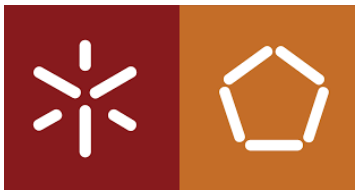

Relatório TP | Inteligência Artificial
Grupo 15 | 2024/2025

João Lobo (A104356)

Mariana Rocha (A90817)

Mário Rodrigues (A100109)

Rita Camacho (A104439)

Índice

Avaliação pelos pares

A104356 João DELTA = 0

A90817 Mariana DELTA = 0

A100109 Mário DELTA = 0

A104439 Rita DELTA = 0

1. Introdução

Em situações de catástrofes naturais, a logística eficiente de distribuição de alimentos e recursos é uma necessidade muito importante, pois temos de garantir que os recursos disponíveis são entregues às áreas mais necessitadas de forma rápida e organizada. Este projeto tem como objetivo desenvolver e implementar algoritmos de procura capazes de otimizar a distribuição de recursos, maximizando o número de pontos assistidos.

Para abordar esse desafio, o problema foi formulado como um problema de procura, no qual as zonas de entrega e os caminhos possíveis são representados num grafo. Este modelo permite simular a realidade das operações logísticas e aplicar diversas estratégias de procura – tanto informadas, como não informadas – de forma a encontrar os percursos mais adequados. Os algoritmos desenvolvidos consideram múltiplos fatores críticos, como prioridades de zonas, condições climatéricas e bloqueio de rotas, de modo a avaliar a eficiência dos algoritmos em cenários reais.

2. Descrição do problema

A distribuição de alimentos e recursos para áreas afetadas por um catástrofe natural é o problema principal a ser desenvolvido neste projeto. O desafio é também garantir que os recursos disponíveis sejam utilizados de forma eficiente, maximizando o número de postos sem necessidade de suplementos.

As zonas afetadas possuem diferentes níveis de prioridade com base na gravidade e as rotas podem ser afetadas com fatores externos, como condições meteorológicas ou rotas bloqueadas. Além disso, os veículos responsáveis pela entrega têm limitações de capacidade e combustível, o que exige uma gestão cuidadosa das rotas e do consumo de recursos.

O objetivo principal é desenvolver algoritmos de procura capazes de encontrar as melhores rotas para distribuir os recursos às zonas críticas, com isso temos que otimizar o uso de veículos e considerar as zonas de prioridade de cada zona, além de lidar com as dinâmicas imprevisíveis do ambiente e das condições externas.

3. Formulação do Problema

3.0.1. Tipo:

Este problema pode ser classificado como um problema de otimização de percursos com múltiplas restrições e também un problema de procura em grafos, especificamente um problema de caminho mínimo com restrições. Pode ser classificado como:

1. Quanto ao ambiente: Dinâmico, Observável, Determinístico, Multi-agente e Sequencial;
2. Quanto à natureza da solução: Caminho (a solução é uma sequência de ações que nos levam ao objetivo), Otimização (procura-se a melhor solução possível), com restrições temporais.

3.0.2. Representação do Estado:

O estado do sistema pode ser representado por uma tupla: (T, V, PD, PR, C, M, G), onde:

- T: Tempo - representa o tempo passado em minutos.
- V: Veículos - lista de tuplas que contém as seguintes informações:
  - Identificador do veículo;
  - Tipo do veículo (Drones, Camiões, Barcos e Helicópteros);
  - Meio de transporte (terrestre, aéreo ou marítimo);
  - Posição atual (x,y);
  - Nível de combustível no momento e máximo (em litros);
  - Peso e volume da carga no momento (em kg e m^3);
  - Capacidade e volume máximo da carga (em kg e m^3);
  - Estado (disponível ou em rota);
  - Velocidade média de deslocamento (km/h).
- PD: Ponto de Distribuição - com as suas coordenadas (x,y).

- PR: Pontos de Recolha - lista de tuplas, cada um contendo:
  - Identificador do ponto;
  - Posição (x,y);
  - Necessidades atuais (mantimentos necessários de cada tipo e as suas quantidades);
  - Zona de prioridade: 0 se não é uma zona crítica, 1 caso contrário;
  - Tipos de acesso disponíveis;
  - Estado (não atendido, parcialmente atendido, atendido).
- C: Condições Climatéricas:
  - Chuva;
  - Neve;
  - Tempestade;
  - Sol.
- M: Inventário de Mantimentos - tupla com:
  - Alimentos;
  - Água potável;
  - Medicamentos;
  - Cada mantimento tem associada uma quantidade expressa em unidades de peso e volume (em kg, m^3).
- G: Geografia e Acessibilidade

  Grafo, que representa o momento, com as seguintes características:

  - Tipos de terreno (água, terra, aéreo)
  - Caminhos disponíveis:
    - Estradas (para transporte terrestre)
    - Rotas aéreas
    - Vias navegáveis
  - Obstáculos e restrições:
    - Zonas intransitáveis
    - Áreas de exclusão aérea

3.0.3. Estado Inicial:

No estado inicial, ainda nenhum posto de recolha foi atendido, de modo que temos:

1. T = 0
2. Posto de Distribuição:
   1. Localização: Coordenadas (x,y) do posto central de distribuição.
3. Inventário de Mantimentos Inicial:
   1. Tipos de mantimentos: alimentos, água potável e medicamentos;
   2. Quantidades de cada tipo de mantimento inicialmente.
4. Transportes Disponíveis:

   Uma lista detalhada dos veículos disponíveis inicialmente com as seguintes características:

   1. A capacidade e o volume de cada veículo encontram-se a 0;
   2. Nível de combustível inicial de cada veículo está no seu valor máximo;

5. Condições Climatéricas:
   1. Todos os nodos apresentam estar num dia de Sol.
6. Pontos de Recolha:
   1. Localização: Coordenadas (x,y);
   2. Necessidades iniciais específicas em termos de mantimentos;
   3. Zona de Prioridade: valor especificado inicialmente.
7. Mapa de Acessibilidade:
   1. Grafo, que representa inicialmente o mapa, com as seguintes características:
      - Tipos de terreno (água, terra, aéreo);
      - Caminhos disponíveis iniciais;
      - Obstáculos e restrições iniciais.

3.0.4. Teste Objetivo:

O teste objetivo para este problema é assegurar que todos os pontos de recolha foram atendidos e estão no estado "não crítico" com o nível máximo de mantimentos pedidos possíveis, respeitando todas as restrições temporais, logísticas, e ambientais do problema.

3.0.5. Operadores:

3.0.5.1. Deslocar:

Descrição:

Move um veículo da sua posição atual para um destino viável, respeitando as restrições de acessibilidade, combustível, carga e condições climatéricas.

Pré-condições: O veículo está no estado "disponível". O destino é acessível ao veículo (compatibilidade do terreno e do meio de transporte). O nível de combustível do veículo é suficiente para cobrir a distância até o destino, considerando possíveis ajustes devido ao clima (chuva, neve, etc.). O destino não possui obstáculos ou restrições que impeçam a movimentação.

Efeitos: Atualiza a posição do veículo para o destino. Reduz o combustível do veículo de acordo com a distância percorrida, o peso da carga transportada e as condições climatéricas. Atualiza o tempo total de acordo com a distância e a velocidade ajustada do veículo. Pode realizar entregas de mantimentos no destino, caso o destino seja um ponto de recolha e o veículo esteja carregado.

3.0.5.2. Carregar Mantimentos:

Descrição:

Carrega mantimentos num veículo no ponto de distribuição ou num outro ponto de recolha com mantimentos disponíveis.

Pré-condições:

O veículo está no ponto onde os mantimentos estão armazenados.

O veículo possui capacidade livre para armazenar os mantimentos necessários (em peso e volume).

Os mantimentos necessários estão disponíveis no local de carregamento.

Efeitos:

Atualiza a carga do veículo, aumentando o peso e o volume transportados.

Reduz a quantidade de mantimentos disponíveis no local de carregamento.

3.0.5.3. Entregar Mantimentos:

Descrição:

Entrega mantimentos num ponto de recolha, conforme as suas necessidades e atualiza o estado do ponto (não atendido, parcialmente atendido, atendido).

Pré-condições:

O veículo está no ponto de recolha. O veículo possui mantimentos compatíveis com as necessidades do ponto.

Efeitos:

Reduz a carga do veículo de acordo com os mantimentos entregues.

Atualiza o estado do ponto de recolha com base nos mantimentos entregues.

Atualiza o tempo total de acordo com o tempo necessário para a entrega.

3.0.6. Custo da solução:

O custo da solução num cenário de contenção é variável e depende de fatores como os recursos entregues, o combustível, a distância percorrida, o tempo envolvido, e as condições externas. Os recursos são os mantimentos entregues em cada ponto, impactam o custo consoante o volume e peso. O consumo de combustível varia com a carga transportada, eficiência dos veículos, velocidade média e distância percorrida. A distância percorrida aumenta os gastos proporcionalmente, considerando rotas mais longa, alterações no percurso, e ainda desvios causados por estradas cortadas ou inacessíveis. O tempo reflete-se na duração das entregas. Além disso, fatores como as condições climatéricas adversas (chuva, neve, ou vento forte) podem agravar a complexidade do transporte, exigindo ajustes nas rotas e maior consumo de recursos, influenciando significativamente o custo final da solução.

$$C = \alpha \cdot R + \beta \cdot D + \gamma \cdot F + \delta \cdot T + \epsilon \cdot P$$

- C: Custo total a ser minimizado.
- R: Custo dos recursos entregues (mantimentos).
- D: Distância total percorrida (em km).
- F: Consumo de combustível total (em litros ou equivalente energético).
- T: Tempo total das entregas (em horas).
- P: Penalidades associadas a condições adversas (como clima e desvios de rota).

- α, β, γ, δ, ϵ: Pesos que representam a importância relativa de cada componente no problema.

4. Estrutura do Projeto

O projeto é estruturado com várias classes, cada uma desempenha um papel específico e importante na simulação.

4.1. Estrutura do Grafo

A classe `Graph` representa o grafo, que contêm diversos nodos (`nodes`), este permite adicionar nodos e arestas, sendo estas a ligação entre dois nodos adjacentes. A classe `Node` representa um nodo do grafo, tem associada uma posição `Position`, uma lista de vizinhos e os terrenos acessíveis até esse ponto. Também verificámos se o nodo é acessível tendo em conta as condições climatéricas presentes no nodo. A classe `Position` define a posição (x,y) no grafo.

4.2. Pontos de Relevância

A classe `EndPoint` representa um ponto de destino que necessita de mantimentos, este mantém os mantimentos necessários e pode atualizá-los conforme recebe as entregas do ponto de distribuição. A classe `StartPoint` indica o ponto de distribuição e partida, armazenando os mantimentos disponíveis para a distribuição.

4.3. Gestão de mantimentos

O `SupplyType` enumera os tipos de mantimentos disponíveis: Água, Comida e Medicação. O `Supply` representa um mantimento e contêm informação como a sua quantidade e tipo.

4.4. Veículos

O `VehicleType` define o tipo de veículos (terrestres, aéreos ou marítimos) com atributos como capacidade de combustível, peso, volume e velocidade média. A sua velocidade é ajustada com base nas condições climatéricas e verifica a acessibilidade ao terreno.

O `Vehicle` representa um veículo com identificador único, posição atual, tipo de veículo, estado (livre ou ocupado) e capacidade atual de combustível, peso e volume.

4.5. Clima e Terreno

O `WeatherCondition` enumera as diferentes condições metereológicas possíveis: Sol, Chuva, Neve e Tempestade.

A classe `Weather` gere as condições climatéricas nas diferentes posições do grafo. Também determina se uma posição está bloqueada por estar Tempestade.

Apesar dos terrenos não terem uma classe específica, o seu acesso é controlado por nodo e pelo tipo de veículo, dependendo da compatibilidade do veículo com o terreno.

Esta arquitetura modular permite fácil expansão, como adição de novos tipos de terreno, veículos ou regras de acessibilidade, de forma a garantir flexibilidade e eficiência no desenvolvimento do projeto.

5. Algoritmos

5.1. Procura Não Informada

5.1.1. Procura Primeiro em Largura

O algoritmo implementado para a entregar os mantimentos no melhor percurso possível, utiliza a procura não informada BFS (*Breadth-First Search*), sendo adaptado para corresponder às complexidades do problema. O BFS é particularmente útil para explorar todos os nodos de um grafo de maneira uniforme, garantindo que o menor caminho seja encontrado em grafos sem custo.

Complexidade Temporal	$O(b^d)$
Complexidade Espacial	$O(b^d)$

Inicialmente, o algoritmo verifica as necessidades de mantimentos no ponto final e compara com os mantimentos disponíveis no ponto inicial, de forma a determinar o que pode ser enviado. Este processo considera tanto a quantidade necessária quanto a disponível, garantindo que nenhum recurso seja desperdiçado.

Durante a execução do **BFS**, o algoritmo armazena os nodos visitados para evitar revisitas e mantém uma fila que pesquisa a posição atual, o caminho percorrido e a distância total acumulada. Uma das características importantes desta implementação é a consideração de rotas bloqueadas, terrenos intransitáveis e efeitos das condições climatéricas. Por exemplo, o algoritmo ajusta a distância percorrida com base no clima (como neve ou chuva), refletindo os cenários reais em diferentes condições meteorológicas. Este detalhe torna o algoritmo mais robusto e realista.

Uma das etapas mais críticas acontece ao encontrar o destino. O algoritmo identifica os veículos disponíveis no ponto inicial, verificando a sua capacidade de combustível, estado (disponível ou ocupado) e compatibilidade com o terreno. Em seguida, utiliza uma função auxiliar para dividir os mantimentos pelos veículos, garantindo que cada veículo transporta apenas o que lhe é possível. Além disso, o algoritmo atualiza o estado global do sistema, ajustando a posição dos veículos, o combustível restante e as quantidades disponíveis de mantimentos.

Por fim, a implementação também considera o tempo necessário para cada veículo completar a entrega, ajustando a sua velocidade com base no terreno e nas condições climatéricas. Isso garante que a solução não entregua apenas os mantimentos corretamente, mas também fornece uma estimativa precisa do tempo necessário. Esta atenção aos detalhes demonstra como o algoritmo combina eficiência com um modelo realista, sendo uma solução eficaz para problemas logísticos complexos.

5.1.2. Procura Primeiro em Profundidade

Depth First Search, ou Procura Primeiro em Profundidade, é um algoritmo de procura não informada. O seu funcionamento baseia-se na realização de sucessivas execuções de pesquisa em profundidade, isto é, são visitados e conhecidos prioritariamente os nodos mais profundos. Este algoritmo é bastante eficiente a nível de memória, no entanto é

de evitar a sua utilização em árvores de profundidade infinita, visto que poderá ficar "preso" num ramo errado. Assim, não é uma solução **completa** nem **ótima**.

Complexidade Temporal	$O(b^m)$
Complexidade Espacial	$O(bm)$

Inicialmente, o algoritmo avalia as necessidades dos mantimentos no ponto destino e compara-as com os mantimentos disponíveis no ponto de partida. Apenas os mantimentos que podem ser enviados são selecionados, evitando desperdícios e garantindo a otimização dos recursos.

A exploração DFS começa no ponto inicial, armazena os nodos visitados para evitar revisitas e utiliza uma *queue* que contém a posição atual, o caminho percorrido e a distância acumulada. Durante o percurso, o algoritmo considera restrições como rotas bloqueadas, compatibilidade de terreno e impacto das condições climatéricas. Por exemplo, condições como neve ou chuva aumentam a distância percorrida proporcionalmente, refletindo cenários realistas.

Ao alcançar o destino, o algoritmo identifica os veículos disponíveis no ponto inicial e verifica se têm combustível suficiente, capacidade compatível e estão disponíveis. Os mantimentos são então divididos entre os veículos, assegurando que cada um transporte apenas o que é possível com base nas suas capacidades. Esta divisão é feita por meio de uma função auxiliar. Além disso, o tempo total da entrega é calculado ajustando a velocidade dos veículos com base nas condições meteorológicas.

5.1.3. Procura iterativa de aprofundamento progressivo

Iterative deepening depth-first search, ou procura iterativa de aprofundamento progressivo, é um algoritmo de procura não informada. Este funciona realizando várias execuções consecutivas de uma procura em profundidade, cada vez com um limite crescente na profundidade explorada. A ideia principal é que, em cada iteração, a busca em profundidade explore apenas até um certo limite de profundidade d, aumentando esse limite gradualmente até encontrar a solução ou até que todos os nodos sejam explorados, permitindo que o algoritmo herde a eficiência de memória da busca em profundidade enquanto mantém a característica de completude da busca em largura.

Complexidade Temporal	$O(b^d)$
Complexidade Espacial	$O(bd)$

O processo começa com uma análise inicial para verificar as necessidades dos mantimentos no destino e comparar com os recursos disponíveis na origem. Apenas os recursos que podem ser enviados, respeitam as capacidades disponíveis e evitam desperdícios são selecionados.

Em seguida, o algoritmo executa uma procura limitada em profundidade para cada limite definido. Durante essa procura, ele explora os nodos do grafo de forma sistemática, armazena os caminhos percorridos e a distância acumulada. A cada passo, verifica a presença de rotas bloqueadas, a compatibilidade do terreno com os veículos disponíveis e

o impacto das condições climatéricas. Quando encontra neve ou chuva, ajusta a distância percorrida para refletir cenários mais realistas.

Ao alcançar o destino, o algoritmo seleciona os veículos disponíveis no ponto inicial que possuem combustível suficiente e são compatíveis com o terreno. Em seguida, utiliza uma função auxiliar para dividir os mantimentos entre os veículos, o que garante eficiência e equidade. Além disso, calcula o tempo necessário para a entrega de cada veículo, considerando a velocidade ajustada com base nas condições climatéricas e a distância entre os pontos do caminho.

Por fim, o estado global do sistema é atualizado. A posição dos veículos é ajustada para o destino, o combustível remanescente é recalculado e os mantimentos entregues são retirados da origem e adicionados ao destino. Caso nenhum caminho seja encontrado no limite de profundidade atual, o algoritmo aumenta o limite e repete o processo, explora progressivamente camadas mais profundas do grafo. Essa estratégia evita que o algoritmo fique preso em ciclos ou ramos muito profundos e mantém o consumo de memória baixo.

Essa abordagem garante a completude do algoritmo, ou seja, ele sempre encontra uma solução se ela existir, enquanto mantém a eficiência de memória típica da busca em profundidade. Além disso, a integração de fatores como rotas bloqueadas, terrenos intransitáveis e condições climatéricas torna a solução altamente adaptável e prática para cenários logísticos reais.

5.1.4. Custo Uniforme

O algoritmo de Custo Uniforme é um algoritmo que explora os caminhos em ordem crescente do custo acumulado, garantindo que o primeiro caminho encontrado para o objetivo seja o mais barato em termos de custo total. Este algoritmo é ideal para situações em que os custos das arestas variam e o objetivo é minimizar o custo total do percurso.

O Custo Uniforme utiliza uma fila de prioridade para priorizar os nodos com base no custo acumulado (g(n)). A função de custo é simplesmente (f(n) = g(n)), onde (g(n)) representa o custo real acumulado para alcançar o nodo (n).

Este algoritmo é ótimo, pois encontra o caminho de custo mínimo, desde que os custos das arestas sejam positivos. Além disso, é completo, pois assegura que, se existir um caminho até ao objetivo, ele será encontrado. No contexto do problema apresentado, o Custo Uniforme é aplicado considerando restrições como o tipo de terreno, condições meteorológicas e rotas bloqueadas. Ele também é responsável por distribuir os mantimentos necessários utilizando os veículos disponíveis, assegurando o menor custo total.

A complexidade temporal e espacial do Custo Uniforme é onde, $\mathbf{b}$ é o fator de ramificação, ou seja, o número médio de filhos de cada nó no grafo. O termo $\boldsymbol{C^*}$ representa o custo da solução ótima, ou seja, o custo mínimo para alcançar o objetivo, enquanto $\boldsymbol{\varepsilon}$ é o custo mínimo das arestas entre os nós. A fórmula mostra que, conforme o custo da solução ótima aumenta, a profundidade da busca necessária também cresce, resultando em um aumento no tempo e no espaço necessários para a execução do algoritmo. Isso

implica que, para grafos com soluções de alto custo, o Custo Uniforme pode se tornar ineficiente em termos de recursos computacionais.

Complexidade Temporal	$O(b^{1+(\frac{C^*}{\varepsilon})}$
Complexidade Espacial	$O(b^{1+(\frac{C^*}{\varepsilon})}$

No contexto do problema de entrega de mantimentos, o custo de cada aresta é determinado pela distância entre os nodos, ajustada de acordo com as condições climatéricas. Condições adversas aumentam o custo da aresta correspondente, tornando esses caminhos menos preferidos na busca. O Custo Uniforme garante que o caminho encontrado será aquele com o menor custo total, considerando penalizações devidas a clima ou terreno.

Por fim, o Custo Uniforme é uma escolha eficaz para o problema de entrega de mantimentos, pois garante que os caminhos encontrados sejam os de menor custo total, considerando restrições como o terreno, o clima e as rotas bloqueadas. No contexto do problema, isso significa que as entregas são realizadas de forma eficiente, minimizando o consumo de combustível dos veículos e o tempo total de transporte, mesmo em condições adversas.

5.2. Procura Informada

5.2.1. Heurísticas

Nos algoritmos de procura informada, as heurísticas desempenham um papel muito importante, porque fornecem estimativas para direcionar a procura de forma eficiente. Neste projeto, foram desenvolvidas várias heurísticas para lidar com diferentes aspetos do problema de entrega de mantimentos, onde tentámos equilibrar fatores como a distância, o tempo, as restrições e a probabilidade de sucesso.

A heurística baseada na **distância de Manhattan** é a mais básica, calculando a soma das diferenças absolutas entre as coordenadas dos pontos. Esta aproximação é útil em ambientes de cadeia, pois reflete o caminho mínimo esperado sem considerar obstáculos ou restrições específicas. No entanto, ignora variáveis importantes, como condições climatéricas ou rotas bloqueadas.

Uma abordagem mais avançada é a **heurística de estimativa de tempo**, que leva em conta a velocidade média dos veículos disponíveis. Esta calcula o tempo mínimo estimado para percorrer a distância entre dois pontos, tendo em conta o veículo mais rápido e as suas condições de operação. Isso permite que o algoritmo dê prioridade a soluções que minimizem o tempo total da entrega, sendo especialmente útil quando o tempo é um fator crítico na decisão.

A **heurística de rotas bloqueadas** introduz penalizações para caminhos que envolvem caminhos indisponíveis. Ao somar um valor arbitrário à distância *Manhattan*, ela desincentiva o uso de rotas bloqueadas ou menos acessíveis. Esta abordagem ajuda o

algoritmo a evitar áreas problemáticas, considerando fatores como acessibilidade do terreno e condições meteorológicas, que podem complicar o trajeto.

A **heurística de prioridade dinâmica de mantimentos** foca-se nas necessidades críticas do ponto de destino. Esta avalia os mantimentos necessários e penaliza casos em que a oferta inicial não é suficiente para atender ao esperado. Isso incentiva o uso de rotas e veículos que possam completar adequadamente as necessidades, garantindo que os recursos mais urgentes sejam entregues primeiro. Essa abordagem é particularmente útil em cenários com limitações de recursos.

A **heurística de probabilidade de sucesso na entrega** é talvez a mais sofisticada, avaliando a viabilidade logística de uma operação. Esta tem em conta a capacidade dos veículos disponíveis em termos de peso, volume e combustível. A heurística calcula a probabilidade de sucesso com base na capacidade total dos veículos em relação às exigências do destino e penaliza soluções com menor probabilidade de sucesso. Esta abordagem dá prioridade a rotas e configurações que maximizem a eficiência e a viabilidade da entrega, equilibrando tempo e capacidade.

Finalmente, a **heurística combinada** combina todas as heurísticas anteriores.

5.2.2. A*

O algoritmo A* combina a exploração de caminhos do BFS com a utilização de heurísticas, tornando-se uma abordagem poderosa para problemas de procura informada. A principal diferença do A* é o uso de uma função de custo $f(n) = g(n) + h(n)$, onde $g(n)$ é o custo acumulado para chegar ao nodo atual e $h(n)$ é a estimativa heurística do custo restante para alcançar o objetivo.

Complexidade Temporal	Nº de nodos com $g(n) + h(n) \leq C^*$
Complexidade Espacial	Nº de nodos com $g(n) + h(n) \leq C^*$

No contexto do seu problema, o A* começa por separar os mantimentos necessários e disponíveis, determinando quais é que podem ser enviados para corresponder às necessidades do destino. Após essa preparação, o algoritmo explora os nodos vizinhos do ponto de partida, calcula o custo acumulativo $g(n)$ baseia-se na distância percorrida (ajustada por condições climatéricas, como chuva ou neve) e soma a isso o custo estimado $h(n)$, calculado através da heurística fornecida. O uso de heurísticas, como distância de *Manhattan* ou tempo estimado, permite dar prioridade a possivelmente melhores caminhos.

O A* também leva em consideração fatores práticos da logística de entrega, como veículos disponíveis no ponto inicial. Os veículos são filtrados pela sua capacidade de combustível, tipo de terreno que podem aceder e estado operacional. Caso seja encontrado um caminho viável, os mantimentos são distribuídos pelos veículos disponíveis usando a função `split_supplies_per_vehicle`. A implementação também calcula o tempo total de entrega considerando a velocidade ajustada de cada veículo em função do clima, além de gerir o consumo de combustível e o estado dos veículos.

Se um caminho for encontrado, o algoritmo fornece o percurso, a distância total, o tempo estimado e uma distribuição dos mantimentos por veículo.

5.2.3. Greedy

O algoritmo Greedy é uma abordagem que utiliza heurísticas para orientar a exploração de caminhos em busca de uma solução eficiente para problemas de procura informada. Ao contrário do A*, que combina custos acumulativos e estimados, o Greedy concentra-se exclusivamente na heurística h(n), priorizando nodos com menor custo estimado para o objetivo. Essa estratégia torna-o mais rápido em muitos casos, mas menos robusto, pois não garante a solução mais curta.

Complexidade Temporal	Nº de nodos com $h(n) \leq C^*$
Complexidade Espacial	Nº de nodos com $h(n) \leq C^*$

No início da execução, o algoritmo verifica os mantimentos necessários no ponto de destino e os mantimentos disponíveis no ponto de partida. Ele separa os mantimentos disponíveis de acordo com o tipo e a quantidade necessária para que apenas os mantimentos que podem ser enviados sejam preparados para transporte. A quantidade de mantimentos consumidos é registada para garantir que o ponto de destino receba exatamente a quantidade necessária.

Depois, o algoritmo usa uma fila de prioridade que irá guardar os nodos a serem explorados, com base em uma estimativa heurística de custo até o objetivo. A fila começa com um ponto de partida, e o algoritmo explora os nodos vizinhos, escolhendo sempre o nodo que possui a menor estimativa heurística. Cada vez que um novo nodo é explorado, o algoritmo calcula a distância até esse nodo, ajustada pelas condições meteorológicas, como neve ou chuva, que afetam o custo da rota. Quando uma rota é considerada viável, o algoritmo ajusta o custo da distância com base no clima, aplicando um fator de aumento, como 1.25 para neve ou 1.1 para chuva.

Quando o algoritmo chega ao ponto de destino, ele verifica quais veículos estão disponíveis no ponto de partida e que podem acessar o terreno necessário para a entrega. Os veículos são filtrados de acordo com o combustível disponível, estado operacional e a sua capacidade de acesso ao terreno. Em seguida, é usada uma função para distribuir os mantimentos pelos veículos de forma otimizada.

O tempo total de entrega é calculado considerando a velocidade ajustada dos veículos de acordo com as condições meteorológicas e a distância percorrida entre os nodos do caminho. A quantidade de combustível dos veículos também é atualizada durante o percurso.

Se o caminho for encontrado, o algoritmo retorna o percurso, a distância total percorrida, o tempo total estimado para a entrega, e uma distribuição dos mantimentos entregues por cada veículo.

6. Interface

A classe `Viewer` é a *interface* gráfica do simulador, foi desenvolvida com o auxílio da biblioteca `Tkinter`, e facilita a interação do utilizador com o sistema. Através da janela exibida, visualizámos o grafo e menus dinâmicos para configurar algoritmos, heurísticas, terrenos e outros parâmetros. O menu superior permite selecionar algoritmos e heurísticas disponíveis, definir o tipo de terreno e iniciar ou reiniciar a simulação. Cada ação no menu atualiza a *interface*, graças a *callbacks* que comunicam as escolhas do utilizador ao sistema principal.

O método `display_graph` desenha o grafo, com os seus nodos, arestas e elementos como ps veículos e os pontos de entrega representados graficamente. As arestas bloqueadas encontram-se a vermelho, enquanto as restantes aparecem a azul. *Tooltips* são exibidos ao passar o cursor sobre os nodos, arestas ou veículos, fornecendo informação detalhada como capacidade dos veículos, mantimentos necessários nos pontos de entrega.

Além disso, as condições meteorológicas de cada nodo são representadas visualmente: clima ensolarado (nodos com cor dourada), clima chuvoso (nodos com cor azul escuro), neve (nodos com azul claro) e tempestade (nodos com cor violeta). O utilizador pode alterar as condições meteorológicas de forma interativa, introduzindo o identificador do nodo e a condição desejada.

O controlo da simulação é intuitivo. O botão `Start Simulation` inicia o processo, enquanto o menu de bloqueio de rotas (`Block Route`) permite que o utilizador insira manualmente rotas no formato "node1,node2" para bloqueá-las. O método `draw_path` desenha o percurso percorrido em tempo real, destacando visualmente as conexões visitadas. Já o método `show_info_box` exibe um resumo da simulação, incluindo a distância total percorrida e o tempo gasto, com uma janela temporária para a sua visualização.

A *interface* também gere os elementos gráficos de forma eficiente. Imagens para veículos, pontos de início e fim, e outros *ícones* são redimensionadas dinamicamente. A modularidade permite adicionar novos algoritmos, heurísticas ou tipos de terreno facilmente, apenas atualizando os dicionários correspondentes e ajustando os *callbacks*.

7. Resultados obtidos

Os resultados obtidos pelos algoritmos de procura serão apresentados de seguida na tabela. Esta mostra para diferentes cenários, a distância total percorrida e o tempo necessário para percorrer esse caminho. Estes resultados permitem analisar a eficiência e o desempenho dos algoritmos em diferentes cenários. Estes resultados foram todos efetuados para o mesmo ponto de destino, sendo este o ponto 1.

Algoritmo	Cenário 1	Cenário 2	Cenário 3	Cenário 4
BFS	D: 3.13 km | T : 0.03 h	D: 2.82 km | T: 0.02 h	D: 3.12 km | T: 0.03 h	D: 3.48 km | T: 0.03 h
DFS	D: 7.60 km | T: 0.07h	: 6.74 km | T: 0.06 h	D: 7.45 km | T: 0.07 h	D: 5.63 km | T: 0.05 h
IDS	D: 4.99km | T: 0.05 h	D: 3.74 km | T: 0.03 h	D: 3.80 km | T: 0.03 h	D: 3.74 km | T: 0.03 h
Uniform-Cost	D: 2.70 km | T: 0.02h	D: 2.82 km | T: 0.02 h	D: 3.05 km | T: 0.03 h	D: 3.10 km | T: 0.03 h
A* 1	D: 2.70 km | T: 0.02 h	D: 2.82 km | T: 0.02 h	D: 3.05 km | T: 0.03 h	D: 3.10 km | T: 0.03 h
A* 2	D: 2.70 km | T: 0.02 h	D: 2.82 km | T: 0.02 h	D: 3.05 km | T: 0.03 h	D: 3.10 km | T: 0.03 h
A* 3	D: 2.70 km | T: 0.02 h	D: 2.82 km | T: 0.02 h	D: 3.05 km | T: 0.03 h	D: 3.10 km | T: 0.03 h
A* 4	D: 2.70 km | T: 0.02 h	D: 2.82 km | T: 0.02 h	D: 3.05 km | T: 0.03 h	D: 3.10 km | T: 0.03 h
A* 5	D: 2.70 km | T: 0.02 h	D: 2.82 km | T: 0.02 h	D: 3.05 km | T: 0.03 h	D: 3.10 km | T: 0.03 h
A* 6	D: 3.13 km | T: 0.03 h	D: 3.13 km | T: 0.03 h	D: 3.47 km | T: 0.03 h	D: 3.50 km | T: 0.03 h
Greedy 1	D: 2.96 km | T: 0.02 h	D: 3.29 km | T: 0.03 h	D: 3.53 km | T: 0.03 h	D: 3.37 km | T: 0.03 h
Greedy 2	D: 2.96 km | T: 0.02 h	D: 3.29 km | T: 0.03 h	D: 3.53 km | T: 0.03 h	D: 3.37 km | T: 0.03 h
Greedy 3	D: 2.96 km | T: 0.02 h	D: 3.29 km | T: 0.03 h	D: 3.53 km | T: 0.03 h	D: 3.37 km | T: 0.03 h
Greedy 4	D: 2.96 km | T: 0.02 h	D: 3.29 km | T: 0.03 h	D: 3.53 km | T: 0.03 h	D: 3.37 km | T: 0.03 h
Greedy 5	D: 3.12 km | T: 0.03 h	D: 3.62 km | T: 0.03 h	D: 3.63 km | T: 0.03 h	D: 3.47 km | T: 0.03 h
Greedy 6	D: 2.96 km | T: 0.02 h	D: 3.29 km | T: 0.03 h	D: 3.53 km | T: 0.03 h	D: 3.37 km | T: 0.03 h

Heurísticas usadas para os algoritmos de procura informada: 1- manhattan; 2- time estimation; 3- blocked route; 4- dynamic supply priority; 5- delivery sucess probability; 6- final combined.

O cenário 1 representa as condições ideias de procura, sem qualquer tipo de restrição nas rotas ou condições adversas. Serve como base para comparar com os outros cenários e permite verificar o desempenho dos algoritmos em condições normais.

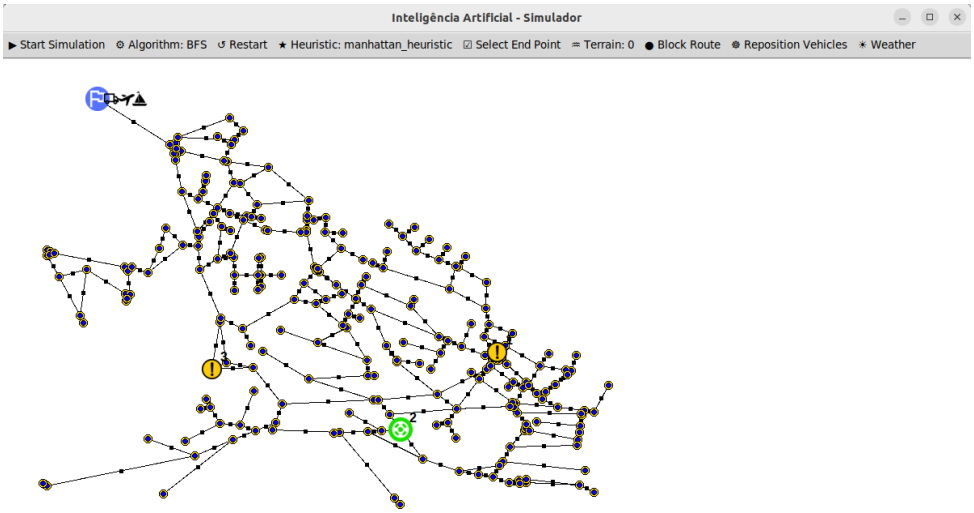

Figura 1: Cenário 1 - Controlo

No cenário 2, algumas rotas específicas foram bloqueadas, de modo que os algoritmos são obrigados a encontrar caminhos alternativos. Os bloqueios incluem ligações como '2,111', '71,74', '106,126', entre outras. Esta situação simula restrições como obras ou acidentes.

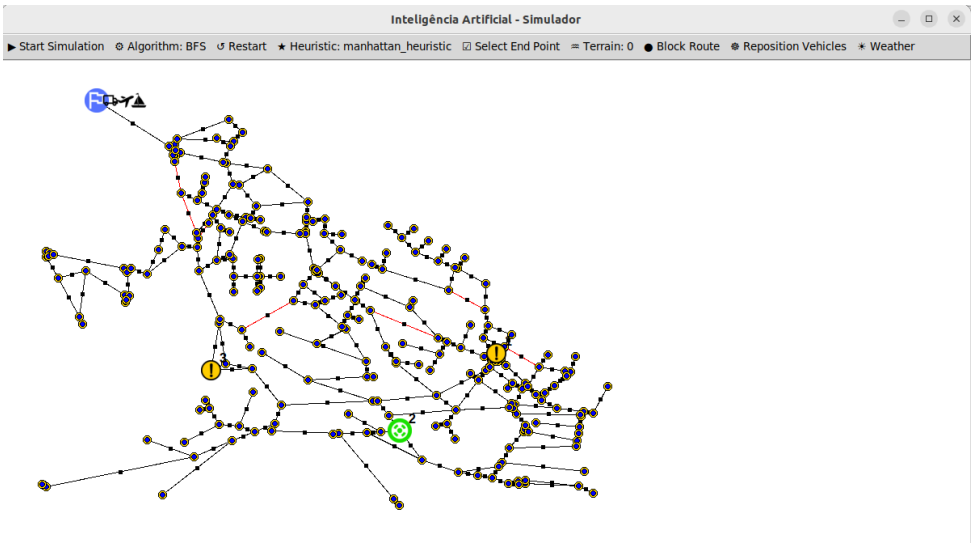

Figura 2: Cenário 2 - Rotas bloqueadas

No cenário 3, são consideradas adversidades como chuva intensa ou neve, que tornam certas rotas mais difíceis de percorrer. Embora quase todas as rotas permaneçam acessíveis, o custo de algumas aumenta significativamente, impactando as decisões dos algoritmos.

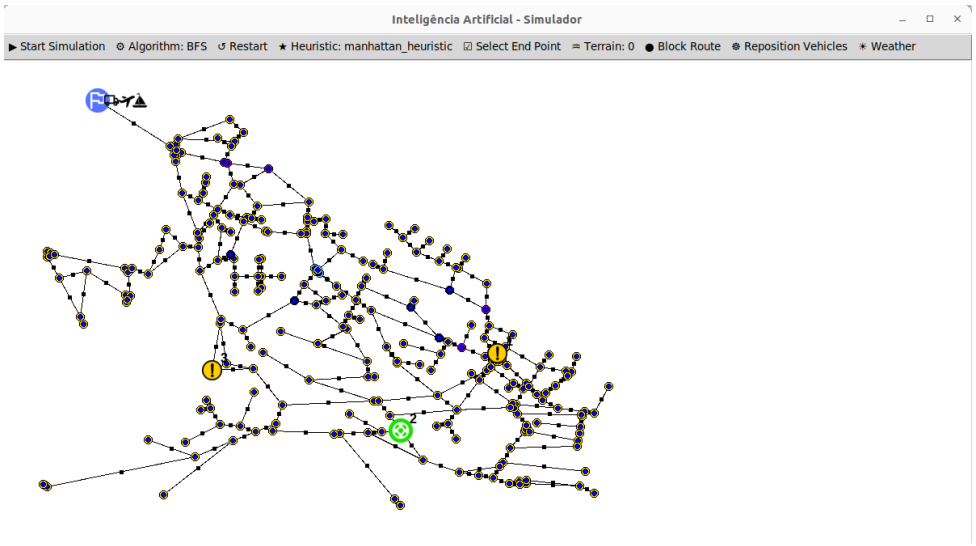

Figura 3: Cenário 3 - Condições climatéricas

O cenário 4 é o mais desafiante, porque combina rotas bloqueadas e condições climatéricas adversas. Os algoritmos precisam de lidar simultaneamente com as restrições físicas e os custos aumentados em algumas rotas, tornando o problema mais complexo.

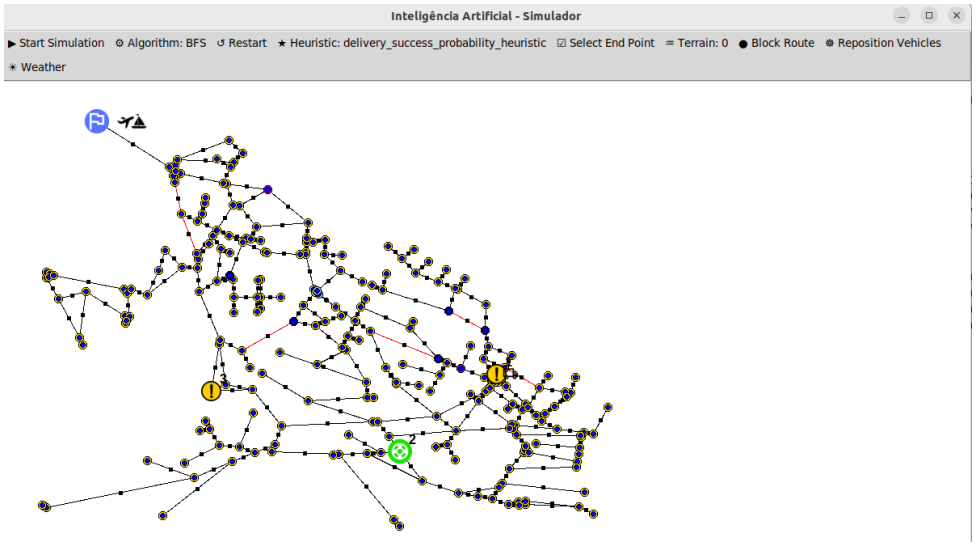

Figura 4: Cenário 4 - Rotas bloqueadas e condições climatéricas

Os resultados obtidos permitem analisar o desempenho dos algoritmos nos diferentes cenários propostos. O BFS destacou-se entre os algoritmos de procura não informada, apresentando as menores distâncias percorridas e tempos de execução em todos os cenários, enquanto o DFS mostrou os piores desempenhos, refletindo a sua natureza de pesquisa em profundidade e não otimizada. O IDS apresentou um equilíbrio entre os dois, com resultados melhores que o DFS, mas ainda inferiores ao BFS. As diversas heurísticas do A* tiveram desempenhos similares na maioria dos cenários, mostrando que todas estão bem ajustadas ao problema. Já o *Greedy Search*, embora eficiente em alguns casos, mostrou resultados menos consistentes, especialmente sob condições desafiadoras.

No cenário de controlo (cenário 1), sem restrições, todos os algoritmos alcançaram seus melhores resultados, enquanto que no cenário 2 (rotas bloqueadas), houve um impacto perceptível nas distâncias percorridas por algoritmos como o BFS e o IDS. No cenário 3 (condições climatéricas adversas), a variação de desempenho foi mais significativa, com algoritmos como o DFS e o Greedy sendo os mais afetados. O cenário 4, que combina as restrições dos cenários 2 e 3, revelou-se o mais complexo, mas o A* conseguiu manter a sua eficiência, destacando-se como a melhor solução em situações adversas. De forma geral, algoritmos como o *Uniform Cost Search* e o A* provaram ser as escolhas mais robustas e consistentes para encontrar caminhos ótimos, enquanto que os outros métodos de procura não informada e o *Greedy Search* foram mais sensíveis às condições impostas pelos cenários.

8. Conclusão

Concluindo, acreditamos ter alcançado todos os objetivos apresentados para o desenvolvimento do projeto. Através do mesmo, pudemos melhorar os nossos conhecimentos sobre formulação de problemas, algoritmos de procura não informada, mas também de procura informada, indo além do lecionado nas aulas da unidade curricular.