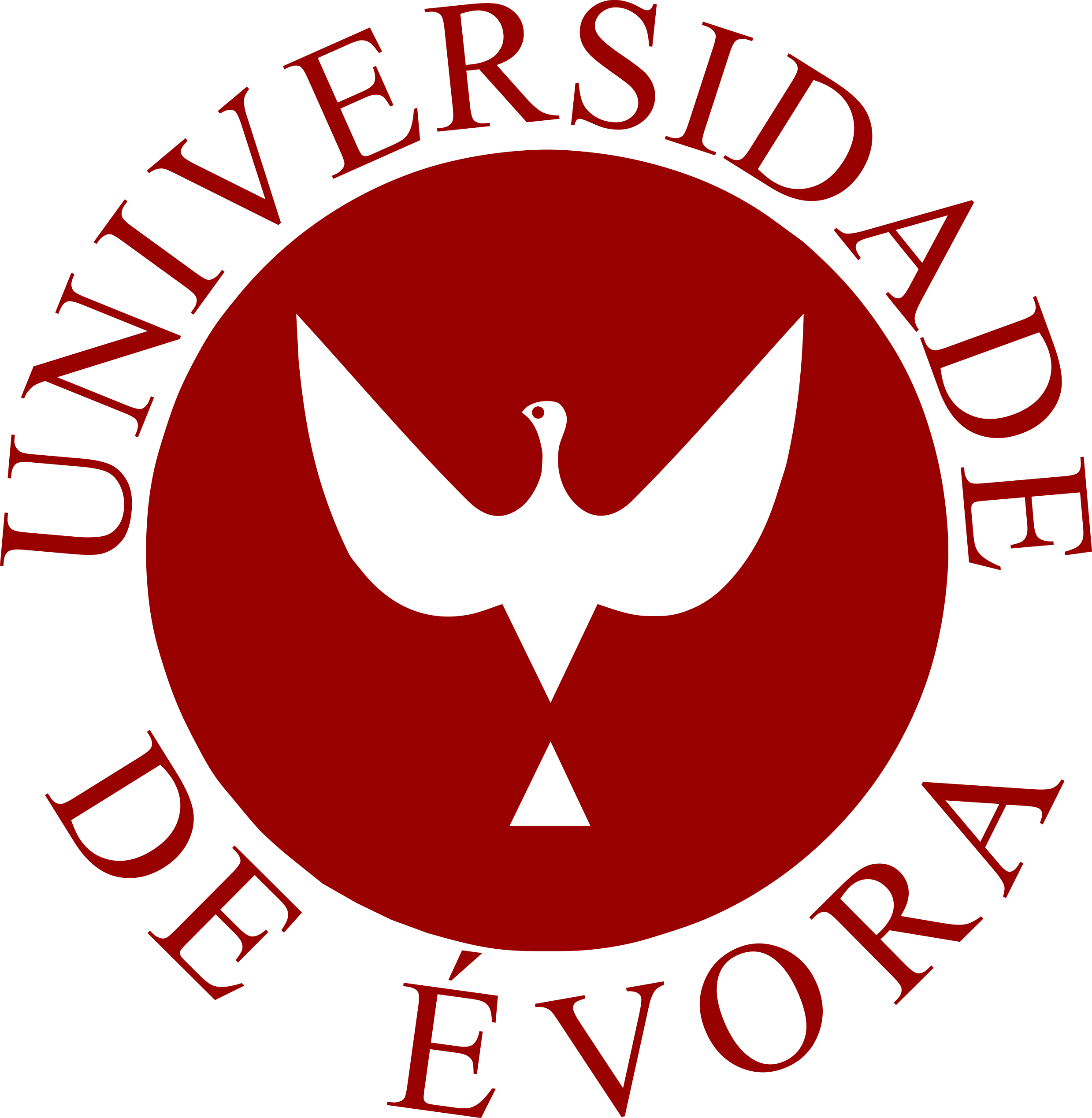
Inteligência Artificial

Trabalho 1

Resolução de problemas de pesquisa em espaço de estados



Discentes:

João Santos nº 29634

André Gouveia nº 26918

Évora 2015

Índice:

[Introdução. 1](#_Toc415940145)

[1 – Respostas. 1](#_Toc415940146)

[1.(a) 1](#_Toc415940147)

[1.(b) 2](#_Toc415940148)

[1.(c) 3](#_Toc415940149)

[1.(d) 4](#_Toc415940150)

[1.(e) 5](#_Toc415940151)

[1.(f) 7](#_Toc415940152)

Introdução.

Neste trabalho foi-nos pedido que encarasse-mos o problema da travessia condicionada de passageiros de um margem para a margem oposta de um rio como um problema de pesquisa em espaços de estados. A condição imposta depende do peso que a barca suporta. É dado o peso de cada passageiro e indicado um peso máximo para a barca de 140Kg.

1 – Respostas.

1.(a)

Para a encararmos como espaço de estados vai ser necessário definir os estados e os operadores de transição.

Os estados são definidos como um tuplo com duas listas, cada uma referente a uma das margens (podia ser efectuado apenas com umas lista, no entanto preferimos fazê-lo desta forma de forma a melhor poder acompanhar as transições de estado).

O estado inicial é definido com a margem de partida cheia e a margem de chegada vazia.

estado\_inicial([28, 65, 45, 57, 98, 120],[]).

O estado final é definido por ter a margem de partida vazia, ou seja, já todos os passageiros foram encaminhados.

estado\_final([],\_).

Para os operadores das transições definimos a remoção de todas as combinações possíveis , mas não repetidas, de passageiros na margem de partida cujo peso combinado não passa os 140Kg. Essas combinações passam da lista de margem de partida para a lista de margem de chegada.

1.(b)

O código de pesquisa que resolve este problema devolvendo sempre a melhor solução é um código de pesquisa em largura. Para a resolução criada utilizamos o código fornecido pela docente, adicionando um contador para o numero de nós visitados.

Código para pesquisa não informada:

:-dynamic(conta/1).

conta(0).

incC(\_):-

conta(X),

retractall(conta(\_)),

Y is X+1,

NovoC =.. [conta,Y],

asserta(NovoC).

pesquisa(Problema,Alg):-

consult(Problema),

estado\_inicial(S0),

pesquisa(Alg,[no(S0,[],[],0,0)],Solucao),

escreve\_seq\_solucao(Solucao).

pesquisa(largura,Ln,Sol):- pesquisa\_largura(Ln,Sol).

pesquisa\_largura([no(E,Pai,Op,C,P)|\_],no(E,Pai,Op,C,P)):- estado\_final(E), incC(\_).

pesquisa\_largura([E|R],Sol):- expande(E,Lseg), esc(E),

insere\_fim(Lseg,R,Resto),

incC(\_),

pesquisa\_largura(Resto,Sol).

expande(no(E,Pai,Op,C,P),L):- findall(no(En,no(E,Pai,Op,C,P),Opn,Cnn,P1),

(op(E,Opn,En,Cn),P1 is P+1, Cnn is Cn+C),L).

insere\_fim([],L,L).

insere\_fim(L,[],L).

insere\_fim(R,[A|S],[A|L]):- insere\_fim(R,S,L).

escreve\_seq\_solucao(no(E,Pai,Op,Custo,Prof)):- write(custo(Custo)),nl,

write(profundidade(Prof)),nl,

conta(X),

write(nos(X)),

retractall(conta(\_)),nl,

escreve\_seq\_accoes(no(E,Pai,Op,\_,\_)).

escreve\_seq\_accoes([]).

escreve\_seq\_accoes(no(E,Pai,Op,\_,\_)):- escreve\_seq\_accoes(Pai),

write(e(Op,E)),nl.

esc(A):- write(A), nl.

1.(c)

Uma vez que se trata de um problema com uma expansão explosiva, ou seja, a memória e análise necessária para a sua resolução são demasiado elevadas, indicaremos a resposta para apenas um problema com 6 passageiros.

Numero de nós visitados: 4804

Profundidade: 4

Para o algoritmo de pesquisa em largura o numero máximo de nós em memória é dado por

n = b^(d+1)

Sendo b o máximo factor de ramificação (todas as combinações possiveis e não repetidas de passageiros na margem inicial e cujo o peso combinado não exceda os 140Kg ) e d a profundidade da solução de menor custo.

Solução:

e([], ([28,65,45,57,98,120],[]))

e(transfere, ([65,57,98,120],[[28,45]]))

e(transfere, ([98,120],[[57,65],[28,45]]))

e(transfere, ([120],[[98],[57,65],[28,45]]))

e(transfere, ([],[[120],[98],[57,65],[28,45]]))

1.(d)

A 1ª heurística que utilizamos é baseada no numero de viagens necessárias para efectuar a viagem. Ao removermos um determinado de grupo, é verificado o peso combinado que sobra na margem de partida e dividido esse valor pela tara da barca. Desta forma obtemos o numero de viagens que falta efectuar removendo esse grupo de passageiros. A opção recai sempre pelo caminho em que menos viagens sobre.

A 2ª heurística é semelhante mas em vez do numero de viagens para levar o peso total, utiliza o a relação entre o peso médio e a tara da barca.

Código para as heurísticas:

pesototal([],0):-!.

pesototal([X|Ls],V1):-

pesototal(Ls,V),

V1 is V+X.

numerodepessoas([],V,V):-!.

numerodepessoas([\_|L],X,V):-

X1 is X+1,

numerodepessoas(L,X1,V).

h1((L,\_),V):- pesototal(L,T), V is T/140.

h2((L,\_),V):- numerodepessoas(L,0,N),pesototal(L,T), T\=0,!, V is T/(N\*140).

h2((L,\_),V):- V is 0.

1.(e)

Tal como na alinea 1.b, utilizamos o algoritmo de pesquisa informada dado pela docente, incluindo apenas o contador para o numero de nós visitados.

Código para a pesquisa informada:

:-dynamic(conta/1).

conta(0).

incC(\_):-

conta(X),

retractall(conta(\_)),

Y is X+1,

NovoC =.. [conta,Y],

asserta(NovoC).

pesquisai(Problema,Alg):-

consult(Problema),

estado\_inicial(S0),

pesquisa(Alg,[no(S0,[],[],0,1,0)],Solucao),

escreve\_seq\_solucaoi(Solucao).

pesquisa(a,E,S):- pesquisa\_a(E,S).

pesquisa\_a([no(E,Pai,Op,C,HC,P)|\_],no(E,Pai,Op,C,HC,P)):- estado\_final(E), incC(\_).

pesquisa\_a([E|R],Sol):- expande\_a(E,Lseg), incC(\_), esc(E),

insere\_ord(Lseg,R,Resto),

pesquisa\_a(Resto,Sol).

expande\_a(no(E,Pai,Op,C,HC,P),L):- findall(no(En,

no(E,Pai,Op,C,HC,P),Opn,Cnn,HCnn,P1),

(op(E,Opn,En,Cn),P1 is P+1, Cnn is Cn+C, h1(En,H),

HCnn is Cnn+H), L).

insere\_ord([],L,L).

insere\_ord([A|L],L1,L2):- insereE\_ord(A,L1,L3), insere\_ord(L,L3,L2).

insereE\_ord(A,[],[A]).

insereE\_ord(A,[A1|L],[A,A1|L]):- menor\_no(A,A1),!.

insereE\_ord(A,[A1|L], [A1|R]):- insereE\_ord(A,L,R).

menor\_no(no(\_,\_,\_,\_,N,\_), no(\_,\_,\_,\_,N1,\_)):- N < N1.

escreve\_seq\_solucaoi(no(E,Pai,Op,Custo,\_HC,Prof)):- write(custo(Custo)),nl,

write(profundidade(Prof)),nl,

conta(X), write(nos(X)),nl,retractall(conta(\_)),

escreve\_seq\_accoesi(no(E,Pai,Op,\_,\_,\_)).

escreve\_seq\_accoesi([]).

escreve\_seq\_accoesi(no(E,Pai,Op,\_,\_,\_)):- escreve\_seq\_accoesi(Pai),

write(e(Op,E)),nl.

esc(A):- write(A), nl.

1.(f)

Heuristica 1 :

Numero de nós visitados: 943

Profundidade: 4

Solução:

e([], ([28,65,45,57,98,120],[]))

e(transfere, ([65,45,57,120],[[28,98]]))

e(transfere, ([45,120],[[57,65],[28,98]]))

e(transfere, ([45],[[120],[57,65],[28,98]]))

e(transfere, ([],[[45],[120],[57,65],[28,98]]))

Heurística 2 :

Numero de nós visitados: 2373

Profundidade: 4

Solução:

e([], ([28,65,45,57,98,120],[]))

e(transfere, ([28,65,45,57,98],[[120]]))

e(transfere, ([28,65,45,57],[[98],[120]]))

e(transfere, ([28,45],[[57,65],[98],[120]]))

e(transfere, ([],[[28,45],[57,65],[98],[120]]))

Para a pesquisa informada, o numero máximo de nós em memória é inferior ao da pesquisa em largura uma vez que a pesquisa é direcionada e não guardamos os nós associados a expansão de caminhos que não percorremos.

# Instruções:

Para resolver o problema basta colocar ambos os ficheiros .pl na mesma pasta e na linha de comandos inserir:

swipl – s p.pl

Caso seja pretendida a pesquisa em largura, inserir:

pesquisa(‘pee.pl’,largura).

Caso seja pretendida a pesquisa informada, inserir:

pesquisai(‘pee.pl’,a).

Para adicionar/remover passageiros, basta alterar a lista de margem de partida no estado inicial.

Ex: estado\_inicial([28, 65, 45, 57, 98],[]). -> removemos o passageiro de 120Kg.

Para alternar, na pesquisa informada, a heurística a utilizar, basta alternar o ficheiro p.pl na função:

expande\_a(no(E,Pai,Op,C,HC,P),L):- findall(no(En,

no(E,Pai,Op,C,HC,P),Opn,Cnn,HCnn,P1),

(op(E,Opn,En,Cn),P1 is P+1, Cnn is Cn+C, h1(En,H),

Alterar para de h1 para h2 para mudar de heuristica

HCnn is Cnn+H), L).