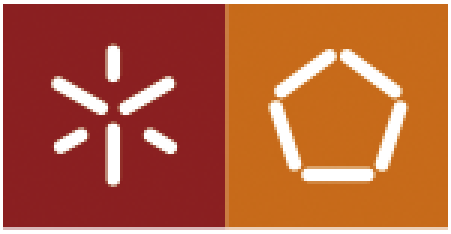

Universidade do Minho
Escola de Engenharia

Green Distribution

Inteligência Artificial

Trabalho em grupo – 1ª Fase

Renato André Machado Gomes	a84696
Sebastião Mendes de Freitas	a71074
Luís Miguel Teixeira Fernandes	a88539
Guilherme Gil Rocha Gonçalves	a88280

02 de dezembro de 2021

Introdução

No âmbito da unidade curricular de Inteligência Artificial, o grupo foi apresentado com a proposta de desenvolvimento de um sistema de representação de conhecimento e raciocínio com capacidade para caracterizar um universo de discurso na área da logística de distribuição de encomendas, entre outros objetos.

Uma vez que se vive em tempos onde a ecologia é um tópico muito relevante na nossa sociedade, o grupo vai desenvolver um projeto com a empresa Green Distribution, que tem como objetivo privilegiar sempre o meio de entrega mais ecológico.

Pré-requisitos

No início do desenvolvimento do projeto, a empresa Green Distribution declarou a existência de algumas condições previamente estabelecidas como:

- o cliente tem a possibilidade de indicar o tempo máximo em que pretende que a sua encomenda seja entregue, existindo assim prazos de entrega (imediato, 2h, 6h, 1 dia, etc);
- o cliente classifica a entrega do estafeta, num ranking entre 0 e 5 estrelas;
- a encomenda distribuída é caracterizada, pelo menos, pelo seu peso e volume.
- o preço do serviço de entrega tem em conta para além da encomenda, pelo menos, o prazo de entrega e meio de transporte utlizado.
- o estafeta que não cumpre o prazo determinado para a entrega da encomenda sofrerá uma penalização, como por exemplo, a diminuição de entregas associadas a si;
- o estafeta está associado um conjunto de entregas a efetuar em determinadas ruas;
- as ruas estão associadas a uma cidade;
- as cidades estão associadas à freguesia correspondente.

O estafeta ainda tem de ter em consideração o seu método de transporte para a realização das entregas das encomendas:

- As bicicletas podem transportar encomendas no máximo até 5 Kg, tendo em conta uma velocidade média de 10km/h;
- No caso das motos, este meio poderá transportar encomendas com um limite máximo de 20 Kg e com uma velocidade média de cerca de 35km/h;
- Relativamente ao carro, este terá uma velocidade média de aproximadamente 25km/h, com um peso máximo de transporte de 100kg.

Requisitos

O projeto a ser desenvolvido pelo grupo terá de apresentar as seguintes funcionalidades propostas:

- identificar o estafeta que utilizou mais vezes um meio de transporte mais ecológico;
- identificar que estafetas entregaram determinada(s) encomenda(s) a um determinado cliente;
- identificar os clientes servidos por um determinado estafeta;
- calcular o valor faturado pela Green Distribution num determinado dia;
- identificar quais as zonas (e.g., rua ou freguesia) com maior volume de entregas por parte da Green Distribution;
- calcular a classificação média de satisfação de cliente para um determinado estafeta;
- identificar o número total de entregas pelos diferentes meios de transporte, num determinado intervalo de tempo;
- identificar o número total de entregas pelos estafetas, num determinado intervalo de tempo;
- calcular o número de encomendas entregues e não entregues pela Green Distribution, num determinado período de tempo;
- calcular o peso total transportado por estafeta num determinado dia.

Tipos de dados

No início do trabalho, o grupo concebeu os seguintes dados com os seus respetivos componentes, para o desenvolver das queries, respeitando os pré-requisitos impostos.

Cliente:

- Identificador;
- Freguesia;

Encomenda:

- Identificador;
- Peso;
- Identificador do estafeta;
- Identificador do cliente;
- Prazo de Entrega (imediata (0), 2h, 6h, 1 dia(24h), etc);
- Data entrega;
- Transporte;
- Entregue (0 | 1);

Estafeta:

- Identificador;
- Lista de Classificações;
- Lista de Encomendas;

Transporte:

- Tipo do Veículo;
- Classificação ecológica (1 menos poluente, 3 mais poluente);
- Peso máximo;
- Velocidade média;
- Preço por Km.

Mapa

Para simular as viagens que os estafetas realizam, foi concebido um mapa de freguesias remetentes a possíveis clientes. Esta informação foi estabelecida através de um grafo de freguesias, com todas as possíveis ligações entre cada nodo. Uma vez o grafo criado, foi apenas necessário produzir capacidades de calcular possíveis caminhos, que possibilita calcular os potenciais caminhos mais curtos de um ponto inicial estático para qualquer destino a escolha, e o seu devido custo.

Predicado shortest(): o resultado deste ficheiro, através deste método é possível calcular o melhor caminho entre dois nodos estabelecidos no grafo, e finalmente receber o custo desse caminho.

```
shortest(A,B,Path,Length) :-
    setof([P,L],path(A,B,P,L),Set),
    Set = [_|_], % fail if empty
    minimal(Set,[Path,Length]).
```

Predicado connected(): estabelece se dois nodos tem conexão.

```
connected(X,Y,L) :- move(X,Y,L) ; move(Y,X,L).
```

Predicado path(): devolve um caminho possível entre dois nodos, estabelecendo o seu custo.

```
path(A,B,Path,Len) :-
        travel(A,B,[A],Q,Len),
        reverse(Q,Path).
```

Predicado Travel(): percorre o percurso entre duas moradas, se possível.

```
travel(A,B,P,[B|P],L) :-
        connected(A,B,L).
travel(A,B,Visited,Path,L) :-
        connected(A,C,D),
        C \== B,
        \+member(C,Visited),
        travel(C,B,[C|Visited],Path,L1),
        L is D+L1.
```

Predicado minimal(): dando uso ao min, percorre uma lista de possíveis caminhos, devolvendo o com o menor custo.

```
minimal([F|R],M) :- min(R,F,M).
```

Predicado min(): estabelece entre dois caminhos, qual a de menor custo.

```
min([],M,M).
min([[P,L]|R],[_,M],Min) :- L < M, !, min(R,[P,L],Min).
min([_|R],M,Min) :- min(R,M,Min).
```

Queries

Para conceber todas as capacidades remetentes aos requerimentos, foi criado um ficheiro queries.pl, onde podem ser encontrados todos os predicados necessários para produzir resultados. Nesta parte do relatório vamos olhar singularmente para cada querie requerida e analisar as soluções concebidas.

Identificar o estafeta que utilizou mais vezes um meio de transporte mais ecológico;

```
maisVerde(Est):-
  encomenda(_,_,Est,_,_,_,_,_,_,_,'bicicleta', 1).

listaVerde(L):- findall(E,maisVerde(E),L).

estafetaMaisVerde(Est,C):-
  listaVerde(L),
  conta_max_elemento(Est,C,L).
```

Para produzir esta informação, foi assumido que apenas entregas feitas através de bicicleta são consideradas, uma vez que os dados usados obrigatoriamente vão conter encomendas onde a bicicleta foi o transporte de escolha. Foram criados três predicados específicos para obter a informação requerida, onde o *conta_max_elemento*, tratasse de um predicado que vai ter uso não só nesta querie, mas o qual também vamos aprofundar aqui.

O predicado *maisVerde* filtra encomendas que foram entregues e que tenham sido efetuadas com o uso de uma bicicleta como meio de transporte. O predicado *listaverde* dá uso do *maisVerde* para percorrer todos os dados referentes a estafetas, assim é possível obter uma lista de todas as encomendas que passam pelo filtro do *maisVerde.* Para completar os resultados, foi necessário percorrer todas as entradas dado pelo *listaverde*, para no final estabelecer quais dos Estafetas é mais comum na lista, isto é apenas possível através do predicado estafetaMaisVerde, para realizar essa tarefa foi criado o *conta_max_elemento.*

```
conta_elemento(X,N,L) :-
  aggregate(count,member(X,L),N).
conta_max_elemento(X,N,L) :-
  aggregate(max(N1),X1,conta_elemento(X1,N1,L),N),
  member(X,L),
  conta_elemento(X,N,L),
  !.
```

O *conta_elemento* é um predicado que recebe uma lista e um elemento possível dessa lista, através do predicado *agregate* foi então possível estabelecer quantas vezes o elemento dado se encontra na lista. Dando uso da conta*_elemento*, foi criado o *conta_max_elemento*, que possibilita percorrer a lista e devolver o elemento mais comum da lista.

Identificar que estafetas entregaram determinada(s) encomenda(s) a um determinado cliente;

```
verEstafeta(Enc,Cli,Est):-
    encomenda(Enc,_,Est,Cli,_,_,_,_,_,_,_,1).
```

Para identificar estafetas referentes a especificas encomendas e clientes, foi criado o predicado *verEstafeta*, este é capaz de filtrar encomendas através de nomes de clientes e encomendas.

Identificar os clientes servidos por um determinado estafeta;

```
clientesServidosEstafetas(Est,Cli):-
    encomenda(_,_,Est,Cli,_,_,_,_,_,_,_,1).
```

Tal como a querie anterior, foi usada uma solução bastante próxima, com apenas a necessidade de percorrer encomendas, filtrando pelo nome do estafeta escolhido.

Calcular o valor faturado pela Green Distribution num determinado dia;

```
calculaPrecoEncomenda(Enc,C):-
  encomenda(Enc,_,_,Cli,_,_,_,_,_,_,T,1),
  cliente(Cli,Local,_),
  shortest(gualtar,Local,_,Len),
  transport(T,_,_,_,CustoKm),
  C is CustoKm * Len.

calculaPrecoEncomendaporDia(C,AA,MM,DD):-
  encomenda(Enc,_,_,_,_,_,AA,MM,DD,_,_,_,1),
  calculaPrecoEncomenda(Enc,C).

calculaPrecoTodasEncomendasporDia(C,[]):- C is 0.
calculaPrecoTodasEncomendasporDia(C,[H|T]):-
  encomenda(H,_,_,_,_,_,AA,MM,DD,_,_,_,1),
  calculaPrecoEncomendaporDia(C1,AA,MM,DD),
  calculaPrecoTodasEncomendasporDia(C2,T),
  C is C1+C2.

totalFaturadoPorDia(C,AA,MM,DD):-
  listaEncDia(L,AA,MM,DD),
  calculaPrecoTodasEncomendasporDia(C,L).

listaEncDia(L,AA,MM,DD):- findall(Enc,restringeData(Enc,AA,MM,DD),L).

restringeData(Enc,AA,MM,DD):-
  encomenda(Enc,_,_,_,_,_,AA,MM,DD,_,_,_,1).
```

Com o objetivo de obter o valor faturado num determinado dia foi necessário dividir a tarefa em duas partes. Em primeiro lugar existe a necessidade de criar uma lista com todas as encomendas feitas no dia selecionado, onde de seguida percorremos a lista de encomendas e obtemos o total faturado, isto encontrasse no predicado *totalFaturadoPorDia*.

Na primeira parte da listagem, existiu a necessidade de filtrar todas as encomendas pelo dia selecionado, isto foi possível através do predicado *restringeData*, o qual através do *findAll* é capaz de percorrer todas as encomendas e estabelecer quais foram feitas na data requerida.

Para finalizar o *calculaPrecoTodasEncomendasporDia* percorre todas as encomendas já filtradas e calcula o valor, o qual é acumulado e devolvido. O cálculo do valor foi estabelecido pelo predicado *calculaPrecoEncomenda*, este estabelece o nome do cliente através da encomenda, onde foi obtido o local, ou seja, o destino da entrega. Uma vez obtidos estes dados são calculados os caminhos mais curtos e os seus custos, onde por final calculamos o valor faturado da encomenda ao multiplicar o custo por quilometro dependente do meio de transporte em uso e o comprimento do caminho calculado previamente.

Identificar quais as zonas (e.g., rua ou freguesia) com maior volume de entregas por parte da Green Distribution;

```
calculaVolumeEncomendaLocal(Local,V):-
  listaLocal(L),
  conta_max_elemento(Local,V,L).

listaLocal(L):- findall(Local,restringeLocal(Local),L).

restringeLocal(Local):-
  encomenda(_,_,_,Cli,_,_,_,_,_,_,_,1),
  cliente(Cli,Local,_).
```

Este caso foi resolvido usando uma logica já previamente estabelecida, tornando a solução bastante trivial. Recorremos a criação de uma lista através do predicado *listaLocal*, que lista todas as encomendas filtrando apenas as que pertencem a freguesia requerida através do predicado *restringeLocal*. Uma vez obtida a lista, é apenas necessário obter o elemento mais comum, o que foi fazível através do predicado já usado anteriormente; *conta_max_elemento*.

Calcular a classificação média de satisfação de cliente para um determinado estafeta;

```
classificaEstafeta(Est,C):-
  estafeta(Est,L,_),
  media(L,C).

media( L, Media ):-
    sumlist( L, Sum ),
    length( L, Length),
    (  Length > 0
    -> Media is Sum / Length
    ;  Media is 0
    ).
```

Para calcular a classificação média foram usados dois passos. Inicialmente é obtido do determinado estafeta uma lista de classificações, de seguida é dado uso a um predicado chamado de *media* para calcular a classificação resultante de todas as classificações. O predicado *media* começa por somar todos os elementos da lista de classificações, obtém o número total de classificações e conclui o seu resultado ao dividir a acumulação de classificações pelo seu número total de classificações, ou resulta em 0 no caso em que a lista seria fazia, ou seja, o estafeta escolhido nunca foi dado uma classificação.

Identificar o número total de entregas pelos diferentes meios de transporte, num determinado intervalo de tempo;

```
listaTrans(T,AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF,L,C):-
  findall(Enc,verificaEntregasTransporteTempo(Enc,T,AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF),L), length(L,C).

verificaEntregasTransporteTempo(Enc,T,AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF):-
  encomenda(Enc,_,_,_,_,AA,MM,DD,HH,Min,T,1),
  verificaIntervaloTempo(AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF,AA,MM,DD,HH,Min, Bool),
  Bool =:= 1.

totalEntregasTrans(T,AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF,C):-
  listaTrans(T,AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF,_,C).
```

Para identificar o valor requerido, recorremos a listar todas as entregas que se encontram dentro de o determinado intervalo de tempo, onde depois se conta todos os elementos da lista de forma a obter o resultado procurado. Para filtrar todas as encomendas foi usado o predicado *verificaEntregasTransporteTempo* o qual recorre ao predicado *verificaIntervaloTempo*, o qual retorna o valor 1 caso o tempo lhe dado se encontra dentro do intervalo de tempo requerido. Através desse predicado é possível filtrar todas as encomendas e assim devolver o valor requerido

```
verificaIntervaloTempo(AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF,AA,MM,DD,HH,Min, Bool):-
  ((AA=:=AI , AA=:=AF), (MM=:=MI,MM=:=MF), (DD=:=DF,DD=:=DI), (HH=:=HI,HH=:=HF), (Min>=MinI, Min=<MinF)->  Bool is 1;
  (AA>AI , AA<AF) -> Bool is 1;
  (AA<AI , AA>AF) -> Bool is 0;
  (AA=:=AI , AA=:=AF), (MM>MI,MM<MF) ->  Bool is 1;
  (AA=:=AI ,AA=:=AF), (MM<MI;MM>MF) ->  Bool is 0;
  (AA=:=AI ,AA=:=AF), (MM=:=MI,MM<MF), (DD>DI) ->  Bool is 1;
  (AA=:=AI ,AA=:=AF), (MM=:=MI,MM<MF), (DD<DI) ->  Bool is 0;
  (AA=:=AI ,AA=:=AF), (MM=:=MI,MM<MF), (DD=:=DI), (HH<HI) ->  Bool is 0;
  (AA=:=AI ,AA=:=AF), (MM=:=MI,MM<MF), (DD=:=DI), (HH>HI) ->  Bool is 1;
  (AA=:=AI ,AA=:=AF), (MM=:=MI,MM<MF), (DD=:=DI), (HH=:=HI), (Min>=MinI) ->  Bool is 1;
  (AA=:=AI ,AA=:=AF), (MM=:=MI,MM<MF), (DD=:=DI), (HH=:=HI), (Min<MinI) ->  Bool is 0;
  (AA=:=AI ,AA=:=AF), (MM=:=MI,MM=:=MF), (DD<DI;DD>DF) ->  Bool is 0;
  (AA=:=AI ,AA=:=AF), (MM=:=MI,MM=:=MF), (DD=:=DF,DD=:=DI), (HH>HI,HH<HF) ->  Bool is 1;
  (AA=:=AI ,AA=:=AF), (MM=:=MI,MM=:=MF), (DD=:=DF), (HH<HF) ->  Bool is 1;
  (AA=:=AI ,AA=:=AF), (MM=:=MI,MM=:=MF), (DD=:=DF), (HH>HF) ->  Bool is 0;
  (AA=:=AI ,AA=:=AF), (MM=:=MI,MM=:=MF), (DD=:=DI), (HH>HI) ->  Bool is 1;
  (AA=:=AI ,AA=:=AF), (MM=:=MI,MM=:=MF), (DD=:=DI), (HH<HI) ->  Bool is 0;
  (AA=:=AI ,AA=:=AF), (MM=:=MI;MM=:=MF), (DD=:=DF), (HH=:=HF), (Min=<MinF) ->  Bool is 1;
  (AA=:=AI ,AA=:=AF), (MM=:=MI;MM=:=MF), (DD=:=DF), (HH=:=HF), (Min>MinF) ->  Bool is 0;
  (AA=:=AI ,AA=:=AF), (MM>MI,MM=:=MF), (DD<DF) ->  Bool is 1;
  (AA=:=AI ,AA=:=AF), (MM>MI,MM=:=MF), (DD>DF) ->  Bool is 0;
  (AA=:=AI ,AA=:=AF), (MM>MI,MM=:=MF), (DD=:=DF), (HH>HF) ->  Bool is 0;
  (AA=:=AI ,AA=:=AF), (MM>MI,MM=:=MF), (DD=:=DF), (HH<HF) ->  Bool is 1;
  (AA=:=AI ,AA=:=AF), (MM>MI,MM=:=MF), (DD=:=DF), (HH=:=HF), (Min=<MinF) ->  Bool is 1;
  (AA=:=AI ,AA=:=AF), (MM>MI,MM=:=MF), (DD=:=DF), (HH=:=HF), (Min>MinF) ->  Bool is 0;
  (AA>AI , AA=:=AF), (MM<MF) ->  Bool is 1;
  (AA>AI , AA=:=AF), (MM>MF) ->  Bool is 0;
  (AA>AI , AA=:=AF), (MM=:=MF), (DD<DF) ->  Bool is 1;
  (AA>AI , AA=:=AF), (MM=:=MF), (DD>DF) ->  Bool is 0;
  (AA>AI , AA=:=AF), (MM=:=MF), (DD=:=DF), (HH>HF) ->  Bool is 0;
  (AA>AI , AA=:=AF), (MM=:=MF), (DD=:=DF), (HH<HF) ->  Bool is 1;
  (AA>AI , AA=:=AF), (MM=:=MF), (DD=:=DF), (HH=:=HF), (Min=<MinF) ->  Bool is 1;
  (AA>AI , AA=:=AF), (MM=:=MF), (DD=:=DF), (HH=:=HF), (Min>MinF) ->  Bool is 0;
  (AA=:=AI , AA<AF), (MM>MI) ->  Bool is 1;
  (AA=:=AI , AA<AF), (MM<MI) ->  Bool is 0;
  (AA=:=AI , AA<AF), (MM=:=MI), (DD=:=DI), (HH>HI) ->  Bool is 1;
  (AA=:=AI ,AA<AF), (MM=:=MI), (DD=:=DI), (HH<HI) ->  Bool is 0;
  (AA=:=AI ,AA<AF), (MM=:=MI), (DD=:=DI), (HH=:=HI), (Min>=MinI) ->  Bool is 1;
  (AA=:=AI ,AA<AF), (MM=:=MI), (DD=:=DI), (HH=:=HI), (Min<MinI) ->  Bool is 0;
  (AA=:=AI ,AA<AF), (MM=:=MI), (DD>DI) ->  Bool is 1;
  Bool is 0).
```

O predicado *verificaIntervaloTempo* apesar de ser uma operação relativamente simples, tornase bastante complexa, pois existem bastantes casos possíveis. Inicialmente foi desenvolvido através de vários métodos auxiliares, mas depois de dificuldades em criar um comportamento desejável chegamos a solução demonstrada em cima. Este predicado vai ser usado mais a frente noutras queries.

Identificar o número total de entregas pelos estafetas, num determinado intervalo de tempo;

```
listaEstTempo(Est,AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF,L,C):-
  findall(Est,verificaEntregasEstafetaTempo(Est,AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF),L), length(L,C).

verificaEntregasEstafetaTempo(Est,AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF):-
  encomenda(_,_,Est,_,_,AA,MM,DD,HH,Min,_,1),
  verificaIntervaloTempo(AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF,AA,MM,DD,HH,Min, Bool),
  Bool =:= 1.

totalEntregasEstTempo(Est,AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF,C):-
  listaEstTempo(Est,AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF,_,C).
```

Para obter o este valor, foi usado um método baseado em criar uma lista, dando como resultado o tamanho. A solução recorre ao predicado *verificaEntregasEstafetaTempo* para filtrar todos as encomendas realizadas num determinado intervalo de tempo. No caso de estabelecer se as entregas se encontram dentro do intervalo requerido, foi usado o predicado *verificaIntervaloTempo* o qual já foi referido anteriormente.

Calcular o número de encomendas entregues e não entregues pela Green Distribution, num determinado período de tempo;

```
verificaEntregasRealizadasTempo(Enc,AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF):-
  encomenda(Enc,_,_,_,_,AA,MM,DD,HH,Min,_,1),
  verificaIntervaloTempo(AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF,AA,MM,DD,HH,Min, Bool),
  Bool =:= 1.

verificaEntregasNaoRealizadasTempo(Enc,AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF):-
  encomenda(Enc,_,_,_,_,AA,MM,DD,HH,Min,_,0),
  verificaIntervaloTempo(AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF,AA,MM,DD,HH,Min, Bool),
  Bool =:= 1.


procuraRealiazadas(AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF,L,C):-
  findall(Enc,verificaEntregasRealizadasTempo(Enc,AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF),L), length(L,C).

procuraNaoRealiazadas(AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF,L,C):-
  findall(Enc,verificaEntregasNaoRealizadasTempo(Enc,AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF),L), length(L,C).


totalEntreguesNEntregues(E,NE,AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF):-
procuraRealiazadas(AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF,_,E),
procuraNaoRealiazadas(AI,MI,DI,HI,MinI,AF,MF,DF,HF,MinF,_,NE).
```

Tal como muitas das soluções anteriores, esta recorre a listagem de elementos os quais são contados no final de forma a obter o resultado desejado. A grande diferença entre esta querie e o as outras que recorrem ao mesmo método encontrasse nos dois predicados *verificaEntregasRealizadaTempo* e *verificaEntregasRealizadasTempo* que distinguem extrengas realizadas e entregas não realizadas perspecticamente, sendo assim usadas como filtros auxiliares para o *procuraRealizadas* e *procuraNaoRealizadas.*

Calcular o peso total transportado por estafeta num determinado dia.

```
calculaPesoDiaEst(Est,P,AA,MM,DD):-
  listaEstDia(L,Est,AA,MM,DD),
  sumlist(L,P).

listaEstDia(L,Est,AA,MM,DD):- findall(P,restringeEstData(Est,AA,MM,DD,P),L).

restringeEstData(Est,AA,MM,DD,P):-
  encomenda(_,P,Est,_,_,AA,MM,DD,_,_,_,_).
```

Para calcular o peso total transportado por um estafeta num determinado dia, foi criado o *calculaPesoDiaEst* o qual recorre ao predicado *listaEstDia* para criar uma lista de todas as encomendas entregues pelo estafeta e dia escolhido, no final recorrendo ao *sumList* para calcular o total. O predicado *restringeEstData* estabelece se a encomenda em questão realmente foi transportada no dia escolhido, assim de forma a criar uma lista com apenas as encomendas prevalentes.

Conclusão

Ocorreram alguns problemas iniciais, especificamente no que toca a compreensão do enunciado, e bastante tempo foi perdido em soluções demasiado complexas para problemas os quais não existia qualquer razão de os resolver. Com um bocado de tentativa e erro, conseguimos finalmente solidificar a nossa perspetiva do enunciado e fazer o tratamento de dados corretamente de forma rápida.

Realizado este projeto, feita a população de dados e a compreensão do enunciado conseguimos produzir as operações necessárias para o estudo da informação relativamente rápido. Assim conseguimos perceber melhor as vantagens de usar uma língua de programação como PROLOG, com a ajuda de ferramentas externas para produzir dados, o potencial das técnicas que realizamos neste projeto tem capacidades bastante desejadas.