



Universidade do Minho  
Licenciatura em Engenharia Informática  
2024 / 2025

*Inteligência Artificial*

**Resolução de Problemas**

***Algoritmos de procura***

Frederico Cunha Afonso - a104001  
Luis Enrique Díaz De Freitas - a104000  
Rui Pedro Pires de Sá Cerqueira - a104006

# Índice

<b>Avaliação pelos pares.....</b>	<b>1</b>
<b>Introdução.....</b>	<b>2</b>
<b>Formulação do Problema.....</b>	<b>2</b>
<b>Modelação do Problema.....</b>	<b>3</b>
Mapa Definido: Recursos Necessários e Tempos de Vida.....	3
Veículos: Propriedades e as suas Limitações em Valência.....	3
Grafos relativos a cada Veículo.....	4
Clima e Logística: A Relação Entre Condições Meteorológicas e Custos.....	5
Grafos relativos a cada Clima.....	5
<b>Algoritmos de Procura: Estratégias e Atributos.....</b>	<b>6</b>
<b>Procura não Informada (cega).....</b>	<b>6</b>
Pesquisa em Largura (Breadth-first Search).....	6
Pesquisa em Profundidade (Depth-First Search).....	6
<b>Procura Informada (heurística).....</b>	<b>6</b>
Procura Gulosa – Greedy-Search.....	6
Procura heurística informada – Procura A*.....	6
<b>Resultados Obtidos dos Algoritmos de Procura.....</b>	<b>7</b>
<b>Critério de Decisão.....</b>	<b>7</b>
<b>Caso de Estudo 1.....</b>	<b>8</b>
Problema.....	8
Resultados obtidos.....	9
Conclusão.....	9
<b>Caso de Estudo 2.....</b>	<b>9</b>
Problema.....	9
Resultados obtidos.....	10
Conclusão.....	10
<b>O Sistema de Turnos e a Escolha do Utilizador.....</b>	<b>10</b>
<b>Alterações Meteorológicas: Impacto nas Rotas e Custos de Transporte.....</b>	<b>12</b>
<b>Conclusão.....</b>	<b>13</b>
<b>Glossário.....</b>	<b>13</b>

## Avaliação pelos pares

A104001 Frederico Cunha Afonso DELTA = 0

A104000 Luis Enrique Díaz De Freitas DELTA = 0

A104006 Rui Pedro Pires de Sá Cerqueira DELTA = 0

# Introdução

Este trabalho vem com o intuito de desenvolver e implementar algoritmos de procura que permitam otimizar a distribuição de **Recursos**<sup>1</sup> em zonas afetadas por uma catástrofe natural. Pretende-se garantir que os **Recursos** disponíveis sejam utilizados de forma eficiente, priorizando as áreas mais necessitadas.

Como métrica de tempo, temos que, após ser escolhido um **Algoritmo de Procura**, um **Município de Origem** e um **Município de Destino**, irá passar-se um **Turno**<sup>2</sup>, afetando de forma imprevisível as zonas afetadas.

Após as recentes cheias em Valência, vimos a oportunidade de usar o acontecimento como objeto de estudo para este projeto. Assim, definimos um mapa de municípios (representado por um grafo de nós), onde cada uma destas terá um determinado **Tempo de Vida**<sup>3</sup> e **Recursos Requisitados**<sup>4</sup>.

O mapa será percorrido por um dado número de **Veículos** inicialmente definidos (Estes serão as entidades que irão distribuir os **Recursos**) num certo **Clima**<sup>7</sup>. Cada um terá uma quantidade máxima de **Recursos** a distribuir e **Combustível** a gastar única.

## Formulação do Problema

**Tipo:** Problema de Procura.

**Estado Inicial:** Um município localizado no mapa (caso seja o primeiro turno, o município depende do input do utilizador, caso contrário, será o **Município de Destino** da travessia anterior do **Veículo**).

**Estado Objetivo:** é alcançado quando, dado um **Município de Origem**, um dos **Veículos** consegue viajar até ao **Município de Destino**, repartindo os recursos disponíveis ao maior número de municípios visitados durante a viagem.

**Operadores:**

- **Mover-se a um Município adjacente:**

**Pré-Condição:**

- Dado um nó (**Município de Origem**) onde se encontra o **Veículo**, deve haver uma ligação com outro nó (**Município de Destino**) adjacente a este;
- O **Tempo de Vida** do **Município de Destino** deve ser superior a 0;
- O **Veículo** deve ter mais **Combustível** que o **Custo do Arco**<sup>5</sup>.

**Efeito:** O **Veículo** muda de localização no mapa, estando agora no nó adjacente que tem uma ligação com o município anterior.

**Custo da solução:** O custo será calculado segundo o **Algoritmo de Procura** usado para a resolução do problema. Posteriormente abordaremos estes algoritmos com mais detalhe.

# Modelação do Problema

## Mapa Definido: Recursos Necessários e Tempos de Vida

Para representar Valência e alguns dos seus **Municípios**, estabelecemos um mapa de nós com um total de 18 **Municípios** presentes (Serra, Rocafort, Requena, Picaña, Oliva, Fontanares, Gandia, Lliria, Jarafuel, Torrent, Llutxent, Massamagrell, Valencia, Barxeta, Alcàsser, Dos Aguas, L'Ènova, Calles). Este mapa não tem nenhum arco predefinido, falaremos mais aprofundadamente sobre este tema no próximo capítulo.

Cada um destes nós terá um dado **Tempo de Vida** (sendo este a heurística dos nós) e uma certa quantidade de **Recursos Requisitados**. Até serem entregues todos os **Recursos** a cada nó, o **Tempo de Vida** destes irá decrementar até chegar a 0, tornando-se impossível o socorrer do **Município** e não sendo possível alcançá-lo de maneira alguma. Pelo contrário, se lhe forem entregues os **Recursos Requisitados** a tempo, iremos considerar o município salvo.

O **Tempo de Vida** destes será usado pelos **Algoritmos de Procura informados** desenvolvidos como prioridade e, caso um **Veículo** tenha uma maior quantidade de **Recursos** do que a quantidade de **Recursos Requisitados** pelo **Município de Destino**, este distribuirá os restantes **Recursos** pelos **Municípios** no caminho, dando prioridade a aqueles com um menor **Tempo de Vida**.

## Veículos: Propriedades e as suas Limitações em Valência

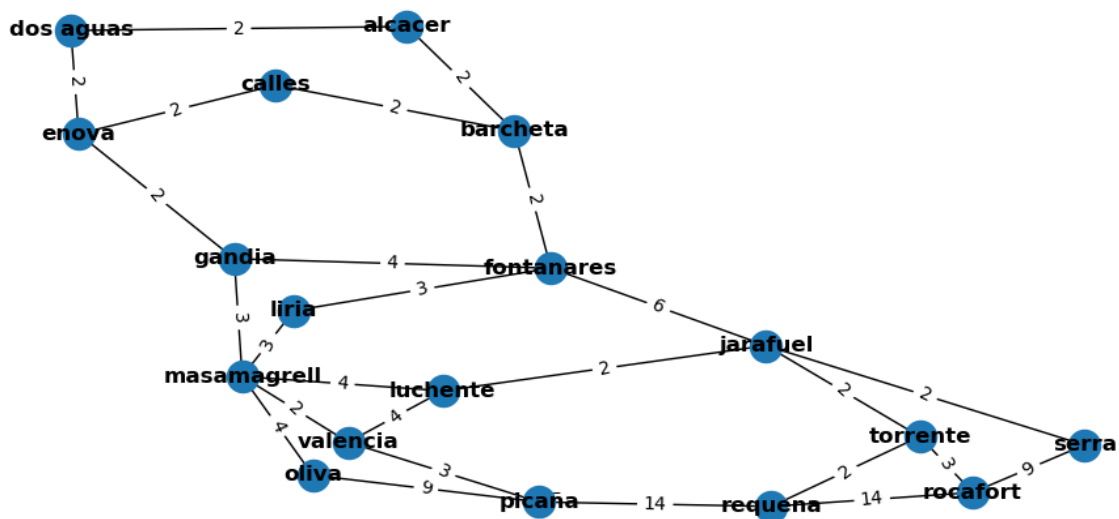
Para satisfazer as exigências de cada **Município**, entendemos que o mais apropriado seria definir 3 diferentes tipos de **Veículo** (**Avião**, **Barco**, **Carro**), cada um com uma diferente **Capacidade** <sup>6</sup> máxima de **Combustível** e **Recursos** diferentes.

```
aviao = Aviao(45,80)
carro = Carro(65,50)
barco = Barco(100,100)
```

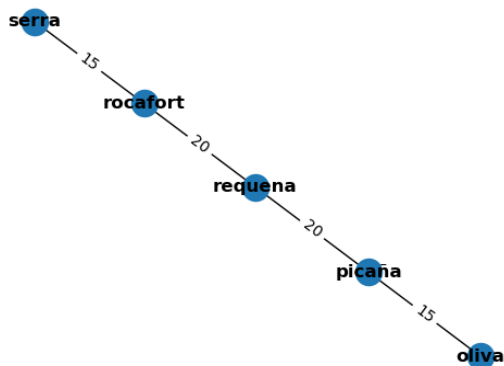
Nem todos os **Veículos** são capazes de acessar a todos os nós do mapa, devido a limitações (físicas) de acesso a certas áreas (sendo o **Avião** o único veículo capaz de alcançar todos os nós). Assim, apesar de todos percorrerem o mesmo mapa, entendemos que seria uma implementação mais simples, direta e acessível se fizéssemos um grafo por **Veículo**, baseado no mapa concebido originalmente.

Apesar desses grafos poderem distinguir-se pelos nós, arcos e **Custos dos Arcos** que lhes são exclusivos, todos partilham informações comuns relacionadas a cada **Município** do mapa, como o **Tempo de Vida** e **Recursos Requeridos**.

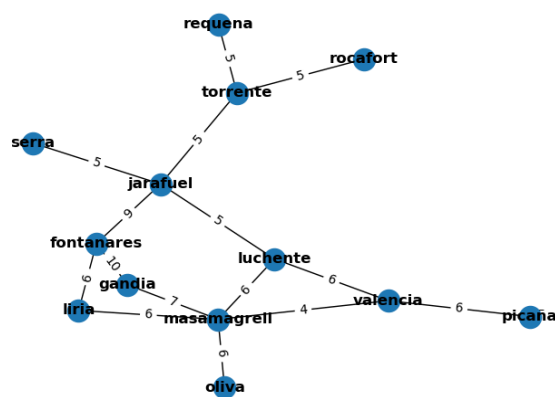
### Grafos relativos a cada Veículo



Grafo do Avião



Grafo do Barco



Grafo do Carro

## Clima e Logística: A Relação Entre Condições Meteorológicas e Custos

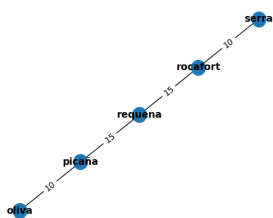
Os grafos apresentados anteriormente representam os **Municípios** do mapa que podem ser atendidos por um dado **Veículo**, mas um fator que não se identifica tão facilmente nestes, é o **Clima**.

Visto que as condições meteorológicas estão em constante mudança, foram definidos 3 grafos por **Veículo** (**Clima Básico**, **Clima Regular** e **Clima Extremo**), sendo estes:

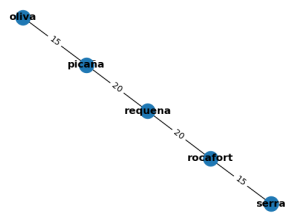
- **Clima Básico:** o clima com o menor custo de transporte;
- **Clima Regular:** com custos superiores aos do Clima Básico;
- **Clima Extremo:** o clima mais severo, tendo os custos de transporte mais dispendiosos.

No contexto dos grafos dos **Climas**, apesar dos nós presentes serem sempre os mesmos (de **Clima** para **Clima**), estes diferem nos **Custos dos Arcos**, assim aumentando com o piorar do **Clima**.

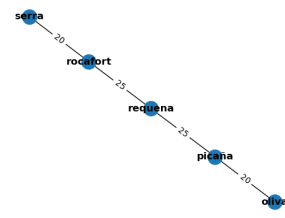
### Grafos relativos a cada Clima



[Barco] Clima Básico



[Barco] Clima Regular



[Barco] Clima Extremo

A seleção do **Clima** é aleatória, tendo cada um destes uma probabilidade de 33% de acontecer.

```
gBasico = [gAviao_climaBasico, gCarro_climaBasico, gBarco_climaBasico]
gRegular = [gAviao_climaRegular, gCarro_climaRegular, gBarco_climaRegular]
gExtremo = [gAviao_climaExtremo, gCarro_climaExtremo, gBarco_climaExtremo]

cidadesVisitadas = []

numAleatorio = random.randint(0, 2)

#escolhemos o clima
match numAleatorio:
    case 0:
        g = gBasico
    case 1:
        g = gRegular
    case 2:
        g = gExtremo
```

# Algoritmos de Procura: Estratégias e Atributos

## Procura não Informada (cega)

### Pesquisa em Largura (Breadth-first Search)

- **Estratégia:** Todos os nós de menor profundidade são expandidos primeiro;
- **Aspetos Positivos:** Procura muito sistemática;
- **Aspetos Negativos:** Normalmente demora muito tempo e sobretudo ocupa muito espaço.

### Pesquisa em Profundidade (Depth-First Search)

- **Estratégia:** Expandir sempre um dos nós mais profundos da árvore;
- **Aspetos Positivos:** Muito pouca memória necessária, bom para problemas com muita soluções;
- **Aspetos Negativos:** Não pode ser usada em árvores com profundidade infinita, pode ficar presa em ramos errados.

## Procura Informada (heurística)

### Procura Gulosa – Greedy-Search

- **Estratégia:** Expandir o nó que parece estar mais perto da solução;
- **Custo:**  $h(n)$  = custo estimado do caminho mais curto do estado  $n$  para o objetivo (função heurística);
- **Exemplo:**  $h(n)$  = distância em linha reta entre  $n$  e o objetivo.

### Procura heurística informada – Procura A\*

- **Estratégia:** evitar expandir caminhos dispendiosos. O **algoritmo A\*** combina a **Procura Gulosa** com a **Uniforme**, minimizando a soma do custo do caminho já percorrido com a estimativa do custo restante até à solução. Usa a função:

$$f(n) = g(n) + h(n)$$

**$g(n)$ :** custo total do caminho já percorrido para chegar ao nó  $n$ , partindo de um nó inicial (custo do percurso efetuado);

**$h(n)$ :** estimativa do custo para chegar a  $n$  partindo do nó atual. Esta heurística não deve superestimar o custo real para chegar à solução;

**$f(n)$ :** custo estimado da solução menos dispendiosa, com destino  $n$ .

- **Custo:**  $f(n)$ .

## Critério de Decisão

Uma vez escolhido o **Algoritmo de Procura** a utilizar e dado um **Nó de Origem** e **Destino** para a solução do problema, é calculado o custo do melhor caminho encontrado para os 3 **Veículos**. Após verificar quais os **Veículos** capazes de partir do **Nó de Origem** e de chegar ao **Nó de Destino**, no caso haja um impasse, serão comparados os resultados de cada veículo e, dependendo do **Critério de Decisão** <sup>8</sup> escolhido, será escolhido 1 deles para efetuar o seu percurso

```
=====
Qual deve ser o Critério de Decisão caso mais de um veículo
seja capaz de efetuar a viagem proposta?

      (1) Menor quantidade de Combustível Gasto
      (2) Maior quantidade de Recursos Distribuidos

[Introduza a sua opcao]: |
```



# Caso de Estudo 1

## Problema

Vamos supor que, num **Clima Regular**, vamos:

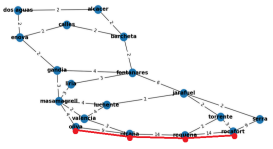
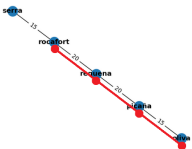
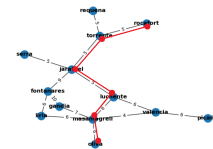
(1) partir de **rocafort**

(2) chegar a **oliva**

(3) usar a **Pesquisa em Largura** como **Algoritmo de Procura**

(4) ter como **Critério de Decisão** a **Menor Quantidade de Combustível Gasto**.

## Resultados obtidos

	Avião	Barco	Carro
Recursos Distribuídos	70	40	50
Combustível Gasto	40	40	35
Caminho Solução	['rocafort', 'requena', 'picaña', 'oliva']	['rocafort', 'requena', 'picaña', 'oliva']	['rocafort', 'torrente', 'jarafuel', 'luchente', 'masamagrell', 'oliva']
Mapa			

## Conclusão

Visto que o **Carro** foi o **Veículo** que gastou a menor quantidade de **Combustível** dos 3, foi escolhido para fazer a travessia, distribuindo **Recursos** a oliva e, subsequentemente, aos municípios no caminho com um menor **Tempo de Vida**.

```
=====
O melhor veículo para esta viagem é o [ Carro ]

{Recursos Distribuídos: 50 }
{Combustível Gasto: 27 }

Caminho solução
['rocafort', 'torrente', 'jarafuel', 'luchente', 'masamagrell', 'oliva']
=====
```

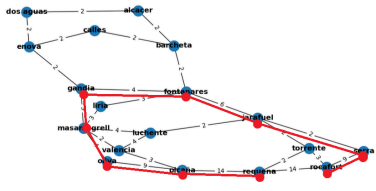
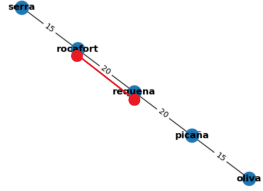
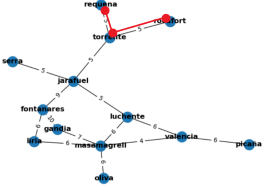
## Caso de Estudo 2

### Problema

Vamos supor que, num **Clima Regular**, vamos:

- (1) partir de **rocafort**
- (2) chegar a **requena**
- (3) usar a **Pesquisa em Profundidade** como **Algoritmo de Procura**
- (4) ter como **Critério de Decisão** a **Maior quantidade de Recursos Distribuídos**.

### Resultados obtidos

	Avião	Barco	Carro
Recursos Distribuídos	80	20	30
Combustível Gasto	51	20	10
Caminho Solução	['rocafort', 'serra', 'jarafuel', 'fontaneres', 'gandia', 'masamagrell', 'oliva', 'picaña', 'requena']	['rocafort', 'requena']	['rocafort', 'torrente', 'requena']
Mapa			

### Conclusão

Apesar do **Avião** ter distribuído mais **Recursos** que o **Barco** ou o **Carro**, a **Capacidade de combustível** deste é de **45**, superada pela quantidade de **Combustível** gasto, tornando a travessia impossível para o **Avião**.

Assim sendo, mais uma vez, será o **Carro** o escolhido, visto que foi aquele que distribuiu a maior quantidade de **Recursos** dos 3.

```
=====
O melhor veículo para esta viagem é o [ Carro ]

{Recursos Distribuídos: 30 }
{Combustível Gasto: 10 }

Caminho solução
['rocafort', 'torrente', 'requena']
=====
```

# O Sistema de Turnos e a Escolha do Utilizador

Falando mais concretamente sobre a implementação, após iniciar o programa, será apresentado este menu de decisões ao utilizador.

```
=====
( 0) Sair
( 1) Imprimir Grafo
( 2) Desenhar Grafo
( 3) Imprimir nodos do Grafo
( 4) Imprimir arestas do Grafo
( 5) Ver Recursos Requisitados pelas cidades
( 6) Ver Tempo de Vida restante das cidades
( 7) Ver Estatísticas
( 8) Definir o Critério de Decisão
( 9) Procura DFS
(10) Procura BFS
(11) Procura A*
(12) Procura Gulosa
=====
[Introduza a sua opcao]:
```

Aqui, ele irá decidir se quer:

- (1 - 4) Melhor visualizar o mapa de Valência e os acessos a diferentes **Municípios**;
- (5 - 6) Ver o estado atual das **Municípios** no decorrer da situação em curso;
- ( 7 ) Abrir o registo de métricas atuais;

```
=====
Avião {Recursos: [80/80] (100.00%)}
      {Combustível:[45/45] (100.00%)}
Barco {Recursos: [100/100] (100.00%)}
      {Combustível:[100/100] (100.00%)}
Carro {Recursos: [50/50] (100.00%)}
      {Combustível:[65/65] (100.00%)}

Recursos Disponíveis: [200/200] (100.00%)
Combustível Disponível: [300/300] (100.00%)

Recursos Requisitados Entregues: [0/155] (0.00%)
Cidades Socorridas: [0/18] (0.00%)
Cidades Perdidas: [0/18] (0.00%)

Cidades em Estado Emergente: [1/18] (5.56%)
Cidades em Estado Estável: [2/18] (11.11%)
Cidades em Estado Alarmante: [10/18] (55.56%)
Cidades em Estado Crítico: [5/18] (27.78%)
=====
```

- ( 8 ) Definir qual o **Critério de Decisão** a que o programa deve dar prioridade;
- (9 -12) Escolher que **Município** pretende socorrer, indicando uma **Origem** e um **Destino**;

As primeiras 8 ações não afetam de maneira alguma o mapa ou os **Municípios** nele representados (estas podem ser vistas como o precedente da ação a ser tomada).

As restantes levarão a uma possível alteração dos **Recursos** e **Combustível Disponível** (caso uma viagem proposta seja bem sucedida) e o **Tempo de Vida** e o **Custo do Arco** entre municípios. Isto deve-se ao **Sistema de Turnos** implementado.

Acreditamos que implementar um sistema assim seria melhor para avaliar e estudar os resultados obtidos dos **Algoritmos de Procura** lecionados, do que implementar um programa que tome todas as decisões de que rotas deveria percorrer e que execute todas essas, do início ao fim, até socorrer todos os **Municípios** ou até exaustar as possíveis escolhas.

Após o utilizador indicar o **Algoritmo de Procura** e o caminho que pretende que seja percorrido, irá acontecer uma de duas:

- Um ou mais **Veículos** serão capazes de fazer a travessia e, após escolher qual deles a fará, irá entregar os **Recursos** que tiver consigo (após isto, a próxima vez que lhe for dado um **Destino**, independentemente da **Origem** dada, assumirá o **Destino** da travessia anterior como a **Origem** da nova travessia);
- Nenhum dos **Veículos** é capaz de realizar a viagem e, como consequência, todos os **Veículos** serão reabastecidos, tanto com **Combustível** quanto com **Recursos**.

Independentemente do resultado obtido, após ser realizada a opção de procura escolhida, virá o fim do **Turno** e com este, o **Tempo de Vida** dos **Municípios** não visitados será decrementado, fazendo com que os **Algoritmo de Procura Informada** deem prioridade a passar por municípios ainda não satisfeitos com um menor **Tempo de Vida** (heurística).

Criando um senso de urgência não derivado de um tempo limitado, mas das decisões tomadas pelo utilizador, enquanto lhe é concedido todo o tempo necessário para agir.

```
=====
Novos Tempos de Vida das cidades
{'alcacer': 2, 'barcheta': 2, 'calles': 1, 'dos aguas': 4, 'enova': 1, 'fontaneres': 1,
 'gandia': 2, 'liria': 3, 'jarafuel': 3, 'luchente': 2, 'masamagrell': 3, 'oliva': 2, '
 picaña': 2, 'requena': 2, 'rocafort': 3, 'serra': 2, 'torrente': 4, 'valencia': 2}
=====

=====
Nodo Inicial para a próxima viagem do Carro : oliva
=====
```

# Alterações Meteorológicas: Impacto nas Rotas e Custos de Transporte

Por último, temos as **Mudanças Meteorológicas**, que têm uma probabilidade de 33% de acontecer no fim de cada **Turno**. Estas mudanças bloqueiam alguns dos acessos entre municípios no mapa.

```
=====
Nodo Inicial para a próxima viagem do Carro : oliva
=====
Aconteceu um evento entre barcheta e fontanares
=====
```

Para representar este “bloqueio”, baseamos-nos na Unidade Curricular de **Comunicações por Computador**, mais especificamente, no método de **envenenamento de rota**, aumentando assim o **Custo do Arco** dos **Municípios** escolhidos para um valor extremamente alto (p.e. 999).

Isto fará com que os **Algoritmos de Procura**, como a **Procura Greedy** ou a **Procura A\***, escolham qualquer outro caminho excepto aqueles com um maior custo (e mesmo que este caminho seja escolhido, nenhum **Veículo** tem **Combustível** suficiente para atravessar tal acesso).

```
def eventoAleatorio(self):
    numeroRandom = random.randint(1, 3)
    if numeroRandom == 3: #33% de acontecer un evento
        nodo_inicial = random.choice(list(self.m_graph.keys()))

        # Verifica si la lista de vecinos no está vacia
        if self.m_graph[nodo_inicial]:
            # Selecciona un vecino aleatorio del nodo inicial
            nodo_adyacente, _ = random.choice(self.m_graph[nodo_inicial])

            # Actualiza el peso del nodo adyacente en el nodo inicial
            for i, (vecino, valor) in enumerate(self.m_graph[nodo_inicial]):
                if vecino == nodo_adyacente:
                    self.m_graph[nodo_inicial][i] = (vecino, 999)
                    break

            # Verifica si el nodo adyacente existe en el grafo
            if nodo_adyacente in self.m_graph:
                # Actualiza el peso del nodo inicial en el nodo adyacente
                for i, (vecino, valor) in enumerate(self.m_graph[nodo_adyacente]):
                    if vecino == nodo_inicial:
                        self.m_graph[nodo_adyacente][i] = (vecino, 999)
                        break
            else:
                print(f"Advertencia: El nodo adyacente '{nodo_adyacente}' no está en el grafo.")
        else:
            print(f"Advertencia: El nodo inicial '{nodo_inicial}' no tiene vecinos.")
            nodo_adyacente = None

        return True, nodo_inicial, nodo_adyacente
    else: return False, None, None
```

## Conclusão

Concluindo, com a realização deste trabalho, aplicamos uma grande parte da matéria lecionada no decorrer desta Unidade Curricular, verificando a eficiência e utilidade de diferentes **Algoritmos de Procura**.

Em termos de implementação, ficamos satisfeitos com as decisões tomadas, especialmente com a implementação do **Sistema de Turnos**, as **Alterações Meteorológicas**, os **Critérios de Decisão**, a **Modulação e Formulação do Problema**.

# Glossário

<sup>1</sup> **Recursos:** alimentos, água e medicamentos.

<sup>2</sup> **Turno:** Ação que o utilizador toma, num qualquer período de tempo.

- Neste projeto, as ações que evocam o fim de um **Turno** seriam as ações relacionadas aos **Algoritmos de Procura**.
- A quantidade de turnos feitos é usada como métrica de tempo.

<sup>3</sup> **Tempo de Vida:** Cada município vai perdurar durante um certo número de **Turnos** até se perder a possibilidade de atendê-la, sendo mais tarde registado como **Perdido**.

<sup>4</sup> **Recursos Requisitados:** Quantidade de **Recursos** que cada município em necessidade vai estar à espera que lhe sejam entregues.

<sup>5</sup> **Custo do Arco:** Quantidade de **Combustível** a ser gasto por um **Veículo** para passar do **Município** em que se encontra para um **Município** adjacente.

- Alternativa: “custos de transporte”.

<sup>6</sup> **Capacidade:** Quantidade máxima de **Recursos** ou **Combustível** que um **Veículo** consegue carregar.

<sup>7</sup> **Clima:** Condições meteorológicas que afetam o **Custo dos Arcos**.

<sup>8</sup> **Critério de Decisão:** Após serem aplicados os **Algoritmos de Procura**, no caso de mais de um **Veículo** ser capaz de efetuar o percurso, será escolhido o **Veículo** que melhor atende ao **Critério de Decisão**.

- Atualmente só foram implementados 2 **Critérios**, estes são:
  - O que visa a menor quantidade de **Combustível Disponível** a ser gasto;
  - O que visa a maior quantidade de **Recursos Disponíveis** a serem distribuídos.