Título - Relatório ALGAV

Subtítulo – Sprint B

Turma DE _ Grupo 26
1181628 _ André Novo
1180782 _ Diogo Ribeiro
1170500 _ Hugo Frias
1161255 _ João Cruz
1180730 _ Vera Pinto

Professor
Jorge Coelho, JMN

Unidade Curricular
Algoritmia Avançada (ALGAV)

Data: Dezembro/2020

1. Introdução

Neste sprint B, adaptamos todos os algoritmos pedidos (gerador de todas as soluções sem findall, A* e Best First) e fizemos os estudos da viabilidade dos algoritmos.

2. Representação do conhecimento do domínio

no(nome, nome curto, ponto de rendição, estação de recolha, latitude, longitude).
linha(nome, número, percurso, duração, distância).
horario(número da linha, horário daquele percurso).
horarioMinimizado(número da linha, lista de listas de horários).
ligaAux(número da linha, horario inicial pretendido).

3. Estudo da viabilidade e complexidade dos geradores de todas as soluções (com e sem findall)

Nº linhas	Nº nos	Nº pontos de rendição e estações de recolha	Caminho Testado (Noi -> Nof)	Nº Soluções	Tempo de resposta da solução com findall (s)	Tempo de resposta da solução sem findall (s)
16	15	4	'ESTPA','CRIST'	4	0,0	0,0
16	15	10	'ESTPA','CRIST'	5026	0,372	0,338
18	17	12	'ESTPA','CRIST'	26533	1,638	1,402
20	19	14	'ESTPA','CRIST'	107478	8,584	7,5301
22	20	15	'ESTPA','CRIST'	-	ERROR: Stack limit (1.0Gb) exceeded	115,809
21	20	15	'ESTPA','CRIST'	275088	28,914	24,692

• O nº de soluções dependerá só do nº de nós?

Não, também depende das linhas, porque aumenta o número de mudanças possíveis entre linhas aumentando o número de soluções.

• Acha que a topologia da rede (ao nível dos nós de rendição e estações de recolha) tem impacto?

Tem impacto. Quantos mais nós nas diferentes linhas, mais soluções serão geradas que por sua vez vai aumentar o tempo.

• Qual complexidade acha que tem o problema que estudou (notação O)?

Não fizemos a complexidade.

4. Adaptação do gerador de todas as soluções (sem findall) à minimização do horário de chegada ao destino pretendido

```
:-dynamic ligaAux/2.
:-dynamic nSOl/1.

gera_ligacoes_horario(Hin,Noi,Nof):-
    retractall(ligaAux(_,_)),
    asserta(ligaAux(0,1000000)),
    asserta(nSol(0)),
    findall(_,(
           linha(_,NL,CL,_,_),
           ordem_membros(Noi,Nof,CL),
           nth1(PosNo,CL,Noi),
           horarioMinimizado(NL,H),
           verificaHorario(Hin,H,PosNo,Hor),
           retract(nSol(NS)),
           NS1 is NS+1,
           asserta(nSol(NS1)),
           ligaAux(_,Aux),
           Hor<Aux,
           retractall(ligaAux(_,_)),
           asserta(ligaAux(NL,Hor))),
        _).

verificaHorario(_,[],_,1000000).
verificaHorario(Hi,[X|_],PosNo,Hor):-
    nth1(PosNo,X,E),
    E > Hi,
    Hor is E.
verificaHorario(Hi,[_|H],PosNo,Hor):-
    verificaHorario(Hi,H,PosNo,Hor).
```

```
:- dynamic horarioMinimizado/2.

mycompare(<,[A1|_],[A2|_]) :- A1 < A2.
mycompare(>, _, _).

gera_horarios:-
        retractall(horarioMinimizado(_,_)),
        findall(L,horario(L,_),LL),
        sort(LL,LLL),
        !, gera_aux(LLL).

gera_aux([]).
gera_aux([X|LL]):-
        findall(T,horario(X,T),ListaHorariosT),
        predsort(mycompare, ListaHorariosT, ListaOrdenada),
        assertz(horarioMinimizado(X,ListaOrdenada),_),
        !, gera_aux(LL).

ordem_membros(No1,No2,[No1|L]):-
    member(No2,L),!.
ordem_membros(No1,No2,[_|L]):-
    ordem_membros(No1,No2,L).
```

Explicação:

- **LigaAux:**

O facto dinâmico *ligaAux* irá alojar o número da linha no 1º elemento e o instante de tempo em que o motorista deve ingressar no Nó pretendido.

O predicado começa por limpar toda a base de conhecimento dinâmica identificada por *ligaAux* .

Posteriormente, cria um facto dinâmico (*ligaAux*) com o valor de 1.000.000, valor este impossível de aparecer num horário.

- **FindAll:**

Começamos por procurar uma qualquer linha e o seu caminho na base de conhecimentos: *(linha(_,NL,CL,_,_)* .

NL - Número da linha,

CL - caminho com os ShortNames (Key de um Nó) de todos os Nós que formam essa linha.

Verificamos se o Nó inicial passado por parâmetro se encontra antes do Nó final (também este passado por parâmetro) nessa mesma linha: *ordem_membros(Noi,Nof,CL)*.

Noi - Nó inicial passado por parâmetro,

Nof - Nó final passado por parâmetro,

CL - Caminho da linha obtido anteriormente através do facto linha.

De seguida, verificamos a posição do Nó inicial nesse caminho (ou seja, o seu index na linha): *nth1(PosNo,CL,Noi)* .

Posteriormente, recorrendo a outro facto dinâmico na base de conhecimento (*horarioMinimizado*), usamos o número da linha anteriormente encontrado para chegar à lista de listas dos horários dessa mesma linha: *horarioMinimizado(NL,H)* .

NL - Número da Linha,

H - Lista com todas as listas de horários possíveis dessa mesma linha.

Consequentemente, verificamos se existe algum horário dessa linha com Index igual ao index do Nó inicial obtido anteriormente (*PosNo*) e com valor estritamente superior ao valor do instante de tempo introduzido por parâmetro: *verificaHorario(Hin,H,PosNo,Hor)* .

Hin - Horário inicial passado por parâmetro, H - Lista com todas as listas de horários,

PosNo - Posição do Nó na linha que está a ser analisada neste período de execução,

Hor - O valor do horário que respeite as condições em cima mencionadas. Caso nenhum horário da linha em questão se apresente "válido", o valor retornado em **Hor** será de 1.000.000.

No instante seguinte, reavemos o atual número de soluções e logo após removemos o mesmo: *retract(nSol(NS))* .

Considerando isso, adicionamos uma unidade ao número de soluções prévio: *NS1 is NS+1* .

E neste ponto de número de soluções, terminamos criando um novo facto dinâmico que irá sustentar o novo número de soluções possíveis: *asserta(nSol(NS1))* .

Em seguida, encontramos o horário auxiliar responsável por ser o menor horário até então encontrado: *ligaAux(_,Aux)* .

Aux - o menor horário encontrado até então.

Verificamos se o novo horário encontrado é menor que o horário auxiliar atual: *Hor<Aux*

Caso seja, apagamos o facto dinâmico *ligAux* responsável por guardar a melhor solução: *retractall(ligaAux(_,_)* .

Finalmente, criamos um novo facto dinâmico usando novamente o *ligaAux* que manterá a melhor solução até então: *asserta(ligaAux(NL,Hor))* .

NL - Número da Linha com a melhor solução até ao momento,

Hor - o menor horário encontrado até então.

5. Adaptação do A* para minimização do horário de chegada ao destino pretendido

```
aStar(Orig,Dest,Cam,HorarioInicial,Custo):-
    asserta(nSol(0)),
    retractall(mm(_,_)),
    asserta(mm(Orig,HorarioInicial)),
    aStar2(Dest,[(_,0,[Orig])],Cam,Custo),!.

aStar2(Dest,[(_,Custo,[Dest|T])|_],Cam,Custo):-
    reverse([Dest|T],Cam),
    nSol(NumeroSolucoes),
    write('Número de Soluções: '),write(NumeroSolucoes),write('.'),
    write('O caminho apresentado apenas considera os pontos de rendição/recolha por onde pass
    retractall(mm(_,_)).

aStar2(Dest,[(_,Ca,LA)|Outros],Cam,Custo):-
    write(LA),
    LA=[Act|_],
    mm(Act,HorarioInicial),
    findall((CEX,CaX,[X|LA]),(
    Dest\==Act,
    liga(Act,X,NL),
    linha(_,NL,CaminhoDaLinha,_,_),
    nth1(PosicaoDoAct,CaminhoDaLinha,Act),
    horarioMinimizado(NL,HorariosDaLinha),
    verificaHorario(HorarioInicial,HorariosDaLinha,PosicaoDoAct,HorarioMaisProximo),
    HorarioMaisProximo<100000,
    nth1(PosicaoDoX,CaminhoDaLinha,X),
    verificaHorario(HorarioMaisProximo,HorariosDaLinha,PosicaoDoX,HorarioXPretendido),
    HorarioXPretendido<100000,
    CustoX is HorarioXPretendido - HorarioInicial,
    \+ member(X,LA),
    CaX is CustoX + Ca, estimativa(X, Dest, NL, Estimativa),
    CEX is CaX + Estimativa,
    asserta(mm(X,HorarioXPretendido)),
    retract(nSol(NS)),
    NS1 is NS+1,
    asserta(nSol(NS1))), Novos),
    append(Outros,Novos,Todos),
    write('Novos='),write(Novos),nl,
    sort(Todos,TodosOrd),
    write('TodosOrd='),write(TodosOrd),nl,
    aStar2(Dest,TodosOrd,Cam,Custo).
```

```
estimativa(N,N,_,0):-!.
estimativa(NoAtual,NoDestino, NumeroLinha, Estimativa):-
    no(_,NoAtual,_,_,X1,Y1),
    no(_,NoDestino,_,_,X2,Y2),
    DistanciaEstimativa is sqrt((X1-X2)^2 + (Y1-Y2)^2),
    linha(_,NumeroLinha,_,TempoLinha,DistanciaLinha),
    TempoLinhaSegundos is TempoLinha*60,
    VelocidadeMedia is DistanciaLinha/TempoLinhaSegundos,
    Estimativa is DistanciaEstimativa/VelocidadeMedia.
```

6. Adaptação do Best First com heurística de minimização da distância ao destino e considerando os horários sobre a solução encontrada

```
bestfs(Orig,Dest,Cam,HorarioInicial,Custo):-
    asserta(nSol(0)),
    retractall(mm(_,_)),
    asserta(mm(Orig,HorarioInicial)),
    bestfs2(Dest,(0,[Orig]),Cam,Custo),!.

bestfs2(Dest,(Custo,[Dest|T]),Cam,Custo):-
    !, reverse([Dest|T],Cam),
    nSol(NumeroSolucoes),
    write('Número de Soluções: '),write(NumeroSolucoes),write('.'),
    write('O caminho apresentado apenas considera os pontos de rendição/recolha por onde pass
    retractall(mm(_,_)).

bestfs2(Dest,(Ca,LA),Cam,Custo):-
    LA=[Act|_],
    mm(Act,HorarioInicial),
    findall((EstX,CaX,[X|LA]),(
    liga(Act,X,NL),
    linha(_,NL,CaminhoDaLinha,_,_),
    nth1(PosicaoDoAct,CaminhoDaLinha,Act),
    horarioMinimizado(NL,HorariosDaLinha),
    verificaHorario(HorarioInicial,HorariosDaLinha,PosicaoDoAct,HorarioMaisProximo),
    HorarioMaisProximo<100000,
    nth1(PosicaoDoX,CaminhoDaLinha,X),
    verificaHorario(HorarioMaisProximo,HorariosDaLinha,PosicaoDoX,HorarioXPretendido),
    HorarioXPretendido<100000,
    CustoX is HorarioXPretendido - HorarioInicial,
    \+member(X,LA),
    estimativa(X,Dest,NL,EstX),
    CaX is Ca+CustoX,
    asserta(mm(X,HorarioXPretendido)),
    retract(nSol(NS)),
    NS1 is NS+1,
    asserta(nSol(NS1)))
    ,Novos),
    sort(Novos,NovosOrd),
    NovosOrd = [(_,CM,Melhor)|_],
    bestfs2(Dest,(CM,Melhor),Cam,Custo).
```

Explicação:

Em ambos os algoritmos (A* e BFS), o nosso grupo alterou a forma de calcular a estimava para considerar o tempo. Basicamente, através da linha que o algoritmo em questão estava a analisar naquele momento, íamos buscar o tempo da linha e a sua distância, convertemos o tempo (em minutos) para segundos e calculávamos a velocidade média da linha. Posteriormente, com essa velocidade média e com uma distância linear entre os dois pontos, calculávamos o tempo que demoraria para chegarmos ao segundo ponto.

Distância Linear: DistanciaEstimativa is sqrt((X1-X2)^2 + (Y1-Y2)^2) .
Velocidade Média: TempoLinhaSegundos is TempoLinha*60, VelocidadeMedia is DistanciaLinha/TempoLinhaSegundos .
Estimava Temporal: Estimativa is DistanciaEstimativa/VelocidadeMedia .

Aliada à alteração do cálculo da estimava, também alterámos o cálculo do custo entre dois pontos usando o tempo. Começámos por ir buscar o número da linha (**NL**) e o caminho dessa mesma linha (**CaminhoDaLinha**), em seguida determinámos qual a posição do Nó atual (aquele que o algoritmo estava a analisar no momento - **Act**) nesse caminho e procurámos o próximo horário válido para que o motorista/passageiro conseguisse entrar nessa paragem, através de todos os horários conhecidos para essa linha (**HorariosDaLinha**) no predicado *verificaHorario* .

Posteriormente, fazemos o mesmo para o próximo nó que tem ligação com **Act** e é ponto de rendição/recolha através do predicado *liga* . Finalmente calculamos a diferença entre esses 2 horários e atribuímos o valor a *CustoX*.

Código:

```
liga(Act,X,NL),
linha(_,NL,CaminhoDaLinha,_,_),
nth1(PosicaoDoAct,CaminhoDaLinha,Act),
horarioMinimizado(NL,HorariosDaLinha),
verificaHorario(HorarioInicial,HorariosDaLinha,PosicaoDoAct,
HorarioMaisProximo),
HorarioMaisProximo<100000,
nth1(PosicaoDoX,CaminhoDaLinha,X),
verificaHorario(HorarioMaisProximo,HorariosDaLinha,PosicaoDoX,
HorarioXPretendido),
HorarioXPretendido<100000,
CustoX is HorarioXPretendido - HorarioInicial.
```

7. Estudo de viabilidade e complexidade e da melhor solução obtida considerando as implementações (Gerador de todas as soluções, A* e Best First)

		Algoritmo sem findall		Algoritmo A*			
Nº pontos de rendição e estações de recolha	Caminho Testado (Noi -> Nof)	Nº Soluções	Tempo de resposta	Nº Soluções	Tempo de resposta	Custo	Solução encontrada
12	SOBRO','CETE'	4	0.00099	418	0.00799	4780	SOBRO, CRIST, CETE
12	PARAD','PARED'	60	0.001	3903	0.038	3040	PARAD,BALTR,PARED
12	MOURZ, LORDL	5	0.0	5966	0.1399	3520	MOURZ, CRIST,LORDL
12	BALTR,PARED	19	0.0	246	0.00299	880	BALTR,PARED

Algoritmo Best First			
Nº Soluções	Tempo de resposta	Custo	Solução encontrada
26	0.001	4780	SOBRO', 'CETE'
3608	0.044	6820	PARAD,PARED
178	0.0029	28420	MOURZ, LORDL
244	0.00299	880	BALTR,PARED

8. Conclusões

Após este Sprint B, o grupo conclui que apesar da dificuldade acrescida da linguagem de programação Prolog, existem certos problemas que ficam muito mais fáceis de entender e resolver usando Prolog.

Sobre os algoritmos adaptados e estudados, concluímos que apesar do BSF (Best First) seja extremamente mais rápido que o A*, nem sempre apresenta a melhor solução. O A* considera mais fatores, então acaba por demorar mais, mas encontra a melhor solução.

9. Divisão de tarefas entre elementos do grupo

Como existiam alguns membros do grupo com maior dificuldade, decidimos que iriamos todos participar na adaptação de todos os algoritmos. Deste modo, um de nós estava a fazer partilha de ecrã enquanto adaptávamos o algoritmo em conjunto.