A picture containing shape

Description automatically generated

**Relatório Sprint B**

**Turma 3DG-Grupo 37**

1161274- Artur Muiria

1190827- Luís Araújo

1191240- Danilton Lopes

1201183- João Oliveira

**Professor:**

António Silva, ASS

**Unidade Curricular:**

Algoritmia Avançada

**Data: 04/12/2022**

Índice

[Introdução 1](#_Toc121079604)

[Representação do conhecimento do domínio 1](#_Toc121079605)

[Obtenção da solução ótima para o Planeamento da Entrega de Mercadorias com 1 camião elétrico 2](#_Toc121079606)

[Estudo da complexidade do problema e da viabilidade 3](#_Toc121079607)

[Heurísticas para geração rápida de soluções 5](#_Toc121079608)

[1. Heurística pelo armazém mais próximo 5](#_Toc121079609)

[2. Heurística pela entrega de maior massa 6](#_Toc121079610)

[3. Heurística a combinar as duas heurísticas anteriores 7](#_Toc121079611)

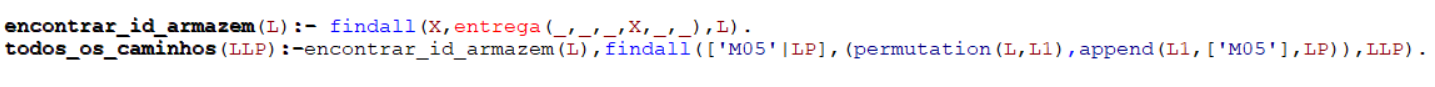
[Análise da Qualidade das Heurísticas 7](#_Toc121079612)

[Conclusões 8](#_Toc121079613)

# Introdução

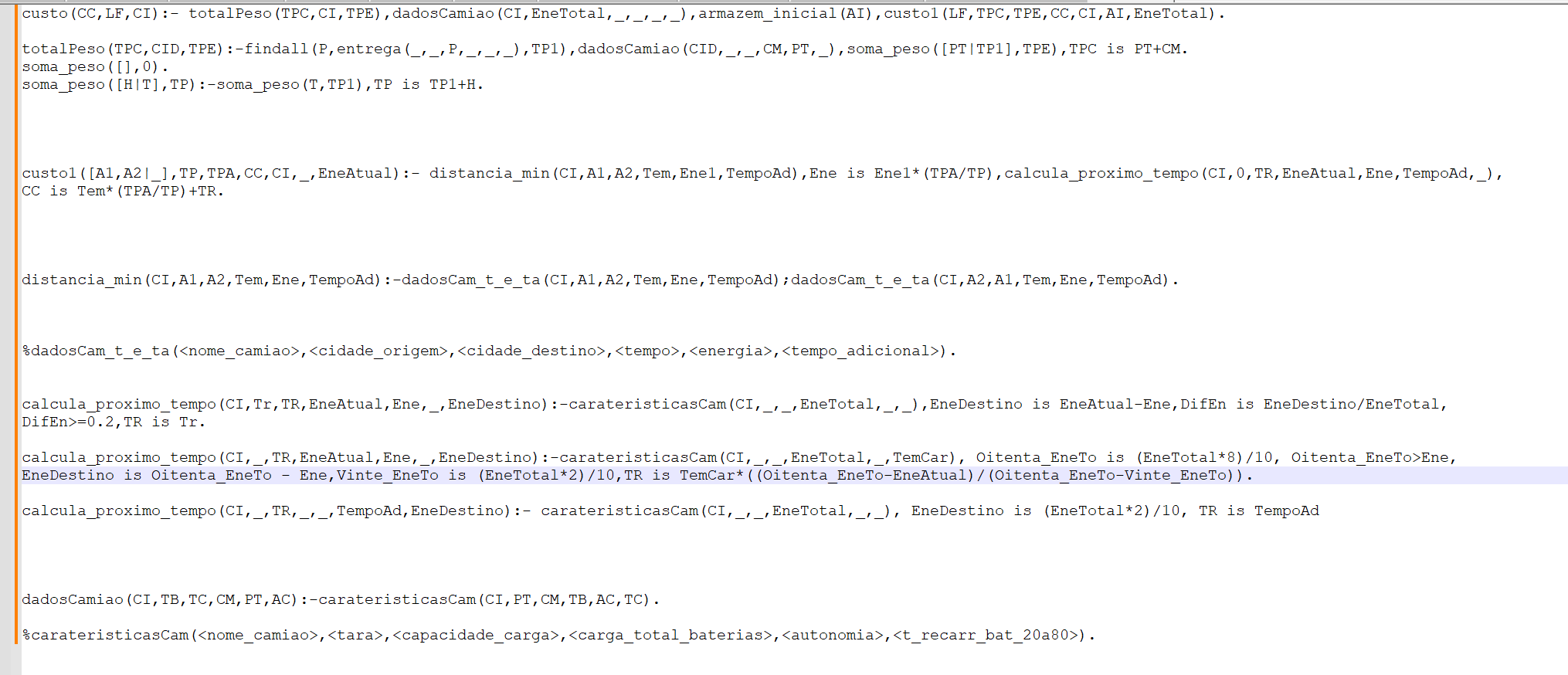
O presente relatório funcionará como apoio e explicação de execução das User Stories pedidas para o Sprint B de ALGAV.

# Representação do conhecimento do domínio

De facto, para este exercício foi pedido que se encontrassem **todas** as **trajetórias** possíveis através de **sequências** de armazéns onde deverão ser feitas as entregas. Para isso considerou-se que o armazém de **Matosinhos** seria o ponto de **partida** e o de **chegada**, havendo a **permutações** dos restantes armazéns.

Como se observa na imagem acima, utilizou-se o predicado ***todos\_os\_caminhos*** que chama o predicado ***encontrar\_id\_armazem***. Este último encontra o ***id*** dos armazéns presentes nas **entregas** que estão na base de conhecimento. Como se verifica, o **X** na entrega corresponde ao ***id*** do armazém. Após isso, é feito um ***findall*** seguido de um ***permutation*** de forma a se encontrarem **todas** as soluções. De notar que se coloca o ***M05*** na cabeça e na cauda da lista final sendo este ***id*** correspondente ao armazém de **Matosinhos**.

# Obtenção da solução ótima para o Planeamento da Entrega de Mercadorias com 1 camião elétrico



Foram criados alguns predicados para obter a solução ótima para o planeamento de entregas. No predicado **totalPeso** utilizamos o **findall** para encontrar todas as entregas e colocar o peso de todas as entregas na lista **TP1**. Posteriormente, é chamado o predicado **soma\_peso** que no primeiro argumento tem uma lista com as massas das várias entregas na tail e a tara do camião na head e no segundo argumento uma variável que ira guardar o valor total das massas das entregas mais a tara do camião (**TPE**). No final do predicado atribuímos a **TPC** a soma da tara com a capacidade de carga do camião.

Seguidamente, temos a chamada do predicado **custo1**, no qual utilizamos o predicado **distancia\_min** para obter a distância entre 2 armazéns, de seguida damos a variável **Ene** o valor da multiplicação entre a energia do percurso e o quociente da divisão entre **TPE** e **TPC** (explicadas anteriormente).

Com esse novo valor de energia iremos chamar o predicado **calcula\_proximo\_tempo** neste predicado atribuímos o valor da diferença entre a energia atual e a energia necessária até ao destino a variável **EneDestino**, que depois ira se dividir pela energia total do camião, guardando o valor em **DifEn** que se for menor que 0.2, o predicado continuara para a linha abaixo. Na segunda linha, é verificado se com 80% de carga consegue chegar ao destino, se não conseguir vai para a linha em baixo que calcula o tempo adicional na viagem. O tempo adicional na viagem traduz-se no tempo de viagem para outro armazém para fazer o carregamento e o tempo de carregamento em si.

# Estudo da complexidade do problema e da viabilidade

Foi feito um aumento na dimensão do problema, portanto colocou-se mais armazéns a visitar durante o percurso de um camião que faz entregas de armazém para armazém. O objetivo foi encontrar a solução ótima através da geração de todas as soluções e por fim estudá-la e verificar quão viável é este método.

Text

Description automatically generated

O predicado principal **encontrar\_o\_melhor\_caminho** tem 3 parâmetros onde cada argumento será uma incógnita e onde vão receber os resultados do custo da solução ótima, o melhor caminho e o tempo de geração da solução. O predicado **get\_time** retorna a hora atual como um TimeStamp. De seguida **todos\_os\_caminhos** é um predicado que foi referido acima que retorna todas as permutações de caminhos entre armazéns com o armazém inicial e final que é o de Matosinhos. O predicado **encontrar\_o\_melhor\_caminho1** tem 2 parâmetros, o primeiro argumento é uma incógnita que vai receber o resultado com uma lista de custos enquanto que o segundo argumento é a lista de permutações buscada com o predicado **todos\_os\_caminhos.** O predicado em si calcula o custo para cada caminho com o predicado **custo** que foi referido anteriormente. Depois o predicado **menorValor** tem 2 parâmetros, o primeiro argumento sendo uma lista e o segundo argumento a incógnita que busca o menor valor da lista. O predicado a seguir **encontrarPosicao** tem 3 parâmetros, o primeiro argumento contém a lista de custos, o segundo o menor custo e o terceiro é uma incógnita que obtém a posição do menor custo na lista de custos. Por fim o último predicado **nth0** tem 3 parâmetros onde o primeiro argumento é a posição do menor custo na lista de custos, o segundo a lista de caminhos e o terceiro é uma incógnita que recolhe o caminho com o menor custo. No final é feito um **get\_time** de novo e faz-se a diferença (get\_time final – get\_time inicial) desses dois predicados onde se calcula o tempo de geração da solução e coloca-se no **TSol**.

A picture containing table

Description automatically generated

Depois de analisar os valores obtidos, notou-se que o tempo de geração da solução foi aumentando de acordo com a inserção de um armazém após cada iteração. A partir dos nove armazéns o predicado ficou preso na sua execução porque este pode vir a demorar minutos, horas, dias ou até anos para obter a melhor solução para este caso e/ou para os próximos com mais armazéns, pois o número de soluções é elevado de mais para o predicado processar. Conclui-se que não é viável este método para encontrar a melhor solução e é por isso que o método superior é optar por heurísticas, apesar de não serem perfeitas conseguem pelo menos terminar a sua execução com um resultado. Foi efetuado uma tentativa de aumentar a stack do programa **SWI-Prolog** com a ajuda deste predicado “**?- set\_prolog\_flag(stack\_limit, 10 000 000 000).”** para possibilitar mais resultados mas não houve sucesso.

# 

# Heurísticas para geração rápida de soluções

Como vistos no estudo da complexidade do predicado para obtenção da solução ótima, acaba aumentando com o aumento de armazéns, passando a demorar alguns minutos e outros até horas, então para isto fora-se encomendado desenvolver heurísticas para apresentarem soluções rápidas, mesmo que estas não sejam as melhores, então foram desenvolvidas 3 heurísticas:

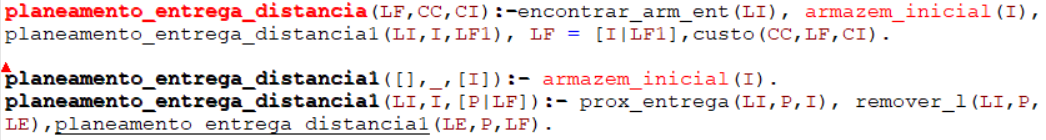
* Pela menor distância ou menor tempo;
* Pela maior massa;
* E a combinar as duas heurísticas anteriores.

## Heurística pelo armazém mais próximo

Primeiramente um predicado para buscar todos os armazéns que fizeram uma entrega e adiciona-se à lista de todos os armazéns e assim poder fazer os cálculos da distância entre os armazéns que estão na lista. Depois de adicionar todos os armazéns na lista, chama outro predicado e que vai buscar o armazém inicial a outro ficheiro e depois e chama outro predicado para dar início à lista de ordem dos armazéns para as entregas, depois de ter o resultado do custo da solução gerada em minutos.

O predicado tem 3 parâmetros, no qual uma têm o valor do id do camião, **CI**, e os outros parâmetros vão receber o valor com os passos do predicado, sendo estes LF (Lista Final) e CC (Custo do Caminho gerado).





O predicado chamado, **planeamento\_entrega\_distancia1**, tem duas condições, no qual o primeiro vê se a lista inicial dos armazéns, **primeiro parâmetro**, está vazia e adiciona a lista resultante, o **terceiro parâmetro**, o armazém inicial, ou seja, encerra as entregas no armazém inicial. O predicado tem 3 parâmetros, sendo estes **LI** (Lista Inicial), I (Armazém inicial/de origem naquele ponto) e **LF** (Lista Final).

No segundo predicado com o mesmo nome do anterior nós vemos um comportamento diferente, no qual chamada outro predicado que tem como objetivo obter o próximo armazém que é o mais próximo do armazém atual, remove o armazém encontrado da lista inicial, primeiro parâmetro, e faz recursividade chamando a si mesmo, passando a lista sem o armazém encontrado e passando como armazém atual o escolhido pela menor distancia, quando está a retornar com a condição de saída, adiciona a lista resultante.



O predicado acima, chama o predicado para ver qual o armazém com o menor tempo de distância com ele. Recebe 3 parâmetros, no qual 2 estão definidas, L (Lista de armazéns) e I (Armazém Inicial), e o parâmetro P (Próximo armazém) irá receber o valor do próximo armazém definido a seguir as instruções.

Uma imagem com texto

Descrição gerada automaticamente

No predicado vai buscar os dados ao outro ficheiro onde existem todas as distâncias medidas em minutos entre os armazéns e compara se a distancia do armazém com o anterior é menor que o que já tem, e se tiver altera, e se não tiver, passa para o predicado abaixo e faz chamada recursiva.

Com os valores dos parâmetros permite a comparação apara ver se substituí ou não o armazém atual.

Os parâmetros recebidos são: Lista dos armazéns a visitar, próximo armazém por se assumir, próximo armazém auxiliar, o armazém de origem e a distância em tempo (minutos).

## Heurística pela entrega de maior massa

Este predicado utiliza a mesma estrutura do predicado anterior, tendo a mesma ordem e formato dos parâmetros para os predicados da solução anterior, a diferença é o predicado chamado para que seja feito a sequência de armazéns a serem entregues é pelo predicado: **planeamento\_entrega\_massa**.

Uma imagem com texto

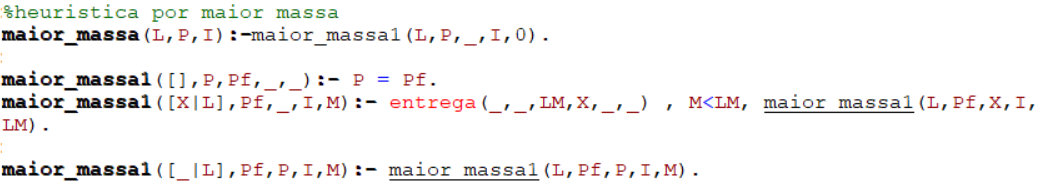
Descrição gerada automaticamente

O **planeamento\_entrega\_massa1**, tem duas condições, no qual o primeiro vê se a lista inicial dos armazéns, **primeiro parâmetro**, está vazia e adiciona a lista resultante, o **terceiro parâmetro**, o armazém inicial, ou seja, encerra as entregas no armazém inicial.

No segundo predicado com o mesmo nome do anterior nós vemos um comportamento diferente, no qual chamada outro predicado que tem como objetivo obter o armazém que tem a entregas com maior massa, remove o armazém encontrado da lista inicial, primeiro parâmetro, e faz recursividade chamando a si mesmo, passando a lista sem o armazém encontrado e passando como armazém atual o escolhido pela entrega de maior massa, quando está a retornar com a condição de saída, adiciona a lista resultante.



O predicado acima, chama o predicado para ver qual o armazém com a entrega de maior massa com ele.



No predicado vai buscar os dados ao outro ficheiro onde existem todas as distâncias medidas em minutos entre os armazéns e compara se a massa da entrega do armazém com a anterior é maior que o que já tem, e se tiver altera, e se não tiver, passa para o predicado abaixo e faz chamada recursiva.

## Heurística a combinar as duas heurísticas anteriores

Como diz o nome, esta heurística mistura os predicados desenvolvidos para as heurísticas e usa os de forma alternada. Segue a mesma estrutura das duas anteriores só se diferencia no que foi dito acima.

Uma imagem com mesa

Descrição gerada automaticamente

# Análise da Qualidade das Heurísticas

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Número de Armazéns para Entrega | Solução ótimas | Tempo para Entregas Heurística do menor tempo ou distância | Tempo para Entregas Heurística da maior massa | Tempo para Entregas Heurística combinada | Melhor solução pelas 3 heurísticas | Melhor solução |
| 5 | 409,2966102 | 0,0000286 | 0,0000286 | 0,0000278 | 378,0137931 | Massa e Tempo |
| 6 | 442,5966102 | 0,000035 | 0,0000277 | 0,0000314 | 408,0396552 | Massa e Tempo |
| 7 | 487,522 | 0,0000386 | 0,00002808 | 0,0000392 | 451,062931 | Tempo/Distancia |
| 8 | 522,2796 | 0,0000454 | 0,0000285 | 0,000042 | 487,175 | Tempo/Distancia |
| 9 | **N/A** | 0,0000527 | 0,000036 | 0,000044 | 508,7275862 | Tempo/Distancia |
| 10 | **N/A** | 0,0000625 | 0,0000395 | 0,000049 | 560,4853448 | Tempo/Distancia |
| 11 | **N/A** | 0,000066 | 0,0000425 | 0,000084 | 586,012931 | Tempo/Distancia |
| 12 | **N/A** | 0,0000675 | 0,000052 | 0,0000785 | 632,875 | Tempo/Distancia |
| 13 | **N/A** | 0,0000765 | 0,0000482 | 0,0001085 | 699,2862069 | Tempo/Distancia |
| 14 | **N/A** | 0,00008825 | 0,00005225 | 0,000775 | 734,0224138 | Tempo/Distancia |
| 15 | **N/A** | 0,00010875 | 0,00006 | 0,00089 | 784,1931034 | Tempo/Distancia |
| 16 | **N/A** | 0,00010375 | 0,00006225 | 0,0008825 | 818,1525862 | Tempo/Distancia |

O comportamento que se observa é que o predicado para a solução ótima apresenta uns valores maiores do que da melhor solução das heurísticas, um comportamento suspeito, mas estes não se diferem muito e ainda que a complexidade para a geração da solução ótima é alta, o que não compensa uma vez que não se sabe qual o volume de entregas que a EletricGO lida por dia, e se for de grande quantidade apresentará um problema uma vez que pode complicar o funcionamento dos processos da organização. Então optamos por usar apenas as heurísticas para evitar constrangimentos no funcionamento e agilização de processos.

Conclusões Após as observações feitas anteriormente conclui-se que a melhor forma de obter uma solução ótima é de especificar a categoria do percurso, se queremos por menor distância/tempo, pela maior massa ou pela menor distância/tempo com a maior massa. Esta maneira é a mais viável e pode-se usar em casos práticos do dia a dia.

A geração de solução ótima comporta-se fenomenal com poucas entregas, mas devido a sua complexidade alta aumenta o tempo para a geração de uma solução quanto ao planeamento de entrega o que complica muito uma vez que a organização deverá ter a capacidade para lidar com grandes volumes de entregas e que a solução seja gerada de forma rápida sem causar atrasos e muitas pendências. Por isso opta-se por usar as Heurísticas que tem uma complexidade de O(n2), que é menor do que a complexidade de Solução ótima, sendo esta de O(n!).