Relatório de ALGAV

Turma 3DG

Grupo 41:

Mário António Afonso Cardoso 1201496

Miguel Corrêa Macedo 1201199

João Pedro Miranda Figueiredo 1201197

Ivo Oliveira 1190679

André Teixeira 1190384

Representação do conhecimento de Domínio

Nós usamos os seguintes dados como base de conhecimento:

```
%idArmazem(<local>,<codigo>)
idArmazem('Arouca',1).
idArmazem('Espinho',2).
idArmazem('Gondomar',3).
idArmazem('Maia',4).
idArmazem('Matosinhos',5).
idArmazem('Oliveira de Azemeis',6).
idArmazem('Paredes',7).
idArmazem('Porto',8).
idArmazem('Povoa de Varzim',9).
idArmazem('Santa Maria da Feira',10).
idArmazem('Santo Tirso',11).
idArmazem('Sao Joao da Madeira',12).
idArmazem('Trofa',13).
idArmazem('Vale de Cambra',14).
idArmazem('Valongo',15).
idArmazem('Vila do Conde',16).
idArmazem('Vila Nova de Gaia',17).
```

Figura 2-Armazéns

```
entrega(4439, 20221205, 200, 1, 8, 10).
entrega(4438, 20221205, 150, 9, 7, 9).
entrega(4445, 20221205, 100, 3, 5, 7).
entrega(4443, 20221205, 120, 8, 6, 8).
entrega(4449, 20221205, 300, 11, 15, 20).
entrega(4398, 20221205, 310, 17, 16, 20).
entrega(4432, 20221205, 270, 14, 14, 18).
entrega(4437, 20221205, 180, 12, 9, 11).
entrega(4451, 20221205, 220, 6, 9, 12).
entrega(4452, 20221205, 390, 13, 21, 26).
entrega(4444, 20221205, 380, 2, 20, 25).
entrega(4455, 20221205, 280, 7, 14, 19).
entrega(4399, 20221205, 260, 15, 13, 18).
entrega(4454, 20221205, 350, 10, 18, 22).
entrega(4446, 20221205, 260, 4, 14, 17).
entrega(4456, 20221205, 330, 16, 17, 21).
```

Figura 1-Entregas

```
carateristicasCam(eTruck01,7500,4300,80,100,60).
```

Figura 3-Camião

```
dadosCam_t_e_ta(eTruck01,1,2,122,42,0).
dadosCam_t_e_ta(eTruck01,1,3,122,46,0).
dadosCam_t_e_ta(eTruck01,1,4,151,54,25).
dadosCam_t_e_ta(eTruck01,1,5,147,52,25).
dadosCam_t_e_ta(eTruck01,1,6,74,24,0).
dadosCam_t_e_ta(eTruck01,1,7,116,35,0).
dadosCam_t_e_ta(eTruck01,1,8,141,46,0).
dadosCam_t_e_ta(eTruck01,1,9,185,74,53).
dadosCam_t_e_ta(eTruck01,1,10,97,30,0).
dadosCam_t_e_ta(eTruck01,1,11,164,64,40).
dadosCam_t_e_ta(eTruck01,1,12,76,23,0).
dadosCam_t_e_ta(eTruck01,1,13,174,66,45).
dadosCam_t_e_ta(eTruck01,1,14,59,18,0).
dadosCam_t_e_ta(eTruck01,1,15,132,51,24).
dadosCam_t_e_ta(eTruck01,1,16,181,68,45).
dadosCam_t_e_ta(eTruck01,1,17,128,45,0).
```

Figura 4-Dados Camião no percurso entre Armazéns

Nota: A figura 4 contém apenas um excerto (apenas de quando o camião parte do armazém para outros) da totalidade.

ALGORITMO GENETICO

1)Criação da população inicial do algoritmo Genético (AG)

```
gera(X,NG,DP,PC,PM):-
        inicializa(NG,DP,PC,PM),
        gera_populacao(Pop),
        avalia_populacao(Pop,PopAv),
        ordena_populacao(PopAv,PopOrd),
        geracoes(NG), % num geracoes
        gera_geracao(0,NG,PopOrd,L),!,getBestRoute(L,X).
```

A primeira população é gerada separadamente do resto das mesmas

Nos fazemos a inicialização com os parâmetros (numero gerações, dimensão da população, probabilidade de cruzamento, e probabilidade de mutação).

Estes parâmetros são utilizados no inicializa que vai dar assert destes valores em predicados dinâmicos que vão ser usados dentro dos passos de gerar população avaliar e ordenar. Esta população depois de ordenada será utilizada para o resto das novas populações.

2)Para a primeira população e de forma a gerar diversidade genética superior. O algoritmo da heurística deveria ser utilizado uma heurística que geraria um individuo que iriamos adicionar a nossa população, mas por alguns erros não, foi adicionado. O algoritmo desenvolvido anteriormente devolve uma lista de armazéns como percurso contando com o de Matosinhos no início e fim então não foi possível implementá-lo no genético.

Tendo em conta que não conseguimos adicionar continuamos normalmente com o número de indivíduos que tínhamos

3) Aleatoriedade no cruzamento entre indivíduos da população

Para garantir que os pontos de cruzamento são feitos de forma aleatória e não em pares ordenados da sequência ordenada é aplicada uma permutação sobre população de forma aleatória de modo. Do mesmo modo sao gerados os pontos de cruzamento tendo um por base um fator aleatório, quê varia entre 1 e o número de entregas máximo de modo a escolher um da lista total

```
gerar_pontos_cruzamento1(P1,P2):-
        entregas(N),
        NTemp is N+1,
        random(1,NTemp,P11),
        random(1,NTemp,P21),
        P11\==P21,!,
        ((P11<P21,!,P1=P11,P2=P21);(P1=P21,P2=P11)).
```

4)Seleção da nova geração

Para selecionar indivíduos que devem continuar a próxima geração criamos um predicado para selecionar os melhores indivíduos. Inicialmente fazemos uma avaliação dos indivíduos constituintes da população atual, asseguir ordenamos segundo a avaliação e mais a seguir selecionamos os melhores. Num entanto este método é método é puramente elitista, pelo que concluímos que seria melhor da oportunidade a indivíduos com pior avaliação a passar a geração seguinte. Mesmo não estando entre os melhores é dada oportunidade.

```
selectMelhor(Pop,NPop,Res):-
        union(Pop,NPop,PopNPopAv),% fa
        ordena_populacao(PopNPopAv,UnionListOrd),
        length(UnionListOrd,T),
        length(Pop,InitPop),
        Limit1 is round(0.3*InitPop),
        sublist(UnionListOrd,0, Limit1,PElem),
        Limit2 is Limit1 + 1,
        sublist(UnionListOrd,Limit2,T,TPElem),
        t_p_eval(TPElem,Eval),
        ordena_populacao(Eval,EvalOrd),
        AppendQuantity is InitPop  - Limit1,
        aplica_ordenacao(EvalOrd,TPElem, TPElemOrd),
        sublist(TPElemOrd,0,AppendQuantity,BestTP),
        append(PElem,BestTP,Res).
```

```
% AVALIACAO

calcula_tempo(LC,Final):-
        nome_camiao(LC,Nome),
        bateria_camiao(Nome,CarM,TempR),
        peso_camiao(Nome,PesoC),
        tempo_mais_recarga(LC,CarM,TempR,RecargaC,PesoC,Nome,Custo,ExtraC),
        tempo_entregas(LC,TempoE),
        somar_tudo(RecargaC,ExtraC,TempoE,Custo,Final).
```

5)Analise de eficácia/eficiência

O algoritmo genético revela-se muito mais eficaz no sentido em que cumpre o seu objetivo muito mais rápido que outras opções. Permite obter uma solução com um valor aceitável, mesmo sendo um pouco menos eficiente do que uma heurística uma vez que não temos um parâmetro de estabilização.

6)Parametrização da condição de termino AG

Atualmente a única condição que foi considerada na aplicação é o número de gerações fornecido no algoritmo. Uma vez que este é atingido vai nos devolver o melhor individuo na última geração. Isto só não é ideal pois o algoritmo pode estabilizar antes da geração final.

```
inicializa(NG,DP,P1,P2):-
        (retract(geracoes(_));true), asserta(geracoes(NG)),
        (retract(populacao(_));true), asserta(populacao(DP)),
        PC is P1/100,
        (retract(prob_cruzamento(_));true),asserta(prob_cruzamento(PC)),
        PM is P2/100,
        (retract(prob_mutacao(_));true), asserta(prob_mutacao(PM)).
```

7)Uso do algoritmo genético para lidar com vários camiões

Para lidar com vários camiões é necessário fazer uma distribuição das encomendas eficaz. Para isso é essencial utilizar o output do algoritmo genéticos com base para a adaptação. Uma vez obtido o algoritmo genético, faz uma verificação dos camiões disponíveis para a responsabilidade, e acumula-se encomendas até o peso exceder o limite

```
camiaoEntregas(B,D,F,G,LF):-retractall(planeamento(_,_)),
listaCamiao(C),
obterTamanhoLista(C,TAMANHOC),
%listaEntregas(E),
gera(V*_,B,D,F,G),
armazensParaEntregas(V,E),
obterTamanhoLista(E,TAMANHOE),
obterResto(TAMANHOE,TAMANHOC,RESTO),
atribuirResto(RESTO,C,E,ListaNovaEntrega),
recursaoCamiao(C,ListaNovaEntrega),
findall(planeamento(X,A),
planeamento(X,A),LF).

obterTamanhoLista([],0).
obterTamanhoLista([_|T],R):-obterTamanhoLista(T,Tamanho),R is Tamanho+1.

obterResto(E,C,RESTO):-RESTO is (E mod C).

%armazensParaEntregas([],[]).
%armazensParaEntregas([H|T],R):-entrega(X,_,_,H,_,_),append(X,R1,R),armazensParaEntregas(T,R1).
%
armazensParaEntregas(StoreList, ResultList) :-
    armazensParaEntregas(StoreList, [], ResultList).

armazensParaEntregas([], Acc, Acc).
armazensParaEntregas([H|T], Acc, ResultList) :-
    entrega(X, _, _, H, _, _),
```

Distribuição de entregas pelos camiões

Para distribuir as entregas pelos camiões disponíveis, vamos recorrer ao método aconselhado pelos slides de apoio ao trabalho, na qual recebemos a lista de entregas já ordenada vindo do Algoritmo genético, e atribuir cada entrega por um camião, então vamos adiquirir a lista de camiões, o seu tamanho, e as entregas vindas do AG, que na realidade retorna armazéns, e vamos descobrir por associação a quais entregas se refere.

```
listaCamiao(C),
obterTamanhoLista(C,TAMANHOC),
%listaEntregas(E),
gera(V*_,B,D,F,G),
armazensParaEntregas(V,E),
```

No final da primeira iteração, em que cada camião já tem uma entrega, inverto a lista de entregas de modo que o camião que recebeu a entrega de menor custo, receberá

agora a de maior custo, o segundo camião a segunda de maior e assim sucessivamente, de modo a manter o equilíbrio dos custos pelos camiões.

Isto apenas seria possível caso o número de camiões fosse múltiplo do número de entregas. Quando não acontece, supondo 3 camiões para 8 entregas, necessitamos tirar as entregas em "excesso", antes de começar a recursividade, para tal fazemos "tamanhoEntregas % tamanhoCamioes" e obtemos o resto, depois iteramos até ao zero, atribuindo essas entregas, aos primeiros camiões, usando o predicado dinâmico planeamento/2, usando assertz/2.

```
obterTamanhoLista(E,TAMANHOE),
obterResto(TAMANHOE,TAMANHOC,RESTO),
atribuirResto(RESTO,C,E,ListaNovaEntrega),
```

```
atribuirResto(0,X,T,T):-!.
atribuirResto(Resto,[HC|TC],[H|T],ListaNovaEntrega):-assertz(planeamento(HC,H)),
Resto1 is Resto-1,atribuirResto(Resto1,TC,T,ListaNovaEntrega)
.
```

Visto que este passo teria de ser, em caso de não múltiplos, executado em primeiro lugar, depois passamos para a fase de atribuição de entregas aos camiões, sendo que já contamos com uma lista a ser múltiplo da outra.

```
recursaoCamiao(C,ListaNovaEntrega),
```

Executamos uma recursividade dupla, que para cada camião percorre a lista de entregas que sobra dando assertz, e no final da primeira, inverte a lista que resta das encomendas de modo a garantir o equilíbrio, visto que volta a repetir este processo mas desta vez com a lista invertida.

```
atribuirCamiao([],T,T):-!.
atribuirCamiao([HC|TC],[HE |TE],ENOVA):-asserta(planeamento(HC,HE)),
atribuirCamiao(TC,TE,ENOVA).

recursaoCamiao(_,[]):-!.
recursaoCamiao(C,E):-atribuirCamiao(C,E,ENOVA2),inverter(ENOVA2,ENOVA),recursaoCam
iao(C,ENOVA).
```

Por último, depois de já ter acabado, executo um findall/3 que arranjo todos os planeamentos/2 para uma lista final.

```
findall(planeamento(X,A),
planeamento(X,A),LF).
```

```
?- camiaoEntregas(10,10,50,50,M).
M = [planeamento(eTruck03, 4437), planeamento(eTruck02, 4432), planeamento(eTruck01, 4439),
 planeamento(eTruck03, 4445), planeamento(eTruck02, 4449), planeamento(eTruck01, 4438), pla
neamento(eTruck01, 4398), planeamento(eTruck02, 4443)].
```

Assim podemos garantir a boa distribuição, tome-se como exemplo: Existem 9 entregas, de 1 a 9 a nível de "custo", e 3 camiões de C1 a C3.
C1→1,6,7 total = 14

C2→2,5,8 total = 15

C3→3,4,9 total = 16

Desta forma asseguramos sempre uma boa distribuição das entregas.

Adição/Cancelamento/Edição de uma Entrega

```
acoesEntregas(EntregaID, Lista, Acao, B, D, F, G) :-
  gera(V*_, B, D, F, G),
  armazensParaEntregas(V,E),
  (   Acao = 'Add'
  -> (member(EntregaID,E) -> write('The delivery you pretend to add is already in the list.') ; (not(member(EntregaID,E))) -> Lista = [EntregaID|E])
  ;   Acao = 'Delete'
  -> ((member(EntregaID,E) -> delete(E, EntregaID, Lista)) ; (not(member(EntregaID,E))) -> write('The delivery is not in the list.'))
  ;   write('Please Insert a valid action')
  ).
```

Esta User Storie tinha como objetivo poder editar, eliminar e adicionar uma entrega à lista que irá ser usada no algoritmo genético.

No entanto devido à falta de tempo e incompatibilidade com a maneira que fazemos o algoritmo genético (não recebe nenhuma lista), implementamos apenas as ações de adicionar e eliminar de uma lista com os Ids das entregas com verificações para ver se a ação que queremos efetuar é válida no momento.

- ## Conclusão

  A partir das user stories desta cadeira que foram feitas recorrendo a matéria disponibilizada no Moodle e lecionada nas aulas, aprofundamos e pusemos em prática o nosso conhecimento sobre algoritmos genéticos,