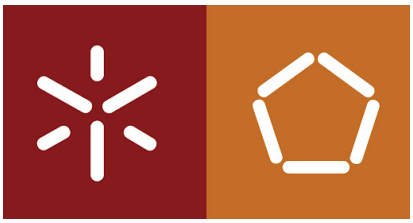

Universidade do Minho

Licenciatura em Engenharia Informática

Inteligência Artificial

Trabalho em grupo – 2ª Fase

Grupo 41

Rui Monteiro (A93179) Rodrigo Rodrigues (A93201)

Daniel Azevedo (A93324) Nuno Peixoto (A93244)

Ano Letivo 2021/2022




1 Resumo

Este documento é um relatório técnico sobre o trabalho desenvolvido durante a segunda fase do trabalho prático da Unidade Curricular Inteligência Artificial. Esta fase do trabalho tem como objetivo o uso de técnicas de formulação de problemas e a aplicação de diversas estratégias para a resolução de problema com o uso de algoritmos de procura. Todo o código aqui apresentado está na íntegra no ficheiro em anexo.

Conteúdo

2 Introdução

Neste documento apresentamos uma solução para a segunda fase do trabalho prático da Unidade Curricular de Inteligência Artificial, perante a proposta de atualizar o sistema anteriormente desenvolvido na 1ª fase do trabalho, criando um sistema de recomendação de circuitos de entrega de encomendas para o caso de estudo. Todo o código foi escrito na linguagem de programação Prolog.

3 Elaboração do Caso Prático

3.1 Cenário onde serão efetuadas as entregas

O armazém de distribuição da empresa *Green Distribution* encontra-se situado em **Gualtar** e as entregas serão efetuadas em Braga, mais especificamente no seguinte mapa de freguesias:

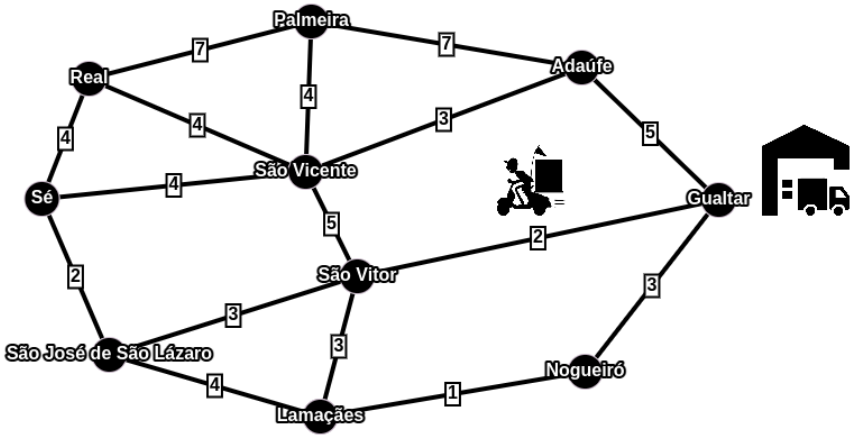

Figura 1: Grafo representativo de um mapa de freguesias de Braga

Estado Inicial: Centro de Distribuição da *Green Distribution* (Gualtar)
Pontos de Entrega: As diversas freguesias
Estado Final: Regresso ao Centro de Distribuição

3.2 Formulação do Problema como um problema de pesquisa

Tipo de problema: Estado Único

a. Conjunto de Estados: conjunto de triplos (L,K,E) em que L representa a localização(freguesia) em que o estafeta se encontra, K representa a distância percorrida pelo estafeta (km) e E representa o número de entregas pendentes.

b. Estado inicial: (Gualtar, 0, E)

c. Estado final: (Gualtar, K, 0)

d. Operadores:

- **Conduzir entre as freguesias**
  Efeitos: Alteração da localização atual do estafeta; Aumento da distância percorrida
- **Realizar entrega**
  Efeitos: Subtração de 1 unidade ao campo E (nº entregas pendentes)

e. Custo da solução: soma dos custos (distâncias) entre freguesias percorridas para realizar a entrega

3.3 Realização de uma entrega

De modo a gerar o caminho que o estafeta terá de percorrer até ponto de entrega da encomenda, foram implementadas 4 tipos diferentes de pesquisas(profundidade, largura, iterativa limitada em profundidade e A*).

O estafeta tem a opção de determinar o caminho com o auxilio de cada uma das diferentes pesquisas, para tal só necessita de indicar o identificador da encomenda que pretende realizar a entrega. Cada pesquisa irá retornar um caminho, no entanto somente a A* irá dar o melhor caminho, que é o caminho de menor distância/custo. De forma a comparar as diferentes pesquisas, foram implementadas funções para realizar uma única entrega, utilizando cada uma das diferentes pesquisas para gerar um caminho/circuito: ***entrega_encomenda_profundidade***,

entrega_encomenda_largura, ***entrega_encomenda_DFID***, ***entrega_encomenda_aStar_Dist*** (privilegia minimização de custos de distância).

3.4 Realização de múltiplas entregas

Visto que a pesquisa A* foi aquela que determinou o melhor caminho, esta foi a estratégia escolhida/usada para determinar o caminho que o estafeta tem de percorrer de forma a realizar a entrega de um conjunto de encomendas.

De modo a alcançar o objetivo, primeiramente é determinado o melhor caminho entre a sede da empresa (**Gualtar**) e o ponto de entrega da primeira encomenda da lista, seguidamente é calculado o caminho entre o primeiro e o segundo ponto de entrega e assim sucessivamente. Quando for alcançado o último ponto de entrega é calculado o caminho de retorno até à sede da empresa. No fim, também é indicado o tempo que o estafeta demorou e a distancia percorrida.

Assim, um estafeta só poderá realizar várias entregas se estas seguirem as seguintes condições:

- Todas as encomendas têm de ser entregues pelo mesmo veículo.
- A soma total do peso de todas as encomendas não pode exceder o limite máximo que o dado veículo pode transportar.

```
entrega_varias(List,C,D,Time) :- head(List,Id),
                                 findall(Vei,(entrega(_,_,_,Id,_,_,Vei,_,_)),S),
                                 head(S,V),
                                 verificaVeiculo(List,V),
                                 pesoTotal(List,Peso),
                                 verificaPeso(V,Peso),
                                 findall((Freg),(entrega(_,_,_,Id,IdC,_,_,_,_),cliente(IdC,IdF),freguesia(IdF,Freg,_,_)),Fr),
                                 head(Fr,Dest),
                                 solve_astar_Dist(gualtar,Dest,CamReverse/Dist1),
                                 reverse_list(CamReverse,R),
                                 removeCabeca(List,H),
                                 aux_entregas(H,Cam,Dist,T,V,Dest),
                                 determinaTempo(V,Dist1,Peso,T1),
                                 Time is T+T1,
                                 D is Dist+Dist1,
                                 append(R,Cam,C).
```

Figura 2: Entrega de várias encomendas utilizando o algoritmo A*

3.5 Alterações e impacto resultantes de cada estafeta poder passar a efetuar mais de uma entrega

```
?- time(entrega_varias([13,16],C,D,T)).
% 25,599 inferences, 0.008 CPU in 0.008 seconds (100% CPU, 3243329 Lips)
C = [gualtar,saovitor,saovicente,adaufe,gualtar],
D = 16,
T = 0.6666666666666667 .

?- time(entrega_encomenda_aStar(13,C,D,T)).
% 714 inferences, 0.000 CPU in 0.000 seconds (99% CPU, 2067001 Lips)
C = [gualtar,saovitor,saovicente,saovitor,gualtar],
D = 14,
T = 0.5870175438596492 .

?- time(entrega_encomenda_aStar(16,C,D,T)).
% 214 inferences, 0.000 CPU in 0.000 seconds (99% CPU, 1418882 Lips)
C = [gualtar,adaufe,gualtar],
D = 10,
T = 0.4222222222222222 .
```

Figura 3: Exemplo de entrega de 2 duas encomendas

Realizando a entrega de duas encomendas juntas podemos verificar que, ainda que o veículo saia do armazém de distribuição das com mais peso, este tipo de entrega é mais eficiente tanto a nível de distância total percorrida como de tempo.

Somando o tempo que o estafeta demoraria a entregar as duas encomendas separadamente, obteríamos aproximadamente 1 hora. Entregar as duas encomendas juntamente (ou seja, na mesma viagem) demoraria aproximadamente 0,66 horas. Relativamente à distância,

entregar as encomendas na mesma viagem implicaria percorrer 16km, que é mais eficiente, do que os 24km correspondentes a entregar as encomendas em separado (viagens individuais).

Sendo assim, podemos concluir que realizar a entrega de várias encomendas juntamente (na mesma viagem) é mais eficiente.

3.6 Base de Conhecimento

Na 2º Fase do projeto, aproveitou-se os predicados inicialmente definidos na Base de Conhecimento da 1º Fase do projeto, fazendo alterações (principalmente adição de novos predicados) à Base de Conhecimento inicial da 1º Fase do projeto.

- Alteração do predicado freguesia: *id, nome, latitude, longitude -> V,F*

```
freguesia(0,gualtar,41.57082270408355,-8.385132957697323).
freguesia(1,adaufe,41.58822016025725,-8.39769439943325).
freguesia(2,saovitor,41.55811981119368,-8.406098832097905).
freguesia(3,nogueiro,41.55064380159762,-8.388788519075284).
freguesia(4,saovicente,41.55643189968566,-8.423919635155087).
freguesia(5,palmeira,41.59702501943494,-8.434789209553655).
freguesia(6,sjsl,41.550581002256486,-8.41930439867474).
freguesia(7,lamacaes,41.546340875881725,-8.394738304278034).
freguesia(8,se,41.55367358737278,-8.428914196332162).
freguesia(9,real,41.5589672332343,-8.44476031167318).
```

Agora, a base de conhecimento conta com 9 freguesias que constituem um mapa de freguesias de Braga. Cada freguesia contém dois campos extra, latitude e longitude, que são as coordenadas reais da freguesia usadas posteriormente para cálculos de distância e estimativas de custos.

- Adição do predicado aresta: *origem, destino, distancia -> V,F*

```
aresta(gualtar,adaufe,5).
aresta(gualtar,saovitor,2).
aresta(gualtar,nogueiro,3).
aresta(adaufe,saovicente,3).
aresta(adaufe,palmeira,7).
aresta(saovitor,saovicente,5).
aresta(saovitor,sjsl,3).
aresta(saovitor,lamacaes,3).
```

```
aresta(nogueiro,lamacaes,1).
aresta(lamacaes,sjsl,3.5).
aresta(sjsl,se,2).
aresta(se,real,3).
aresta(se,saovicente,4).
aresta(real,saovicente,4).
aresta(real,palmeira,7).
aresta(palmeira,saovicente,4).
```

Arestas que constituem um grafo representativo do mapa de freguesias de Braga, ou seja, o cenário onde serão efetuadas as entregas.

```
rota(adaufe,palmeira,real).
rota(saovitor,saovicente,se).
rota(nogueiro,lamacaes,sjsl).
```

Existem agora 3 rotas de entregas principais: Norte, Centro e Sul.

4 Estratégias de procura

4.1 Procura não informada

Nesta secção encontram-se as implementações das diferentes estratégias de procura solicitadas no enunciado, feitas pelo grupo, bem como explicações acerca dos métodos usados para efetuar as mesmas.

4.1.1 Profundidade (DFS - Depth-First Search)

Esta estratégia consiste em expandir sempre um dos nós mais profundos da árvore. Uma vez que é necessário muito pouca memória, esta estratégia é vantajosa para problemas com muitas soluções. Por outro lado, uma das desvantagens é o facto de devolver a primeira solução que encontra, em vez de devolver a solução ideal (i.e com menor custo).
Complexidade no Tempo: O(b^m)
Complexidade no Espaço: O(bm)

```
resolve_pp(Nodo,Destino,R,C):-
    registaGoal(Destino),
    profundidadeprimeiro1(Nodo,[Nodo],Caminho),
    append([Nodo],Caminho,S),
    reverse_list(S,CaminhoInverso),
    removeCabeca(CaminhoInverso,C2),
    append(S,C2,R),
    calculaCusto(R,C),
    removerGoal(Destino).

profundidadeprimeiro1(Nodo,_,[]):-
    goal(Nodo).
profundidadeprimeiro1(Nodo, Historico,[ProxNodo|Caminho]):-
    adjacente(Nodo,ProxNodo,_),
    nao(membro(ProxNodo,Historico)),
    profundidadeprimeiro1(ProxNodo,[ProxNodo|Historico], Caminho).

entrega_encomenda_profundidade(Id,Cam,Dist,Time):-
    findall((Freg,(Peso,Vei)),(encomenda(Id,Peso,_),entrega(_,_,_,Id,IdC,_,Vei,_,_),cliente(IdC,IdF),freguesia(IdF,Freg,_,_)),S),
    extrai_primeiro(S,Dest),
    extrai_segundo(S,Inf),
    colocaList(Inf,Li),
    extrai_primeiro(Li,P),
    extrai_segundo(Li,V),
    resolve_pp(gualtar,Dest,Cam,Dist),
    Aux is Dist/2,
    velocida_veiculo(V,Velo),
    determinaTempo(V,Aux,P,Time1),
    Time is Time1+(Aux/Velo).
```

Figura 4: Implementação da estratégia de procura em Profundidade

4.1.2 Largura (BFS - Breadth-First Search)

Esta estratégia corresponde em expadir primeiro todos os nós de menor profundidade de um grafo, realizando uma pesquisa sistemática e exaustiva (bom apenas para pequenos problemas).
Complexidade no Tempo: $O(b^d)$
Complexidade no Espaço: $O(b^d)$

```
resolve_largura(Inicio,Destino, S3,C) :-
    registaGoal(Destino),
    bfs2(Destino,[[Inicio]],R),
    reverse_list(R,Solucao),
    removeCabeca(Solucao,S2),
    append(R,S2,S3),
    calculaCusto(S3,C),
    removerGoal(Destino).

bfs2(Dest,[[Dest|T]|_],Cam):- reverse_list([Dest|T],Cam).
bfs2(Dest,[LA|Outros],Cam):- LA=[Act|_],
findall([X|LA],
(Dest\==Act,adjacente(Act,X,_),\+member(X,LA)),Novos),
append(Outros,Novos,Todos),
bfs2(Dest,Todos,Cam).

entrega_encomenda_largura(Id,Cam,Dist,Time):-
    findall((Freg,(Peso,Vei)),(encomenda(Id,Peso,_),entrega(_,_,_,Id,IdC,_,Vei,_,_),cliente(IdC,IdF),freguesia(IdF,Freg,_,_)),S),
    extrai_primeiro(S,Dest),
    extrai_segundo(S,Inf),
    colocaList(Inf,Li),
    extrai_primeiro(Li,P),
    extrai_segundo(Li,V),
    resolve_largura(gualtar,Dest,Cam,Dist),
    Aux is Dist/2,
    velocida_veiculo(V,Velo),
    determinaTempo(V,Aux,P,Time1),
    Time is Time1+(Aux/Velo).
```

Figura 5: Implementação da estratégia de procura em Largura

4.1.3 Busca iterativa Limitada em Profundidade

Esta estratégia consiste em executar pesquisa em profundidade limitada, iterativamente, aumentando sempre o limite da profundidade.
Complexidade no Tempo: $O(b^d)$
Complexidade no Espaço: $O(b^d)$

```
solve_DFID( Node, Destino,Solucao,C) :-
    registaGoal(Destino),
    depthFirstIterativeDeepening(Node, R),
    reverse_list(R,S1),
    removeCabeca(R,S2),
    append(S1,S2,Solucao),
    calculaCusto(Solucao,C),
    removerGoal(Destino).


path(Node,Node,[Node]).
path(FirstNode,LastNode,[LastNode|Path]) :-
path(FirstNode, OneButLast,Path),
adjacente(OneButLast,LastNode,_),
 \+ member(LastNode,Path).

 depthFirstIterativeDeepening(Node, Solution) :-
    path(Node,GoalNode,Solution),
    goal(GoalNode).

entrega_encomenda_DFID(Id,Cam,Dist,Time):-
    findall((Freg,(Peso,Vei)),(encomenda(Id,Peso,_),entrega(_,_,_,Id,IdC,_,Vei,_,_),cliente(IdC,IdF),freguesia(IdF,Freg,_,_)),S),
    extrai_primeiro(S,Dest),
    extrai_segundo(S,Inf),
    colocaList(Inf,Li),
    extrai_primeiro(Li,P),
    extrai_segundo(Li,V),
    solve_DFID(gualtar,Dest,Cam,Dist),
    Aux is Dist/2,
    velocida_veiculo(V,Velo),
    determinaTempo(V,Aux,P,Time1),
    Time is Time1+(Aux/Velo).
```

Figura 6: Implementação da estratégia de procura iterativa limitada em profundidade

4.2 Procura informada

4.2.1 A* (A estrela)

Esta estratégia de pesquisa informada consiste em evitar expandir caminhos "caros". É uma combinação da *Pesquisa Gulosa* com a *Uniforme*.

Visto que este tipo de procura necessita da utilização de uma estimativa, de forma a realizar uma avaliação do custo para chegar ao objetivo(ponto de entrega), no caso da procura A* cujo objetivo é minimizar custos de distância(**aStar_Dist**), são usadas as coordenadas geográficas, em que é calculada a distância, em linha reta, do nó que o estafeta se encontra até ao ponto de entrega.

```
solve_astar_Dist(Node,Dest ,Path/Cost) :-
    registaGoal(Dest),
    distancia_entre_freguesias(Node,Dest,Estimate),
    astar_Dist(Dest,[[Node]/0/Estimate], RevPath/Cost1/_),
    Cost is 2*Cost1,
    reverse_list(RevPath, Path1),
    removeCabeca(RevPath,S2),
    append(Path1,S2,Path),
    removerGoal(Dest).

astar_Dist(_,Paths, Path) :-
    get_best_Dist(Paths, Path),
    Path = [Node|_]/_/_,
    goal(Node).
astar_Dist(Dest,Paths, SolutionPath) :-
    get_best_Dist(Paths, BestPath),
    seleciona(BestPath, Paths, OtherPaths),
    expand_astar_Dist(Dest,BestPath, ExpPaths),
    append(OtherPaths, ExpPaths, NewPaths),
    astar_Dist(Dest,NewPaths, SolutionPath).

get_best_Dist([Path], Path) :- !.
get_best_Dist([Path1/Cost1/Est1,_/Cost2/Est2|Paths], BestPath) :-
Cost1 + Est1 =< Cost2 + Est2, !,
get_best_Dist([Path1/Cost1/Est1|Paths], BestPath).
get_best_Dist([_|Paths], BestPath) :-
get_best_Dist(Paths, BestPath).

expand_astar_Dist(Dest,Path, ExpPaths) :-
    findall(NewPath, move_astar_Dist(Dest,Path,NewPath), ExpPaths).

move_astar_Dist(Dest,[Node|Path]/Cost/_, [NextNode,Node|Path]/NewCost/Est) :-
    adjacente(Node, NextNode, StepCost),
    \+ member(NextNode, Path),
    NewCost is Cost + StepCost,
    distancia_entre_freguesias(NextNode,Dest,Est).

entrega_encomenda_aStar_Dist(Id,Cam,Dist):-
    findall((Freg),(entrega(_,_,_,Id,IdC,_,_,_,_),cliente(IdC,IdF),freguesia(IdF,Freg,_,_)),S),
    head(S,Dest),
    solve_astar_Dist(gualtar,Dest,Cam/Dist).
```

Figura 7: Implementação da estratégia de procura A*

5 Resultados

Análise comparativa entre as diferentes estratégias de procura implementadas, em termos de: tempos de execução, utilização de memória e se encontra a melhor solução.

Estratégia	**Tempo execução(s)**	**Espaço**	**(Custo, Tempo)**	**Encontrou a melhor solução?**
DFS	0.000	O(bm)	(36, 4.07)	Não
BFS	0.000	O(b^d)	(24, 2.718)	Não
BIL em Profundidade	0.000	O(bd)	(24, 2.718)	Não
A *	0.000	O(b^d)	(14, 1.586)	Sim

Tabela 1: Análise comparativa entre as diferentes estratégias de procura

(Custo, Tempo): Custo da solução (km), Tempo que demora a percorrer o circuito (h)

BIL: busca iterativa limitada

b: fator de ramificação (número máximo de sucessores de um nó) de uma árvore de pesquisa

d: profundidade da melhor solução

m: máxima profundidade do espaço de estados

Os resultados foram obtidos simulando entregas, com cada estratégia de pesquisa, para o mesmo destino. Foram implementadas as funções ***entrega_encomenda_*** <***estratégia_pesquisa***> que determinam os circuitos e simulam as entregas das encomendas, utilizando a estratégia de pesquisa indicada. Exemplo:
entrega_encomenda_profundidade encontra a solução de caminho para o local de entrega através da estratégia de pesquisa não informada ***Depth-First Search***.

```
?- time(entrega_encomenda_profundidade(9,Cam,Dist,Time)).
% 163 inferences, 0.000 CPU in 0.000 seconds (99% CPU, 986683 Lips)
Cam = [gualtar,adaufe,saovicente,saovitor,sjsl,se,sjsl,saovitor,saovicente,adaufe,gualtar],
Dist = 36,
Time = 4.078481012658227 .
```

Figura 8: Entrega de encomenda com pesquisa em profundidade

```
?- time(entrega_encomenda_largura(9,Cam,Dist,Time)).
% 642 inferences, 0.000 CPU in 0.000 seconds (99% CPU, 2896119 Lips)
Cam = [gualtar,adaufe,saovicente,se,saovicente,adaufe,gualtar],
Dist = 24,
Time = 2.7189873417721517 .
```

Figura 9: Entrega de encomenda com pesquisa em largura

```
?- time(entrega_encomenda_DFID(9,Cam,Dist,Time)).
% 189 inferences, 0.000 CPU in 0.000 seconds (96% CPU, 1795946 Lips)
Cam = [gualtar,adaufe,saovicente,se,saovicente,adaufe,gualtar],
Dist = 24,
Time = 2.7189873417721517 .
```

Figura 10: Entrega de encomenda com pesquisa iterativa limitada em profundidade

```
?- time(entrega_encomenda_aStar_Dist(9,Cam,Dist)).
% 705 inferences, 0.000 CPU in 0.000 seconds (99% CPU, 2011493 Lips)
Cam = [gualtar,saovitor,sjsl,se,sjsl,saovitor,gualtar],
Dist = 14 .
```

Figura 11: Entrega de encomenda com pesquisa A* (A estrela) - custo (km)

5.1 Análise de Resultados

Analisando os exemplos anteriores, onde se demonstra o resultado de realizar uma entrega entre Gualtar e Sé, podemos observar que só o algoritmo a* estrela obteve a solução ideal, ou seja aquela que tem menor custo.

A nível de tempo de execução, todas as estratégias apresentaram o mesmo resultado (execução quase instantânea). No entanto, pode-se observar diferenças na utilização de memória entre as estratégias de pesquisa.

A estratégia de procura em profundidade dá como resultado o primeiro caminho que foi encontrado mas, no entanto podemos verificar que o seu número de inferências é o menor, ou seja, a utilização de CPU e memória é menor comparada com as outras pesquisas.

No que diz respeito às pesquisas em largura podemos verificar que a pesquisa em largura e a pesquisa em largura iterativa limitada devolvem a mesma solução, no entanto a segunda usa menos recursos computacionais.

Em relação ao algoritmo a* este descobre a melhor solução do problema, no entanto como temos mais inferências, durante a sua execução, este algoritmo é o que necessita de mais CPU e memória.

6 Funcionalidades do Programa

O programa é capaz de realizar as seguintes funcionalidades:

1. Gerar os circuitos de entrega, caso existam, que cubram um determinado território (freguesia);
2. Representação dos diversos pontos de entrega em forma de grafo, tendo em conta que apenas se devem ter localizações (freguesia) disponíveis;
3. Identificar quais os circuitos com maior número de entregas (por volume e peso);
4. Comparar circuitos de entrega tendo em conta os indicadores de produtividade;
5. Escolher o circuito mais rápido (usando o critério da distância);
6. Escolher o circuito mais ecológico (usando um critério de tempo);

6.1 Funcionalidade 1: Gerar os circuitos de entrega, caso existam, que cubram um determinado território (e.g. rua ou freguesia);

Parar gerar os circuitos de entrega, que cubram um determinado território implementamos vários funções que realizam esta tarefa com base em diferentes estratégias de pesquisa para determinar um caminho até ao local de entrega. Entre estas estão: pesquisa em profundidade, pesquisa em largura, pesquisa iterativa limitada em profundidade, e algoritmo A* (a estrela). No entanto, por análise dos resultados obtidos, o algoritmo A* mostrou os melhores resultados (circuitos de entrega com os menores custos de distância), pelo que é esta a estratégia de pesquisa principal para gerar os circuitos de entrega. Os circuitos de entrega gerados foram divididos em 3 secções: Região Norte, Região Centro e Região Sul.

6.2 Funcionalidade 2: Identificar quais os circuitos com maior número de entregas (por volume e peso);

Os circuitos de entrega encontram-se divididos em 3 secções: Região Norte, Região Centro e Região Sul. O programa é capaz de identificar quais os circuitos com o maior número de entregas. Neste caso, o programa determina corretamente que a região centro é onde existe maior número de entregas.

```
?- circuito_maior_numero_entregas(R).
R = Circuito Centro .
```

Figura 12: Resultado da funcionalidade "circuito com maior número de entregas"@

6.3 Funcionalidade 3: Comparar circuitos de entrega tendo em conta os indicadores de produtividade;

Com o objetivo de comparar a eficiência de um circuito, após este ser calculado, com o auxílio de uma das pesquisas, são utilizados os indicadores de produtividade, ou seja, tempo e distancia. Nas funções ***entrega_encomenda_*** <***estratégia_pesquisa***>(ex: entrega_encomenda_-

profundidade) que geram os circuitos e simulam as entregas das encomendas, são utilizados os indicadores de produtividade (ex: através da invocação do predicado *determinaTempo*). Invocando duas estratégias de pesquisa diferentes para realizar entregas, serão gerados

caminhos diferentes, e com os indicadores de produtividade pode-se comparar os circuitos/caminhos.

6.4 Funcionalidade 4: Escolher o circuito mais rápido (usando o critério da distância);

O circuito de entrega mais rápido, ou seja, com menor distância percorrida, é gerado pelo algoritmo A* através deste é indicado o melhor caminho. O circuito mais rápido pode ser obtido através da função **entrega_encomenda_aStar**, que estima os custos utilizando coordenadas geográficas das freguesias.

6.5 Funcionalidade 5: Escolher o circuito mais ecológico (usando um critério de tempo);

O circuito que demora menos tempo é aquele que percorre menos distância, sendo assim, este é obtido com o auxilio da pesquisa A*. Pode-se obter este circuito através da função **entrega_encomenda_aStar**.

7 Comentários Finais e Conclusão

Tendo terminado o desenvolvimento do projeto, o grupo pode constatar que este projeto se tornou um desafio, no sentido de compreender os conteúdos lecionados durante as aulas da unidade curricular e saber aplicá-los num contexto prático, recorrendo à linguagem de programação Prolog como a ferramenta de implementação.

Finalmente, face ao projeto desenvolvido, o grupo encara-o como sendo um sucesso, uma vez que todos os requisitos foram alcançados com a devida correção e desempenho do programa, avaliando assim seu desempenho de maneira positiva.