**Análise do Algoritmo de Planeamento desenvolvido no âmbito do Projeto Integrador**

***Electric Go***

**ALGAV**

AFONSO PIMENTEL – 1151399

JOSÉ PINTO – 1181654

RICARDO RIBEIRO – 1181350

RUI RAFAEL – 1181357

VÍTOR PAIVA - 1140951

**Conteúdo**

[**Distribuição de tarefas** 2](#_Toc121079133)

[**Problema** 3](#_Toc121079134)

[Algoritmo Custo Mínimo com Permutações 4](#_Toc121079135)

[Funcionamento do Algoritmo 4](#_Toc121079136)

[**Análise Algoritmo** 10](#_Toc121079137)

[**Heurísticas** 12](#_Toc121079138)

[Peso 12](#_Toc121079139)

[Funcionamento do Algoritmo 12](#_Toc121079140)

[Distância 13](#_Toc121079141)

[Funcionamento do Algoritmo 13](#_Toc121079142)

[Produto Peso Distancia 14](#_Toc121079143)

[Funcionamento do Algoritmo 14](#_Toc121079144)

[**Heurísticas x Permutações** 16](#_Toc121079145)

[**Minorantes e Majorantes** 16](#_Toc121079146)

[**Conclusão** 17](#_Toc121079147)

# **Distribuição de tarefas**

|  |  |  |
| --- | --- | --- |
| Tarefa | Link Jira | Responsável |
| Evaluate generated trajectories and choose the one the least expensive one. | [LS75-177](https://lei-isep-ricardo.atlassian.net/jira/software/projects/LS75/boards/1?selectedIssue=LS75-177) | Rui Rafael – 1181357  Afonso Pimentel – 1151399 |
| Generate all possible trajectories for 1 truck | [LS75-187](https://lei-isep-ricardo.atlassian.net/jira/software/projects/LS75/boards/1?selectedIssue=LS75-187) | Rui Rafael – 1181357  Afonso Pimentel – 1151399 |
| Increase problem dimension | [LS75-197](https://lei-isep-ricardo.atlassian.net/jira/software/projects/LS75/boards/1?selectedIssue=LS75-197) | Ricardo Ribeiro – 1181350 |
| Implement heuristics | [LS75-198](https://lei-isep-ricardo.atlassian.net/jira/software/projects/LS75/boards/1?selectedIssue=LS75-198) | José Pinto – 1181654 |
| Create a minor and max for the deliveries | [LS75-217](https://lei-isep-ricardo.atlassian.net/jira/software/projects/LS75/boards/1?selectedIssue=LS75-217) | Vítor Paiva – 1140951 |

# **Problema**

No âmbito do projeto integrador, foi solicitado o desenvolvimento de uma aplicação denominada de “Electric Go”, sendo esta aplicação, responsável por gerir 4 entidades: Armazéns, Caminhos, Entregas e Camiões.

De forma a rentabilizar o tempo e percurso que os camiões terão de percorrer para efetuar as suas entregas, utilizamos a disciplina de ALGAV para construir o algoritmo a utilizar. Para isto, teremos de ter em consideração algumas características fornecidas sobres estas entidades, sendo estas: os percursos, que já foram estudados, o tempo despendido, a distância, e bateria necessária para percorrer cada um deles. É de ressaltar que, os camiões serão elétricos, tendo cargas lentas e rápidas com tempos diferentes, e deverá, em trabalho, manter a usa bateria entre 20% e 80% para manter a sua viabilidade e assim aumentar a sua durabilidade.

Sendo que este é um exercício académico, foram efetuadas algumas simplificações relativamente à viagem do camião nomeadamente:

* Gasto de bateria linear, em relação ao peso;
* Tempo de viagem linear, em relação ao peso;
* Carregamento rápido da bateria linear, entre 20% e 80%;
* Não é considerado trânsito no percurso, o tempo dado no caminho não é afetado por fatores externos, sem ser o peso.

Deste modo, terão, ainda, de ser considerados os seguintes cenários:

* Camião não tem carga suficiente para o trajeto;
* Camião tem carga suficiente para o trajeto, mas chega lá com menos de 20% da bateria;
* Camião tem, obrigatoriamente, uma carga intermédia nesse trajeto;
* Cargas intermédias serão sempre até aos 80%;
* Camião sai sempre de Matosinhos com 100% bateria;
* Última paragem é sempre em Matosinhos com o mais próximo de 20% de bateria possível;
* Última carga pode ser otimizada para chegar ao armazém de Matosinhos com apenas 20%.

# Algoritmo Custo Mínimo com Permutações

Este algoritmo consiste em gerar todas as permutações possíveis entre os armazéns que será necessário efetuar entregas, e depois comparar o resultado para saber qual será o caminho com menor custo.

## Funcionamento do Algoritmo

O algoritmo é constituído por vários predicados com diferentes responsabilidades:

#### entrega

Armazena os seguintes valores referentes a uma entrega:

* Código
* Data
* Peso
* Armazém
* Tempo Carga
* Tempo Descarga

#### carateristicasCam

Armazena os seguintes valores referentes a um camião:

* Nome
* Tara
* Capacidade Carga
* Carga Total Baterias
* Autonomia
* Tempo Carregamento Rápido (20-80)

#### armazem\_principal

Armazena os seguintes valores referentes o armazém principal:

* Código
* Nome

#### tempo

Armazena os seguintes valores referentes ao tempo de um caminho:

* Armazém Origem
* Armazém Destino
* Tempo Com Carga Maxima

*energia*

Armazena os seguintes valores referentes a energia de um caminho:

* Armazém Origem
* Armazém Destino
* Energia Com Carga Máxima

*tempo\_extra*

Armazena os seguintes valores referentes a energia de um caminho:

* Armazém Origem
* Armazém Destino
* Tempo Extra

*custo\_min*

Armazena os seguintes valores referentes ao percurso com custo mínimo:

* Lista Ordem De Entregas
* Lista Ordem De Armazéns
* Lista Carregamentos Armazém
* Lista Quantidades Carregamento
* Lista Tempos Carregamento Armazém
* Custo

#### sequencia\_custo\_min

Este é o predicado de entrada responsável por iniciar o processo de cálculo das permutações e retornar o custo mínimo.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *ListaEntregas* | *LOrdemDeEntregas*  *LOrdemDeArmazens*  *ListaCarregamentosArmazem*  *ListaQuantidadesCarregamento*  *ListaTemposCarregamentoArmazem*  *Custo* | *run*  *custo\_min* |

#### run

Executa o processo para gerar todas as permutações da lista de entregas, calcula o custo do caminho para cada permutação e determina qual permutação tem o menor custo.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *ListaEntregas* |  | *retractall*  *assertz*  *permutation*  *encontrar\_dados\_para\_entregas*  *calcula\_custo*  *atualiza* |

#### encontrar\_dados\_para\_entregas

Determina quais os pesos e armazéns para cada entrega.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *ListaEntregas* | *ListaPesos*  *ListaArmazens* | *encontrar\_dados\_para\_entregas1*  *reverse* |

#### calcula\_custo

Calcula o custo total para uma sequência de entregas.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *ListaEntregas*  *ListaArmazens*  *ListaPesos* | *Custo*  *ListaCarregamentosArmazem*  *ListaQuantidadesCarregamento*  *ListaTemposCarregamentoArmazem* | *soma\_pesos*  *carateristicasCam*  *acrescenta\_tara*  *armazem\_principal*  *append*  *last*  *energia\_maxima*  *custo* |

#### soma\_pesos

Soma todos os pesos que o camião tem que carregar, numa determinada sequência de entregas.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *ListaArmazens*  *ListaPesos* | *ListaPesosAcumulados* | *soma\_pesos* |

#### acrescenta\_tara

Acrescenta a Tara do camião à lista de pesos que o camião tem que carregar, numa determinada sequência de entregas.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *Tara*  *ListaPesos* | *ListaPesosComTara* | *acrescenta\_tara* |

#### energia\_maxima

Obtém a energia máxima que um determinado camião consegue armazenar.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
|  | *EnergiaMaxima* | *carateristicasCam* |

#### custo

Calcula o custo para uma sequência de entregas e determina as paragens para carregamento da bateria, o tempo de carregamento e a quantidade de bateria a carregar.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *ListaEntregas*  *ListaArmazens*  *ListaPesosTotais* | *EnergiaAtual*  *Custo*  *ListaCarregamentos*  *ListaQuantidadesCarregamento*  *ListaTemposCarregamento* | *limites\_energia*  *calculo\_metricas\_trajeto*  *nao\_tem\_energia\_suficiente*  *trajeto\_atual\_e\_o\_trajeto\_retorno*  *calculo\_tempo\_carregamento*  *calcula\_tempo\_espera*  *tempo\_extra*  *custo* |

#### limites\_energia

Obtém os limites de energia do camião.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
|  | *EnergiaA80PorCento*  *EnergiaA20Porcento* |  |

#### calculo\_metricas\_trajeto

Calcula para um determinado trajeto, qual o tempo e energia necessários para o percorrer.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *ArmazemOrigem*  *ArmazemDestino* | *PesoAtual*  *Tempo*  *Energia* | *tempo*  *carga\_maxima*  *converter\_decimal\_para\_minutos*  *energia* |

#### carga\_maxima

Obtém a carga máxima que um determinado camião consegue transportar.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
|  | *CargaMaxima* | *carateristicasCam* |

#### converter\_decimal\_para\_minutos

Converte um número decimal no seu equivalente em minutos.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *TempoComoDecimal* | *TempoEmMinutos* | *float\_integer\_part*  *float\_fractional\_part* |

#### nao\_tem\_energia\_suficiente

Verifica se o camião tem energia suficiente para percorrer um determinado percurso.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *DiferencaEnergia* |  | *limites\_energia* |

#### trajeto\_atual\_e\_o\_trajeto\_retorno

Verifica se o trajeto que o camião tem que percorrer é o trajeto de retorno para o armazém principal.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *ListaArmazens* |  |  |

#### calculo\_tempo\_trajeto\_final

Calcula o tempo total do trajeto de retorno (trajeto final).

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *ListaArmazens* |  | *Entrega*  *limites\_energia* |

#### calculo\_tempo\_carregamento

Calcula o tempo de carregamento necessário para o trajeto a ser efetuado.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *CargaAtual* | *TempoCarga*  *QuantidadeCarga* |  |

#### calcula\_tempo\_espera

Calcula o tempo de espera no trajeto atual.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *TempoCarregamentoBateria*  *NumeroEntrega* | *TempoEspera* | *entrega* |

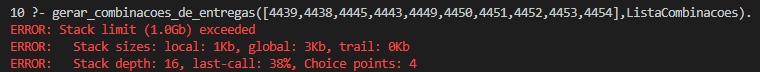
#### atualiza

Verifica e atualiza continuamento o trajeto com menor custo.

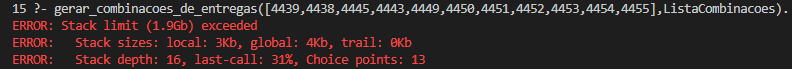
|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *LOrdemEntregas*  *LOrdemArmazens*  *Custo*  *ListaCarregamentosArmazem*  *ListaQuantidadesCarregamento*  *ListaTemposCarregamentoArmazem* |  | *custo\_min*  *retract*  *assertz* |

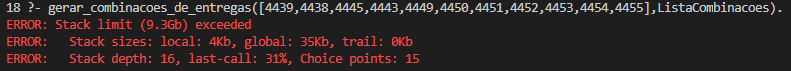
## **Análise Algoritmo**

Este algoritmo não utiliza o *findAll* pois este predicado, em conjunto com o predicado *permutation,* rapidamente esgotaria a *stack* reservada para o *prolog*.

Com 10 entregas diferentes já esgotaria 1Gb de memória.

Podemos aumentar este número para o dobro 2Gb com predicado: set\_prolog\_flag(stack\_limit, 2000000000).

Mas analisámos que com 11 entregas volta a falhar.

Colocando um número muito maior como 10Gb ele não iria ser capaz de gerar nem as respetivas 11 entregas.

Para passar este problema de memória à frente, o algoritmo usa um loop que vai gerando 1 permutação de cada vez, verificando se o seu custo é melhor ou pior. E podemos observar que encontramos um novo problema, que é o tempo necessário para gerar as mesmas 11 entregas.

O gráfico abaixo indica o tempo demorado em relação ao número de entregas.

**Figura 1:** Tempo demorado em relação ao número de entregas.

Podemos observar que o tempo demorado é fatorial, o que torna este algoritmo inviável a partir das 8 entregas.

Para resolver este problema foram criadas heurísticas para melhorar o algoritmo. Os dados foram obtidos a partir do ficheiro “analises.pl” com o predicado *testAlgoritm*, que se encontra no repositório.

# **Heurísticas**

As heurísticas são utilizadas para obter um resultado, num tempo muito mais reduzido, embora este resultado não seja sempre o melhor possível.

## Peso

Este algoritmo tem por base o peso de cada entrega, e decide entregar primeiro a mais pesada.

### Funcionamento do Algoritmo

#### heuristica\_peso

Inicia o processo de cálculo do caminho, com menor custo associado, com recurso à heurística do peso transportado pelo camião.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *ListaEntregas* | *LOrdemDeEntregas*  *LOrdemDeArmazens*  *ListaCarregamentosArmazem*  *ListaQuantidadesCarregamento*  *ListaTemposCarregamentoArmazem*  *Custo* | *ordenar\_entregas\_por\_peso*  *encontrar\_dados\_para\_entregas*  *calcula\_custo* |

#### ordenar\_entregas\_por\_peso

Ordena as entregas, de modo descendente, pelo peso a ser descarregado em cada entrega.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *ListaEntregas* | *Result* | *findall*  *entrega*  *member*  *sort*  *reverse*  *maplist* |

## Distância

Este algoritmo segue sempre para o ponto de entrega mais próximo do atual.

### Funcionamento do Algoritmo

#### heuristica\_distancia

Inicia o processo de calculo do caminho, com menor custo associado, com recurso à heurística da distância percorrida pelo camião.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *ListaEntregas* | *LOrdemDeEntregas*  *LOrdemDeArmazens*  *ListaCarregamentosArmazem*  *ListaQuantidadesCarregamento*  *ListaTemposCarregamentoArmazem*  *Custo* | *encontrar\_dados\_para\_entregas*  *armazem\_principal*  *determinar\_trajeto\_distancia*  *append*  *encontrar\_identificadores\_entregas\_para\_armazens*  *encontrar\_dados\_para\_entregas*  *calcula\_custo* |

#### determinar\_trajeto\_distancia

Determina qual o trajeto no qual se percorre a menor distância possível.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *ListaEntregas* | *Result* | *atualizar\_lista*  *proximo\_destino\_com\_menor\_distancia*  *determinar\_trajeto\_distancia* |

#### atualizar\_lista

Atualiza a lista de armazéns, aos quais ainda falta efetuar a entrega.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *ListaArmazens*  *ProximoDestino*  *NovoArmazemAtual* | *ListaArmazensAtualizadaComNovoArmazemAtual* | *delete* |

#### proximo\_destino\_com\_menor\_distancia

Determina qual o próximo armazém, para o qual a distância percorrida é menor.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *ListaOpcoesArmazens*  *Origem* | *ProximoDestino*  *MenorDistancia* | *distancia*  *proximo\_destino\_com\_menor\_distancia* |

#### encontrar\_identificadores\_entregas\_para\_armazens

Identifica quais as entregas associadas aos respetivos armazéns.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *ListaArmazens ArmazemPrincipal* | *ListaEntregas* | *delete*  *encontrar\_identificadores\_entregas\_para\_armazens1*  *reverse* |

## Produto Peso Distancia

Este algoritmo utiliza um produto entre o peso e a distância para decidir qual a próxima melhor entrega.

Funcionamento do Algoritmo

#### heuristica\_produto

Inicia o processo de calculo do caminho, com menor custo associado, com recurso à heurística da distância percorrida e do peso transportado pelo camião.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *ListaEntregas* | *LOrdemDeEntregas*  *LOrdemDeArmazens*  *ListaCarregamentosArmazem*  *ListaQuantidadesCarregamento*  *ListaTemposCarregamentoArmazem*  *Custo* | *encontrar\_dados\_para\_entregas*  *armazem\_principal*  *determinar\_trajeto\_produto*  *append*  *encontrar\_identificadores\_entregas\_para\_armazens*  *encontrar\_dados\_para\_entregas*  *calcula\_custo* |

#### determinar\_trajeto\_produto

Determina qual o trajeto no qual o produto da distância pela massa é menor.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *ListaArmazens* | *TrajetoFinal* | *proximo\_destino\_com\_menor\_produto*  *atualizar\_lista*  *determinar\_trajeto\_produto* |

#### próximo\_destino\_com\_menor\_produto

Determina qual o próximo armazém, para o qual, o produto da distância pela massa, é menor.

|  |  |  |
| --- | --- | --- |
| Parâmetros Entrada | Parâmetros Saída | Dependência de Predicados |
| *ListaOpcoesArmazens*  *Origem* | *ProximoDestino*  *MenorProduto* | *entrega*  *distancia*  *carateristicasCam*  *proximo\_destino\_com\_menor\_produto* |

# **Heurísticas x Permutações**

Os seguintes dados de comparação das heurísticas com o algoritmo de permutações, foram obtidos através do ficheiro “heuristicasTest.pl”. Não foi possível obter dados da solução ótima a partir de 12 entregas pois estas já estavam a demorar mais de 2 horas a executar.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nº Entregas | Solução Ótima | Tempo Execução | Heurística Peso | Heurística  Distancia | Heurística  Produto | Melhor Heurística |
| 5 | 340.479 | 8 ms | 486.584 | 340.479 | 340.479 | Dist/Prod |
| 6 | 412.548 | 20 ms | 613.794 | 452.290 | 452.290 | Dist/Prod |
| 7 | 479.238 | 155 ms | 722.907 | 479.238 | 479.238 | Dist/Prod |
| 8 | 505.847 | 1405 ms | 774.946 | 547.045 | 547.045 | Dist/Prod |
| 9 | 594.955 | 14052 ms | 945.577 | 628.667 | 628.667 | Dist/Prod |
| 10 | 682.188 | 158077 ms | 1023.626 | 749.453 | 749.453 | Dist/Prod |
| 11 | 730.249 | 1858332 ms | 1197.376 | 877.843 | 806.892 | Prod |
| 12 | N/a | N/a | 1494.333 | 864.807 | 864.807 | Dist/Prod |
| 13 | N/a | N/a | 1587.737 | 919.251 | 919.251 | Dist/Prod |
| 14 | N/a | N/a | 1827.774 | 975.633 | 975.633 | Dist/Prod |
| 15 | N/a | N/a | 1884.809 | 1036.002 | 1059.532 | Dist |

# **Minorantes e Majorantes**

O minorante e o majorante são usados para obter o caminho com o custo menor de tempo e o custo maior de tempo respetivamente, dado uma entrega com vários pontos intermediários.

## Minorante

Este algoritmo recebe os seguintes parâmetros de entrada:

* Orig: Armazém onde o camião inicia a viagem
* Dest: lista de todos os armazéns onde o camião tem entregas para efetuar
* C: variável para armazenar o custo total da viagem consoante o tempo

Este predicado recorre a chamada de outros predicados:

* conta: verifica o tamanho da lista Dest onde estão armazenados os destinos das entregas.
* menor: este predicado retorna o acumulado da variável C, ou seja menor custo consoante o tempo, recorrendo também a chamada de vários predicados:
  + visited: serve para verificar se o destino passado por parâmetro já foi visitado.
  + Procura: calcula o caminho de menor custo consoante o tempo e devolve o valor do custo para o predicado menor, recorre também a chamada de outro predicado:
    - calcula: calcula o custo total de tempo entre dois armazéns.

## Majorante

Este predicado e em tudo igual ao minorante a exceção da chamada do predicado maior que faz exatamente o oposto do predicado menor usado no minorante, recorrendo a chamada do predicado procura2.

Infelizmente devido a várias dificuldades da pessoa responsável por esta issue, estes dois predicados não ficaram completamente funcionais.

# **Conclusão**

Os resultados obtidos, utilizando o algoritmo de permutações, evidenciam que é sempre obtido o caminho mais eficiente. No entanto, esta eficiência está associada a duas restrições, tempo de geração de resultados e limitação por memória do hardware. Facto evidenciado quando o algoritmo é usado para um número de entregas acima de 12.

Após a recolha do resultado das Heurísticas, é possível afirmar que a Heurística de Distância e de Produto são as melhores aproximações da solução ótima, por outro lado, a Heurística de Peso afasta-se mais visto que esta não tem em conta a distância que tem elevada importância no cálculo do custo.