por uma matriz bidimensional onde:

- **‘#’** representa obstáculos (não transponíveis);
- **‘I’** é o ponto inicial;
- **‘F1’, ‘F2’, ‘F3’** são os pontos finais das zonas de entrega;
- **‘ ’** é o terreno regular;
- **‘A’** é terreno aquático (apenas barcos podem atravessar);
- **‘T’** é terreno acidentado (apenas camiões podem atravessar).

3.2. Condições Meteorológicas

O sistema inclui três estados climáticos:

- **Céu limpo:** Sem impacto nas operações.
- **Chuva:** Reduz a velocidade dos veículos em 30%.
- **Tempestade:** Reduz a velocidade em 60% e pode bloquear zonas.

3.3. Veículos

Três tipos de veículos são utilizados:

- **Carro:** Atravessa apenas terreno regular, velocidade de 80 km/h.
- **Camião:** Atravessa terrenos regular e acidentado, velocidade de 60 km/h.
- **Barco:** Atravessa apenas terreno aquático, velocidade de 50 km/h.

Cada veículo possui capacidade de carga, limite de combustível e consumo baseados no peso e distância percorrida.

4. Funcionamento do programa

4.1. Criação do grafo

O grafo é construído a partir do labirinto, onde cada posição transitável é convertida em um nodo do grafo. O processo de construção segue os seguintes passos:

1. Análise de posições: Para cada coordenada (i,j) do labirinto, o programa verifica se algum veículo pode atravessar aquela posição. Uma posição é considerada transitável se:

- For terreno regular (' '): acessível por carros e caminhões
- For terreno aquático ('A'): acessível por barcos
- For terreno acidentado ('T'): acessível por caminhões
- Não for uma parede ('#'): inacessível por qualquer veículo

2. Criação de nodos: Cada posição transitável é convertida num nodo do grafo, mantendo as suas coordenadas originais para referência.

3. Estabelecimento de arestas: Para cada nodo criado, o programa verifica as quatro posições adjacentes (norte, sul, leste, oeste) e cria arestas entre nodos vizinhos quando:

- A posição adjacente existe no labirinto
- A posição adjacente é transitável por algum veículo
- Não há parede separando as posições

4. Validação especial:

- O ponto inicial ('I') é sempre incluído como nodo

- Os pontos finais ('F1', 'F2', 'F3') são incluídos como nodos se forem acessíveis

- As arestas são não direcionadas, permitindo movimento em ambos os sentidos

Este processo resulta num grafo não direcionado onde:

- Vértices representam posições válidas no labirinto

- Arestas representam movimentos possíveis entre posições adjacentes

- A estrutura preserva todas as restrições de movimento do labirinto original

- As capacidades específicas de cada tipo de veículo são consideradas na navegação

O grafo resultante serve como base para todos os algoritmos de procura implementados no sistema, permitindo a identificação de rotas válidas, considerando as características específicas de cada veículo e tipo de terreno.

4.2. Menu

```
Escolha uma opcao:
1 - Encontrar rota completa com BFS
2 - Encontrar rota completa com DFS
3 - Encontrar rota completa com Greedy
4 - Encontrar rota completa com A*
5 - Encontrar rota completa com Dijkstra
6 - Visualizar grafo de procura
7 - Atualizar condicoes meteorologicas
8 - Ver status dos veiculos
9 - Ver status das zonas
10 - Sair
Digite o numero da opcao:
```

Figura 2 - Menu Inicial

O Menu apresenta as seguintes opções:

1-5. Encontrar rota completa com BFS/DFS/Greedy/A*/Dijkstra

6. Visualização grafo de procura

7. Atualizar condições meteorológicas

8. Ver status dos veículos

9. Ver status das zonas

10. Sair

Haverá 4 opções de algoritmos de procura, 2 algoritmos de procura não informados e 2 algoritmos de procura informada

5. Estratégias de procura não informada

As estratégias de procura não informada implementadas neste projeto foram a **Procura em profundidade (DFS)** e a **Procura em Largura (BFS)**.

5.1. Procura de Profundidades (DFS)

Este tipo de procura usa como estratégia, expandir sempre um dos nodos mais profundos do grafo.

As **vantagens** deste tipo de procura são:

- Pouco uso de memória.
- Bom para problemas com múltiplas soluções, pois a probabilidade de estar a procurar por um caminho possível aumenta.

A **desvantagem** deste tipo de procura são:

- Pouca eficiência em grafos com uma profundidade elevada e poucas soluções.
- A solução obtida pode não ser a solução ótima.

No entanto, no nosso problema o circuito terá vários caminhos possíveis para chegar ao destino final, logo esta estratégia irá conseguir obter um percurso válido e de forma relativamente eficiente. A nossa implementação do algoritmo de procura em profundidade impede que os nodos que já tenham sido visitados sejam explorados pelo algoritmo de maneira a evitar a ocorrência de loops no processo de pesquisa. Esta estratégia de procura apresenta as seguintes propriedades:

- Tempo: $O(b\ m\ )$
- Espaço: $O(bm)$

Onde b é o número máximo de sucessores de um nodo da árvore de procura e m a máxima profundidade do espaço de estados.

5.1.1 Exemplo de procura de profundidades (DFS)

```
Condições Meteorológicas Atuais: CLEAR
Zonas Afetadas: set()

Planejando rota completa:
--------------------------------------------------

Analisando rota para F3:
Prioridade: 4/5
População: 267
Tempo Restante: 32.0 horas
Distância do segmento: 43 movimentos
Custo do segmento: 640.0

Distribuição de veículos para este segmento:
- car: (1, 0) -> (1, 1) (distance: 20)
- truck: (2, 1) -> (2, 2) -> (2, 3) -> (3, 3) -> (3, 4) -> (3, 5) -> (3, 6) -> (3, 7) -> (4, 7) -> (5, 7) -> (5, 8) -> (6, 8) -> (6, 9) -> (7, 9) (distance: 140)
- boat: (7, 10) -> (7, 11) (distance: 20)
- car: (8, 11) -> (9, 11) (distance: 20)
- truck: (10, 11) -> (11, 11) -> (12, 11) -> (12, 10) -> (12, 9) (distance: 50)
- boat: (12, 8) (distance: 10)
- car: (12, 7) -> (11, 7) -> (11, 8) (distance: 30)
- boat: (11, 9) -> (11, 10) -> (10, 10) (distance: 30)
- truck: (10, 9) -> (10, 8) -> (10, 7) -> (10, 6) -> (10, 5) -> (11, 5) -> (12, 5) -> (12, 4) (distance: 80)
- boat: (11, 4) -> (11, 3) -> (11, 2) (distance: 30)
- car: (12, 2) (distance: 10)
```

```
Analisando rota para F1:
Prioridade: 3/5
População: 217
Tempo Restante: 19.0 horas
Distância do segmento: 92 movimentos
Custo do segmento: 1330.0

Distribuição de veículos para este segmento:
- car: (12, 2) -> (12, 1) -> (11, 1) (distance: 30)
- boat: (11, 2) -> (11, 3) -> (11, 4) (distance: 30)
- car: (11, 5) -> (10, 5) (distance: 20)
- truck: (10, 6) -> (10, 7) -> (10, 8) -> (10, 9) (distance: 40)
- boat: (10, 10) (distance: 10)
- truck: (10, 11) -> (9, 11) (distance: 20)
- boat: (9, 10) (distance: 10)
- truck: (9, 9) -> (9, 8) -> (9, 7) -> (9, 6) -> (9, 5) (distance: 50)
- boat: (9, 4) -> (10, 4) -> (10, 3) -> (10, 2) (distance: 40)
- car: (10, 1) -> (9, 1) (distance: 20)
- boat: (8, 1) -> (7, 1) -> (7, 2) -> (6, 2) -> (6, 1) -> (5, 1) -> (5, 2) -> (5, 3) -> (5, 4) -> (5, 5) (distance: 100)
- car: (5, 6) -> (5, 7) -> (5, 8) -> (6, 8) -> (6, 9) -> (7, 9) (distance: 60)
- boat: (7, 10) -> (7, 11) -> (7, 12) (distance: 30)
- car: (7, 13) -> (6, 13) -> (6, 12) (distance: 30)
- boat: (6, 11) (distance: 10)
- car: (5, 11) -> (5, 12) -> (5, 13) -> (4, 13) -> (4, 12) (distance: 50)
- boat: (4, 11) -> (4, 10) -> (4, 9) -> (3, 9) -> (3, 10) -> (3, 11) -> (3, 12) (distance: 70)
- car: (3, 13) -> (2, 13) -> (2, 12) -> (2, 11) -> (2, 10) (distance: 50)
- truck: (2, 9) -> (2, 8) -> (2, 7) -> (3, 7) -> (4, 7) -> (4, 6) -> (4, 5) (distance: 70)
- boat: (4, 4) -> (4, 3) (distance: 20)
- truck: (4, 2) -> (4, 1) -> (3, 1) -> (3, 2) -> (3, 3) -> (3, 4) -> (3, 5) -> (3, 6) -> (2, 6) -> (1, 6) -> (1, 7) -> (1, 8) -> (1, 9) -> (1, 10) -> (1, 11) -> (1, 12)
```

```
Analisando rota para F2:
Prioridade: 2/5
População: 369
Tempo Restante: 20.0 horas
Distância do segmento: 15 movimentos
Custo do segmento: 220.0

Distribuição de veículos para este segmento:
- car: (1, 13) -> (2, 13) -> (3, 13) -> (4, 13) -> (5, 13) -> (6, 13) -> (7, 13) (distance: 70)
- boat: (7, 12) -> (7, 11) (distance: 20)
- car: (8, 11) -> (9, 11) (distance: 20)
- truck: (10, 11) -> (11, 11) -> (12, 11) -> (12, 12) -> (12, 13) (distance: 50)
```

```
============================
Rota Completa
============================
- car: (1, 0) -> (1, 1) (distance: 20)
- truck: (2, 1) -> (2, 2) -> (2, 3) -> (3, 3) -> (3, 4) -> (3, 5) -> (3, 6) -> (3, 7) -> (4, 7) -> (5, 7) -> (5, 8) -> (6, 8) -> (6, 9) -> (7, 9) (distance: 140)
- boat: (7, 10) -> (7, 11) (distance: 20)
- car: (8, 11) -> (9, 11) (distance: 20)
- truck: (10, 11) -> (11, 11) -> (12, 11) -> (12, 10) -> (12, 9) (distance: 50)
- boat: (12, 8) (distance: 10)
- car: (12, 7) -> (11, 7) -> (11, 8) (distance: 30)
- boat: (11, 9) -> (11, 10) -> (10, 10) (distance: 30)
- truck: (10, 9) -> (10, 8) -> (10, 7) -> (10, 6) -> (10, 5) -> (11, 5) -> (12, 5) -> (12, 4) (distance: 80)
- boat: (11, 4) -> (11, 3) -> (11, 2) (distance: 30)
- car: (12, 2) (distance: 10)
- car: (12, 2) -> (12, 1) -> (11, 1) (distance: 30)
- boat: (11, 2) -> (11, 3) -> (11, 4) (distance: 30)
- car: (11, 5) -> (10, 5) (distance: 20)
- truck: (10, 6) -> (10, 7) -> (10, 8) -> (10, 9) (distance: 40)
- boat: (10, 10) (distance: 10)
- truck: (10, 11) -> (9, 11) (distance: 20)
- boat: (9, 10) (distance: 10)
- truck: (9, 9) -> (9, 8) -> (9, 7) -> (9, 6) -> (9, 5) (distance: 50)
- boat: (9, 4) -> (10, 4) -> (10, 3) -> (10, 2) (distance: 40)
- car: (10, 1) -> (9, 1) (distance: 20)
- boat: (8, 1) -> (7, 1) -> (7, 2) -> (6, 2) -> (6, 1) -> (5, 1) -> (5, 2) -> (5, 3) -> (5, 4) -> (5, 5) (distance: 100)
- car: (5, 6) -> (5, 7) -> (5, 8) -> (6, 8) -> (6, 9) -> (7, 9) (distance: 60)
- boat: (7, 10) -> (7, 11) -> (7, 12) (distance: 30)
- car: (7, 13) -> (6, 13) -> (6, 12) (distance: 30)
- boat: (6, 11) (distance: 10)
- car: (5, 11) -> (5, 12) -> (5, 13) -> (4, 13) -> (4, 12) (distance: 50)
- boat: (4, 11) -> (4, 10) -> (4, 9) -> (3, 9) -> (3, 10) -> (3, 11) -> (3, 12) (distance: 70)
- car: (3, 13) -> (2, 13) -> (2, 12) -> (2, 11) -> (2, 10) (distance: 50)
- truck: (2, 9) -> (2, 8) -> (2, 7) -> (3, 7) -> (4, 7) -> (4, 6) -> (4, 5) (distance: 70)
- boat: (4, 4) -> (4, 3) (distance: 20)
- truck: (4, 2) -> (4, 1) -> (3, 1) -> (3, 2) -> (3, 3) -> (3, 4) -> (3, 5) -> (3, 6) -> (2, 6) -> (1, 6) -> (1, 7) -> (1, 8) -> (1, 9) -> (1, 10) -> (1, 11) -> (1, 12)
- car: (1, 13) -> (2, 13) -> (3, 13) -> (4, 13) -> (5, 13) -> (6, 13) -> (7, 13) (distance: 70)
- boat: (7, 12) -> (7, 11) (distance: 20)
- car: (8, 11) -> (9, 11) (distance: 20)
- truck: (10, 11) -> (11, 11) -> (12, 11) -> (12, 12) -> (12, 13) (distance: 50)

Custo Total da Rota: 2190.0
============================
```

```
=== Comparacao de Desempenho dos Algoritmos ===

Metricas medias por algoritmo:
--------------------------------------------------------------------------------
Algoritmo       Tempo (s)    Memoria (MB)    Nos Explorados  Tamanho Rota
--------------------------------------------------------------------------------
DFS             0.0020       0.00            209             153
```

Figura 3 - Output da Procura DFS

O algoritmo demonstrou um tempo de execução de 0.0020 segundos, com um consumo de memória nulo, explorando 209 nós para gerar rotas com um total de 153 movimentos.

O DFS conseguiu elaborar um plano abrangente que cobriu três zonas distintas (F3, F1 e F2), cada uma com suas respetivas prioridades e características populacionais específicas. No total, o plano desenvolvido atende uma população de 853 pessoas, distribuídas entre as três zonas, com tempos restantes variando entre 19.8 e 32 horas para execução das operações.

A consideração das condições meteorológicas adversas foi outro ponto relevante na execução do algoritmo. Várias rotas foram identificadas como "Weather Affected", indicando que o DFS foi capaz de incorporar estas condições em seu processo de planejamento, adaptando as rotas

conforme necessário. Esta característica é particularmente importante em cenários de emergência, onde condições adversas podem impactar significativamente a execução das operações.

Os custos e distâncias das rotas geradas foram distribuídos de forma estruturada, com F3 apresentando 43 movimentos e custo de 640.0, F1 com 92 movimentos e custo de 1330.0, e F2 com 15 movimentos e custo de 220.0. Esta distribuição reflete a natureza da busca em profundidade do algoritmo, que tende a explorar caminhos completos antes de considerar alternativas.

Em termos de eficiência computacional, o número de nós explorados (209) indica uma busca relativamente profunda no espaço de soluções, característica típica do DFS. Este nível de exploração, combinado com o baixo tempo de processamento, sugere um bom equilíbrio entre profundidade de busca e eficiência computacional, crucial em emergências onde o tempo de resposta é crítico.

5.2. Procura em largura (BFS)

Este tipo de procura expande primeiro os nodos de menor profundidade do grafo.

As **vantagens** deste tipo de procura são:

- Solução ótima quando todas as arestas têm custo 1.

As **desvantagens** deste tipo de procura são:

- Tempo de pesquisa elevado visto que tende a percorrer muitos mais nodos do que os necessários para criar o caminho válido.
- Ocupa muito espaço em memória.

No nosso problema o tamanho do grafo irá depender do tamanho do circuito logo este tipo de procura irá tornar-se mais lento para circuitos de grandes dimensões.

Esta estratégia de procura apresenta as seguintes propriedades:

- Tempo: $O(b\ d\ )$
- Espaço: $O(b\ d\ )$

Onde b é o número máximo de sucessores de um nodo da árvore de procura e d a profundidade da melhor solução.

6. Estratégias de Procura Informada

As estratégias de procura informada implementadas neste projeto foram a Procura Gulosa (Greedy) e a Procura A*. No nosso problema existe a possibilidade de existirem vários nodos finais por isso o valor da heurística é a menor distância entre o nodo atual e um nodo final.

6.1. Heurística

A heurística da distância de Manhattan é usada em algoritmos de busca como o A* e o Greedy Search para estimar o custo de atingir um objetivo a partir de um ponto inicial. Esse cálculo é especialmente útil em ambientes representados por grades (como mapas ou tabuleiros), onde os movimentos permitidos ocorrem apenas na horizontal ou vertical.

A fórmula da distância de Manhattan é a soma das diferenças absolutas das coordenadas x e yentre o ponto inicial e o objetivo

Distância de Manhattan=|xstart−xgoal|+|ystart−ygoal|

Esse valor estimado guia o algoritmo a explorar os caminhos mais promissores, priorizando os que parecem mais curtos com base na heurística.

A principal vantagem dessa heurística está na sua simplicidade e eficiência, sendo adequada quando os movimentos são restritos a eixos cartesianos, sem deslocamentos diagonais ou obstáculos complexos.

6.2. Procura Gulosa (Greedy)

Este tipo de procura expande para o nodo que parece estar mais perto da solução utilizando para isso uma heurística.

As **vantagens** deste tipo de procura são:

- Tempo de procura reduzido se a função heurística for boa

As **desvantagens** deste tipo de procura são:

- A solução obtida pode não ser a solução ótima.
- Ocupa muito espaço em memória.

No nosso problema a eficácia da função heurística irá depender do circuito pois parte do seu cálculo tem em conta a distância até à meta e no caso de existirem paredes entre os nodos e a meta poderá levar a uma escolha errada do próximo nodo a ser explorado, e com isso a performance ficaria afetada.

Esta estratégia de procura apresenta as seguintes propriedades:

- Tempo: $O(b$ (com uma boa função heurística pode diminuir) m)
- Espaço: $O(b\ m$)

Onde b é o número máximo de sucessores de um nodo da árvore de procura e m a máxima profundidade do espaço de estados.

6.2.1. Exemplo de procura gulosa (Greedy)

```
Condições Meteorológicas Atuais: CLEAR
Zonas Afetadas: set()

Planejando rota completa:
--------------------------------------------------

Analisando rota para F2:
Prioridade: 5/5
População: 69
Tempo Restante: 30.0 horas
Distância do segmento: 30 movimentos
Custo do segmento: 490.0

Distribuição de veículos para este segmento:
- car: (1, 0) -> (2, 0) (distance: 20)
- truck: (2, 1) -> (3, 1) -> (4, 1) (distance: 30)
- boat: (5, 1) -> (6, 1) -> (7, 1) -> (8, 1) (distance: 40)
- car: (9, 1) -> (10, 1) -> (11, 1) -> (12, 1) -> (12, 2) (distance: 50)
- boat: (11, 2) -> (11, 3) -> (11, 4) (distance: 30)
- car: (12, 4) -> (12, 5) -> (11, 5) -> (10, 5) (distance: 40)
- truck: (10, 6) -> (10, 7) -> (11, 7) -> (12, 7) (distance: 40)
- boat: (12, 8) (distance: 10)
- car: (12, 9) -> (12, 10) (distance: 20)
- truck: (12, 11) -> (12, 12) -> (12, 13) (distance: 30)
```

```
Analisando rota para F1:
Prioridade: 4/5
População: 193
Tempo Restante: 25.0 horas
Distância do segmento: 15 movimentos
Custo do segmento: 260.0

Distribuição de veículos para este segmento:
- car: (12, 13) (distance: 10)
- truck: (12, 12) -> (12, 11) -> (11, 11) -> (10, 11) -> (9, 11) -> (8, 11) (distance: 60)
- boat: (7, 11) -> (6, 11) (distance: 20)
- car: (5, 11) (distance: 10)
- boat: (4, 11) -> (3, 11) (distance: 20)
- car: (2, 11) -> (1, 11) -> (1, 12) -> (1, 13) (distance: 40)

Analisando rota para F3:
Prioridade: 1/5
População: 176
Tempo Restante: 14.0 horas
Distância do segmento: 24 movimentos
Custo do segmento: 310.0

Distribuição de veículos para este segmento:
- car: (1, 13) -> (1, 12) -> (1, 11) -> (1, 10) -> (1, 9) (distance: 50)
- truck: (1, 8) -> (1, 7) -> (1, 6) -> (2, 6) -> (2, 5) -> (3, 5) -> (3, 4) -> (3, 3) -> (3, 2) -> (4, 2) (distance: 100)
- boat: (5, 2) -> (6, 2) -> (7, 2) -> (7, 1) -> (8, 1) (distance: 50)
- car: (9, 1) -> (10, 1) -> (11, 1) -> (12, 1) -> (12, 2) (distance: 50)
```

```
============================
Rota Completa
============================
- car: (1, 0) -> (2, 0) (distance: 20)
- truck: (2, 1) -> (3, 1) -> (4, 1) (distance: 30)
- boat: (5, 1) -> (6, 1) -> (7, 1) -> (8, 1) (distance: 40)
- car: (9, 1) -> (10, 1) -> (11, 1) -> (12, 1) -> (12, 2) (distance: 50)
- boat: (11, 2) -> (11, 3) -> (11, 4) (distance: 30)
- car: (12, 4) -> (12, 5) -> (11, 5) -> (10, 5) (distance: 40)
- truck: (10, 6) -> (10, 7) -> (11, 7) -> (12, 7) (distance: 40)
- boat: (12, 8) (distance: 10)
- car: (12, 9) -> (12, 10) (distance: 20)
- truck: (12, 11) -> (12, 12) -> (12, 13) (distance: 30)
- car: (12, 13) (distance: 10)
- truck: (12, 12) -> (12, 11) -> (11, 11) -> (10, 11) -> (9, 11) -> (8, 11) (distance: 60)
- boat: (7, 11) -> (6, 11) (distance: 20)
- car: (5, 11) (distance: 10)
- boat: (4, 11) -> (3, 11) (distance: 20)
- car: (2, 11) -> (1, 11) -> (1, 12) -> (1, 13) (distance: 40)
- car: (1, 13) -> (1, 12) -> (1, 11) -> (1, 10) -> (1, 9) (distance: 50)
- truck: (1, 8) -> (1, 7) -> (1, 6) -> (2, 6) -> (2, 5) -> (3, 5) -> (3, 4) -> (3, 3) -> (3, 2) -> (4, 2) (distance: 100)
- boat: (5, 2) -> (6, 2) -> (7, 2) -> (7, 1) -> (8, 1) (distance: 50)
- car: (9, 1) -> (10, 1) -> (11, 1) -> (12, 1) -> (12, 2) (distance: 50)

Custo Total da Rota: 1060.0
============================
```

```
=== Comparacao de Desempenho dos Algoritmos ===

Metricas medias por algoritmo:
--------------------------------------------------------------------------------
Algoritmo       Tempo (s)    Memoria (MB)    Nos Explorados  Tamanho Rota
--------------------------------------------------------------------------------
GREEDY          0.0012       0.00            72              72
```

Figura 4 - Output da Procura Greedy

O método Greedy demonstrou características interessantes neste cenário de evacuação que merecem destaque. Primeiro, vamos analisar os aspectos positivos e depois os potenciais pontos de atenção:

Pontos Fortes do Desempenho:

O algoritmo demonstrou eficiência computacional notável, com tempo de execução de apenas 0.0012 segundos e consumo de memória praticamente negligenciável. Isto é uma característica típica dos algoritmos gulosos, que tomam decisões localmente ótimas sem necessidade de manter estados complexos em memória.

Em termos de qualidade da solução, o algoritmo conseguiu gerar rotas que consideram diversos fatores complexos:

A priorização foi respeitada, como podemos ver pela ordem de atendimento das zonas F2 (prioridade 5/5), F1 (4/5) e F3 (1/5), demonstrando que o algoritmo considera adequadamente os níveis de urgência.

O algoritmo demonstrou capacidade de adaptação multimodal, alternando entre diferentes tipos de veículos (carros, barcos e caminhões) conforme necessário para otimizar o trajeto.

Não foi necessário ter em conta as condições meteorológicas adversas visto esta estar definido como limpo (CLEAR), caso fosse o algoritmo geraria rotas alternativas quando necessário.

Pontos de Atenção:

O método Greedy, por sua natureza de decisões locais, pode não ter encontrado a solução globalmente ótima. Por exemplo:

A distribuição de veículos entre os segmentos poderia potencialmente ser otimizada para reduzir ainda mais as distâncias totais.

Algumas rotas, como a de F2 com 30 movimentos, são consideravelmente mais longas que outras, o que pode indicar que uma visão mais global do problema poderia levar a uma distribuição mais equilibrada.

No contexto de emergências, onde o tempo de resposta é crítico, a capacidade do algoritmo de gerar rapidamente uma solução viável pode ser mais importante que encontrar a solução ótima, tornando o trade-off entre qualidade da solução e tempo de computação aceitável para este cenário específico.

6.3. Procura A*

Este tipo de procura evita expandir caminhos que são dispendiosos, combinando para isso o algoritmo de procura gulosa com o algoritmo de procura uniforme.

Utiliza então a seguinte função para a escolha do nodo a ser explorado de seguida: $f(n) = g(n) + h(n)$

Onde:

- g(n) = custo acumulado
- h(n) = custo estimado para chegar ao destino (heurística)

As **vantagens** deste tipo de procura são:

- Obtém a solução ótima se a sua heurística for admissível.

As **desvantagens** deste tipo de procura são:

- Ocupa muito espaço em memória.

No nosso problema a eficácia da heurística irá depender do circuito como já foi explicado na análise da Procura Gulosa. No entanto este algoritmo de procura para a escolha do próximo nodo, analisa a heurística e o custo acumulado do seu percurso até ao momento. Isso permite ao algoritmo encontrar a solução ótima para o problema.

Esta estratégia de procura apresenta as seguintes propriedades:

- Tempo: $O(b$ (com uma boa função heurística pode diminuir m) significativamente)
- Espaço: $O(b\ m)$

  Onde b é o número máximo de sucessores de um nodo da árvore de procura e m a máxima profundidade do espaço de estados.

7. Conclusão

Concluindo, neste projeto desenvolvemos um programa que permite a utilização de algoritmos de procura para explorar caminhos possíveis e atingir uma solução, tendo sido descritas todas as vantagens e desvantagens de cada estratégia utilizada. Finalmente, gostaríamos de referir que, quanto à heurística escolhida para os algoritmos de procura informada, tivemos alguns problemas de início a entender como incluir a velocidade do carro no seu cálculo e como isso afetaria o nosso problema.