Instituto Superior de Engenharia do Porto

Licenciatura em Engenharia Informática

3DI

Relatório de Algoritmia Avançada

Grupo 051

Por Beatriz Seixas (1190424),

Jéssica Alves (1190682)

Pedro Santos (1190967),

Tiago Costa (1191460)

Porto, janeiro de 2022

Índice geral

1. INTRODUÇÃO

Na sequência do projeto referente à unidade curricular de *ALGAV* foi proposto o desenvolvimento de várias *user stories* com fim a criar uma aplicação cuja missão é fornecer aplicações de manipulação e visualização de grafos de redes sociais.

Neste documento iremos apresentar uma breve explicação do problema, assim como o pensamento para cada *user story*, e exemplos demonstrativos.

As competências requisitadas para este sprint, foram as seguintes:

- Criação de uma rede à parte com os utilizadores que podem ser alcançados até N ligações a partir de um dado utilizador Sugestão de grupos
- Adaptação do A* ao problema da determinação do caminho mais forte (máximo de N ligações)Estado de arte- Lingua Natural
- Estimativa Implementada
- Adaptação do Best First ao problema da determanação do caminho mais forte (máximo de N ligações)
- Adaptação do Primeiro em Profundidade para gerar a melhor solução (já implementado no Sprint anterior) para o máximo de N ligações
- Comparação dos 3 métodos com vários exemplos, comparando tempos de geração da solução e valor da solução gerada Implementação da função multicritério que contemple forças de ligação e diferença entre likes e dislikes
- Adaptação dos 3 métodos (Primeiro em Profundidade, Best First e A*) para considerar a função multicritério do ponto anterior
- Comparação dos 3 métodos com vários exemplos e usando a função multicritério

2. CRIAÇÃO DE UMA REDE À PARTE COM OS UTILIZADORES QUE PODEM SER ALCANÇADOS ATÉ N LIGAÇÕES A PARTIR DE UM DADO UTILIZADOR

2.1. Funcionalidade desenvolvida por Todos os Elementos do Grupo

Para este sprint, foi criada uma base de conhecimento parcial onde podem ser alcançados os jogadores até N ligação de um jogador atual. Nesta rede, o jogador 1 tem dois amigos: o jogador 2 e o 3, por sua vez estes tem 2 e 1 amigos respetivamente, com a particularidade do jogador 5 ser amigo de ambos o jogador 2 e do 3. Assim temos uma plataforma melhor para a execução destes algoritmos e resultados esperados.

```
% Knowledge Base - Standalone Mode

node(1,john,[a,b,c]).
node(2,jane,[e,f,g]).
node(3,steve,[h,i,j]).
node(4,james,[a,b,g]).
node(5,carol,[a,b,c]).
node(6,michelle,[e,f,g]).



connection(2,1,10,8,50,0).
connection(2,4,2,6,100,12).
connection(2,5,19,-19,12,19).
connection(2,6,4,0,25,24).
connection(1,3,11,3,35,90).
connection(3,5,12,-2,10,31).

occ(1, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5).
occ(2, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5).
occ(3, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5).
occ(4, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5).
occ(5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5).
occ(6, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5).


hope(1, 4).
fear(1, 6).
fear(1, 4).


synonym('c#', csharp).
```

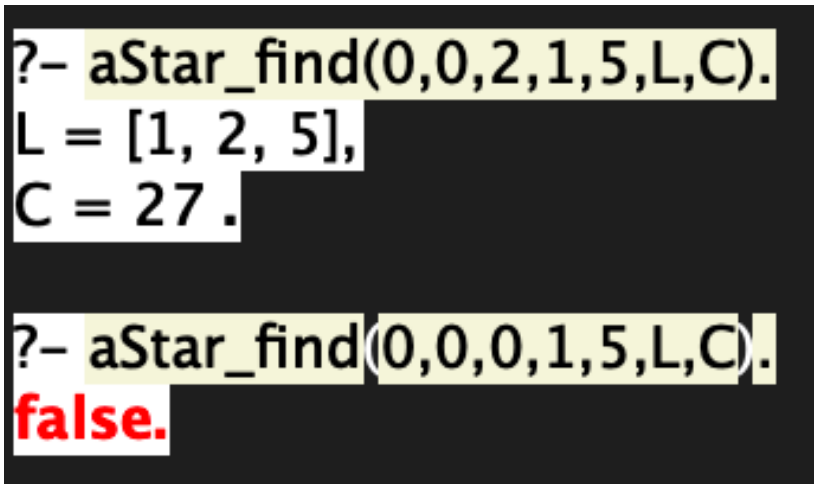
3. ADAPTAÇÃO DO A* AO PROBLEMA DA DETERMINAÇÃO DO CAMINHO MAIS FORTE (MÁXIMO DE N LIGAÇÕES)

3.1. Funcionalidade desenvolvida por Pedro Santos (1190967)

Nesta funcionalidade, o predicado chamado irá ser o *aStar_find*/7. Os parâmetros para este predicado serão *Mode*, que pode tomar um valor de 0 quando apenas considera a força de ligação ou o valor de 1, quando é executado o modo multicritério, tendo em conta a força de ligação e relação, implementada no predicado *getMulticriteria*/3); o segundo parâmetro neste método será *EmocionalBool*, um valor que indica se o estado emocional é considerado ou não, não estando este valor no âmbito do sprint C); *Threshold, representa* o número máximo de ligações para limitar o algoritmo; *Orig, representando o* id do jogador de origem; *Dest*, o id do jogador de destino; *Path*, será o caminho resultante que será retornado pela função desenvolvida neste requisito; por fim, *Cost* representa o custo do caminho retornado. Ao predicado fornecido foi adicionado a condição de paragem baseada no número máximo de ligações fornecido por parâmetro. Este método tem em conta uma estimativa calculada no predicado *aStar_estimate*/3, sendo este predicado abordado no ponto seguinte do presente relatório. Para limitar o uso do algoritmo para N ligações foi utilizado um contador que irá identificar o nível atual, tendo como condição de paragem quando este contador M for maior ou igual ao nível máximo fornecido pelo utilizador N.

```
aStar_find(Mode, EmotionBool, Threshold, Orig, Dest, Path, Cost):-
    (retract(aStar_orderedList(_));true),
    aStar_getStrengthListByPlayer(Mode, Threshold, Orig, StrengthList),
    asserta(aStar_orderedList(StrengthList)),
    aStar_aux(Mode, EmotionBool, 0, Threshold, Dest,[(_,0,[Orig])],Path,Cost).
aStar_aux(_, _, M, N, Dest,[(_,Cost,[Dest|T])|_],Path,Cost):- M >= N,reverse([Dest|T],Path).
aStar_aux(0, EmotionBool, M, N, Dest,[(_,Ca,LA)|Others],Path,Cost):-
    LA=[Act|_],
    findall((CEX,CaX,[X|LA]),
    (Dest\==Act,
    (connection(Act,X,CostX, _, _, _);
    connection(X, Act, _, CostX, _, _)),
    emotion_checkSameEmotion(EmotionBool, Act, X),
    \+ member(X,LA),
    CaX is CostX + Ca,
    aStar_estimate(N,M,EstX),
    CEX is CaX + EstX),
    New),
    append(Others,New,All),
    sort(All,AllOrd),
    M1 is M + 1,
    aStar_aux(0, EmotionBool, M1, N, Dest,AllOrd,Path,Cost).
```

Um exemplo da utilização desta funcionalidade seria o cálculo do caminho do jogador 1 da base de conhecimento parcial para o utilizador com o id 5. Neste exemplo, iremos fornecer o modo apenas com forças de ligação (valor 0), ignorar o estado emocional pois essa funcionalidade é referente ao *sprint* seguinte, fornecer o nível máximo de 2 ligações, do jogador 1 para o 5. Com estes parâmetros, obtemos o caminho [1,2,5] que seria o caminho esperado, com um custo de 27. Numa segunda execução, fornecemos o nível máximo de zero e o predicado, como esperado, não encontra um caminho com um máximo de ligações desse valor.



4. ESTIMATIVA IMPLEMENTADA

4.1. Funcionalidade desenvolvida por Pedro Santos (1190967)

No cálculo de caminhos utilizando o algoritmo A-Star, é utilizado um cálculo da estimativa do custo para ir desse nó até a solução, estando portanto sujeita a erro. Sendo desejável uma estimativa que seja um majorante o ideal é tê-la tão próxima quanto o valor real que iremos ter por um dado caminho da árvore, para isso, nesta implementação, em vez de utilizar um valor fixo no cálculo da estimativa, são utilizados valores da rede do utilizador. Para isso, antes da execução do A-Star é chamado o predicado *aStar_getStrengthListByPlayer*/4. Neste método, inicialmente é obitda a lista de todas as forças de ligação unidirecionais presentes na rede do utilizador, sendo esta lista ordenada decrescentemente e armazenada no facto dinâmico *aStar_orderedList*/1. Na obtenção desta lista, o método irá percorrer todos níveis (até N ligações), percorrer cada jogador desse nível, cada conexão desse jogador nesse nível, construindo através do predicado auxiliar *listUnion*/3 uma lista com todos os valores das forças de ligação das conexões unidirecionais na rede do utilizador em questão até N ligações. Seguidamente, depois da obtencção desta lista, esta é ordenada decrescentemente, utilizando o prodicado *sort*/4.

```
aStar_getStrengthListByFriendsList(_, _, [], []):-!.
aStar_getStrengthListByFriendsList(0, PlayerId, [FriendId|FriendList], [FirstStrength|[SecondS
trength|StrengthList]]):-
    (connection(PlayerId, FriendId, FirstStrength, SecondStrength, _, _);
    connection(FriendId, PlayerId, FirstStrength, SecondStrength, _, _)),
    aStar_getStrengthListByFriendsList(0, PlayerId, FriendList, StrengthList).
```

Já no predicado representativo do algoritmo A-Star, a estimativa é chamada, onde o cálculo é feito segundo a equação $Estimativa = H * (N - M)$, em que H represeta o maior valor atualmente presente na lista anteriormente calculada, N o número máximo de ligações e M, representa o nível atual da execução. Assim, é obtida a estimativa utilizada no predicado do A-Star.

```
aStar_estimate(N,M, Est):-
    retract(aStar_orderedList([H|List])),
    Est is H * (N - M),
    asserta(aStar_orderedList(List)).
```

5. ADAPTAÇÃO DO BEST FIRST AO PROBLEMA DA DETERMANAÇÃO DO CAMINHO MAIS FORTE (MÁXIMO DE N LIGAÇÕES)

5.1. Funcionalidade desenvolvida por Beatriz Seixas

Nesta funcionalidade, o predicado chamado irá ser o *best_first*/7. Os parâmetros para este predicado serão *Mode*, que pode tomar um valor de 0 quando apenas considera a força de ligação ou o valor de 1, quando é executado o modo multicritério, tendo em conta a força de ligação e relação, implementada no predicado *getMulticriteria*/3); o segundo parâmetro neste método será *EmocionalBool*, um valor que indica se o estado emocional é considerado ou não, não estando este valor no âmbito do sprint C); *N, representa* o número máximo de ligações para limitar o algoritmo; *Orig, representando o* id do jogador de origem; *Dest*, o id do jogador de destino; *Cam*, será o caminho resultante que será retornado pela função desenvolvida neste requisito; por fim, *Cost* representa o custo do caminho retornado.

Ao predicado fornecido foi adicionado a condição de paragem baseada no número máximo de ligações fornecido por parâmetro. Para limitar o uso do algoritmo para N ligações foi utilizado um contador que irá identificar o nível atual, tendo como condição de paragem quando este contador M for maior ou igual ao nível máximo fornecido pelo utilizador N.

Um exemplo da utilização desta funcionalidade seria o cálculo do caminho do jogador 1 da base de conhecimento parcial para o utilizador com o id 5. Neste exemplo, iremos fornecer o modo apenas com forças de ligação (valor 0), ignorar o estado emocional pois essa funcionalidade é referente ao *sprint* seguinte, fornecer o nível máximo de 3 ligações, do jogador 1 para o 5. Com estes parâmetros, obtemos o caminho [1,2] que seria o caminho esperado, com um custo de 8. Numa segunda execução, fornecemos o nível máximo de zero e o predicado, como esperado, não encontra um caminho com um máximo de ligações desse valor.

.

```
best_first(Mode, EmotionBool, N, Orig,Dest,Cam,Custo):-
    best_firstAux(Mode, EmotionBool, 0, N, Dest,[[Orig]],Cam,Custo).

best_firstAux(Mode,EmotionBool,_,_, Dest,[[Dest|T]|_],Cam,Custo):-
    reverse([Dest|T],Cam),
    best_costCalc(Mode,EmotionBool,Cam,Custo).
best_firstAux(Mode, EmotionBool, M,N,Dest,[[Dest|_]|LLA2],Cam,Custo):-
    !, best_firstAux(Mode, EmotionBool, M, N,Dest,LLA2,Cam,Custo).
best_firstAux(_, _,M, N, _, _, _, _):-
    M >= N, !, false.
best_firstAux(0, EmotionBool,M, N, Dest,LLA,Cam,Custo):-
    best_firstMember(LA,LLA,LLA1),
    LA=[Act|_],
    ((Act==Dest,!,
      best_firstAux(0, EmotionBool, M, N, Dest,[LA|LLA1],Cam,Custo));
    (findall((CX,[X|LA]),
    ((connection(Act,X,CX,_,_,_);
     connection(X,Act,_,CX,_,_)),
    emotion_checkSameEmotion(EmotionBool, Act, X),
     \+member(X,LA)),Novos),
     Novos\==[],!,
     sort(0,@>=,Novos,NovosOrd),
     best_removeCosts(NovosOrd,NovosOrd1),
     append(NovosOrd1,LLA1,LLA2),
     M1 is M + 1,
     best_firstAux(0, EmotionBool, M1, N, Dest,LLA2,Cam,Custo) )).
```

```
?- best_first(0,0,3,1,2,L,C).
LLA2=[[3,1],[2,1]]
LLA2=[[5,3,1],[2,1]]
LLA2=[[2,5,3,1],[2,1]]
Caminho=[1,2]
L = [1, 2],
C = 8 .

?- best_first(0,0,0,1,2,L,C).
false.
```

6. ADAPTAÇÃO DO PRIMEIRO EM PROFUNDIDADE PARA GERAR A MELHOR SOLUÇÃO (JÁ IMPLEMENTADO NO SPRINT ANTERIOR) PARA O MÁXIMO DE N LIGAÇÕES

6.1. Funcionalidade Desenvolvida por Todos os Elementos do Grupo

À semelhança dos restantes, o Primeiro em Profundidade foi adaptado para um máximo de N ligações, utilizando um contador que vai verificando quando o nível atual é superior ao fornecido por parametro. Se esta condição de paragem for encontrada sem chegar ao destino, o algoritmo não considera este caminho. Um exemplo deste método seria o caminho entre o jogador 1 e o jogador 5, onde o caminho retornado será [1,3,5].

```
shortest_dfs(Mode, EmotionBool, N, Orig, Dest, Strength, Path):- shortest_dfsAux(Mode, Emotion
Bool, 0, N, Orig, Dest, [Orig], Strength, Path).

shortest_dfsAux(_, _, _, _, Dest, Dest, LA, 0, Path):- !, reverse(LA, Path).
shortest_dfsAux(_, _, M, N, _, _, _, _, _):- M >= N, !, false.
shortest_dfsAux(0, EmotionBool, M, N, Current, Dest, LA, Strength, Path):-
    (connection(Current, X, StrengthA, _, _, _);
    connection(X, Current, _, StrengthA, _, _)),
    emotion_checkSameEmotion(EmotionBool, Current, X),
    \+ member(X,LA),
    M1 is M + 1,
    shortest_dfsAux(0, EmotionBool, M1, N, X,Dest,[X|LA], Strength1, Path),
        Strength is Strength1 + StrengthA.
```

```
?- shortest_route(0,0,3,1,5,Strength,Path).
Solution generation time:8.797645568847656e-5
Strength = 23,
Path = [1, 3, 5].
```

7. COMPARAÇÃO DOS 3 MÉTODOS COM VÁRIOS EXEMPLOS, COMPARANDO TEMPOS DE GERAÇÃO DA SOLUÇÃO E VALOR DA SOLUÇÃO GERADA

Depth First Search

Nº de Camadas Intermédias (sem nós de origem e destino)	**Nº de soluções**	**Tempo para gerar a solução com menor nº de ligações**
2	255	0.0s
3	431	0.0s
4	631	0.0s

A*

Nº de Camadas Intermédias (sem nós de origem e destino)	**Nº de soluções**	**Tempo para gerar a solução com menor nº de ligações**
2	399	0.0s
3	641	0.12661398794866523
4	N/A	Tempo excessivo (>3 mins)

Best First Search

Nº de Camadas Intermédias (sem nós de origem e destino)	**Nº de soluções**	**Tempo para gerar a solução com menor nº de ligações**
2	217	0.0s
3	345	0.0s
4	436	0.0S

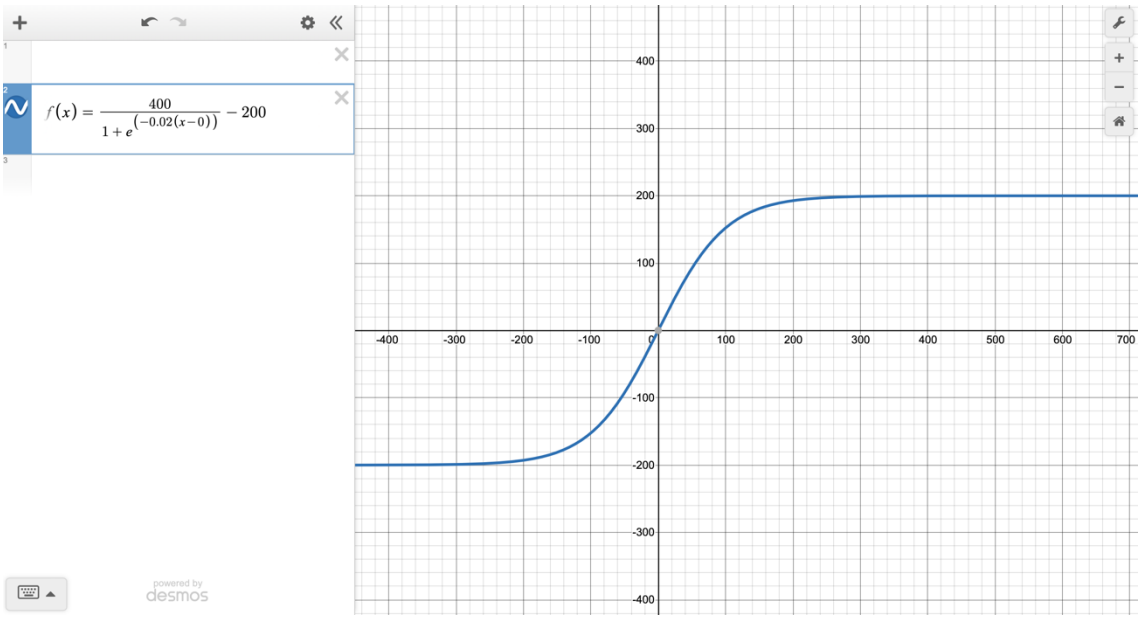
8. IMPLEMENTAÇÃO DA FUNÇÃO MULTICRITÉRIO QUE CONTEMPLE FORÇAS DE LIGAÇÃO E DIFERENÇA ENTRE LIKES E DISLIKES

8.1. Funcionalidade Desenvolvida por Todos os Elementos do Grupo

Para a implementação da força de relação (diferença entre *likes* e *dislikes*), como o problema é um de maximização, as estimativas deveriam ser um majorante. Para isso, foi implementada uma função com saturação de mínimo ou máximo, baseada em sigmoides, onde apresenta sempre valores entre -200 e 200, utilizando a equação:

$$f(x) = \frac{A_1}{1 + e^{\left(A_2\left(x - A_3\right)\right)}} + A_4$$

Onde A_1 representa o intervalo entre o limite superior e inferior da função (intervalo entre 200 e -200 no nosso caso, ou seja, 400); A_2 é o coeficiente de ganho, um valor que podemos ajustar para dar a curvatura que queremos à função (o valor escolhido foi -0.02); A_3 está relacionado com valor de x do ponto médio da função, sendo este o ponto de ganho máximo (no nosso caso, o ponto intermédio seria x = 0); por último, A_4 representa o limite inferior da função (-200 nesta implementação). Com estes valores, a equação calculada para esta implementação e sua representação gráfica seria a seguinte:



Assim, qualquer valor proveniente da diferença entre likes e *dislikes* será normalizado para um valor entre -200 e 200, sendo um exemplo do mesmo o valor inicial de 200 passará a 192.80549700575096, assim apresentando uma diferença visível entre valores superiores a 200 e o próprio número pois estes crescem infinitamente mas nunca ultrapassando o limite superior de 200. O mesmo não aconteceria numa implementação em que qualquer valor superior a 200 tivesse o mesmo impacto na força de relação do que o limite superior. Este tipo de funções, normalmente com limites de 0 e 1, são muito utilizados na área da aprendizagem automática (*Machine Learning*), nomeadamente em redes neuronais (*Neural Networks*). Esta função é utilizada ao ser introduzida a força de ligação no termo *connection*/6.

```
sigmoid(X, ReturnValue):-
    ReturnValue is (400/(1 + (2.71828 ** (-0.02 * X)))) - 200.
```

```
?- sigmoid(200, L).
L = 192.80549700575096.
```

Para o cálculo da força multicritério que incluí as forças de ligação e relação, segundo a tabela em baixo representada. Para isso, foi implementado o predicado *getMulticriteria*/3, onde este recebe uma força de ligação e outra de relação (já normalizada pelo predicado e função previamente descritos), calcula a força multicritério, tendo cada força individual um peso de 50% no valor final (de 0 a 100). Como a força de ligação já varia de 0 a 100, é necessário apenas a sua divisão por 2, já a de relação (que varia entre -200 e 200 pois esta já se encontra com limites aplicados), é necessário somar 200, passando esta a variar entre 0 e 400, assim ao ser dividida por 4, obtemos um valor de 0 a 100 e similarmente à força de ligação, esta é divida novamente por 2 para obter um valor que varia entre 0 e 50. Um exemplo seria uma força de ligação de 50 e uma força de relação de 200 resultariam numa força multicritério de 75.

```
getMulticriteria(ConnectionStrength, RelationStrength, Output):-
    MultiConnection is ConnectionStrength / 2,
    MultiRelation is ((RelationStrength + 200) / 4) / 2,
    Output is MultiConnection + MultiRelation.
```

Força de Ligação	Força de Relação Likes-Dislikes	Resultado da Função Multicritério
0	≤-200	0
0	0	25
0	≥+200	50
50	≤-200	25
50	0	50
50	≥+200	75
100	≤-200	50
100	0	75
100	≥+200	100

```
?- getMulticriteria(50, 200, Output).
Output = 75.
```

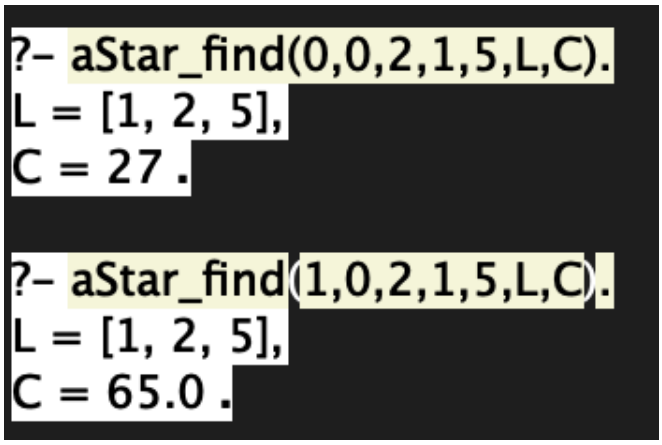
9. ADAPTAÇÃO DOS 3 MÉTODOS (PRIMEIRO EM PROFUNDIDADE, BEST FIRST E A*) PARA CONSIDERAR A FUNÇÃO MULTICRITÉRIO DO PONTO ANTERIOR

9.1. A (Funcionalidade Desenvolvida por Jéssica Alves (1190682))*

Ao método anteriormente referente ao algoritmo de pesquisa A-Star anteriormente desenvolvido, foi adicionado o cálculo da função multicritério, descrita previamente no presente relatório, utilizando uma função com saturação de mínimo ou máximo (baseada em sigmoides). Para isto, foi adicionado um parâmetro denominado de modo ao predicado, com o intuito de chamar o conversor *getMulticriteria*/3 para utilizar a força multicritério (força de ligação e relação) no cálculo do custo presente no algoritmo A-Star. A estimativa também foi ajustada para ter em consideração da força multicritério.

Um exemplo desta implementação seria a mudança de modo para o caminho com origem no jogador com id 1 para o destino com o id 5 da base de conhecimento parcial, até 2 níveis de ligação. Na primeira execução, utilizando o modo de força de ligação, este retorna um custo de 27, enquanto utilizando o modo multicritério, o custo transforma-se em 65, podendo verificar a diferença destes modos.

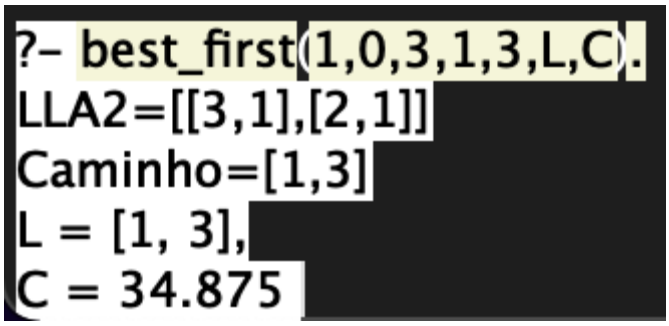
```
aStar_aux(1, EmotionBool, M, N, Dest,[(_,Ca,LA)|Others],Path,Cost):-
    LA=[Act|_],
    findall((CEX,CaX,[X|LA]),
    (Dest\==Act,
    (connection(Act,X,ConnStrength, _, RelStrength, _);
    connection(X, Act, _, ConnStrength, _, RelStrength)),
    emotion_checkSameEmotion(EmotionBool, Act, X),
    \+ member(X,LA),
    getMulticriteria(ConnStrength, RelStrength, CostX),
    CaX is CostX + Ca,
    aStar_estimate(N,M,EstX),
    CEX is CaX + EstX),
    New),
    append(Others,New,All),
    sort(All,AllOrd),
    M1 is M + 1,
    aStar_aux(1, EmotionBool, M1, N, Dest,AllOrd,Path,Cost).
```



9.2. Best First (Funcionalidade Desenvolvida por Tiago Costa)

Ao método anteriormente referido, referente ao algoritmo de pesquisa Best First anteriormente desenvolvido, foi adicionado o cálculo da função multicritério, descrita previamente no presente relatório, utilizando uma função com saturação de mínimo ou máximo (baseada em sigmoides). Para isto, foi adicionado um parâmetro denominado de modo ao predicado, com o intuito de chamar o conversor *getMulticriteria*/3 para utilizar a força multicritério (força de ligação e relação) se o mode tiver o valor de 1.

```
best_firstAux(1, EmotionBool, M, N, Dest,LLA,Cam,Custo):-
    best_firstMember(LA,LLA,LLA1),
    LA=[Act|_],
    ((Act==Dest,!,
      best_firstAux(1, EmotionBool,M, N, Dest,[LA|LLA1],Cam,Custo));
    (findall((CX,[X|LA]),
    ((connection(Act,X,ConStrength,_,RelationStrength,_);
     connection(X,Act,_,ConStrength,_,RelationStrength)),
     emotion_checkSameEmotion(EmotionBool, Act, X),
     getMulticriteria(ConStrength,RelationStrength, CX),
     \+member(X,LA)),Novos),
     Novos\==[],!,
     sort(0,@>=,Novos,NovosOrd),
     best_removeCosts(NovosOrd,NovosOrd1),
     append(NovosOrd1,LLA1,LLA2),
     M1 is M + 1,
     best_firstAux(1, EmotionBool,M1, N, Dest,LLA2,Cam,Custo) )).
```

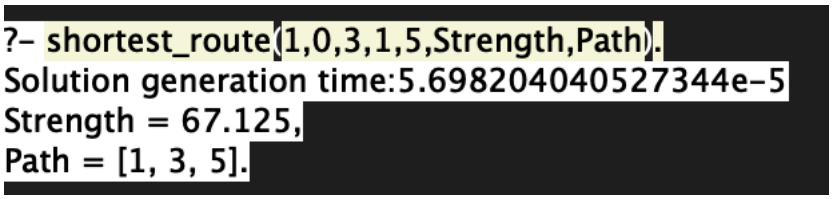


9.3. Primeiro em Profundidade (Funcionalidade Desenvolvida por Todos os Elementos do Grupo)

À semelhança dos predicados anteriormente descritos, o DFS caso o modo seja igual a 1 (modo multicritério), chama o predicado *getMulticriteria*/3 para considerar as forças de ligação e relação.

Um exemplo deste predicado seria o caminho [1,3,5] da base de conhecimento parcial teria uma força multicritério total de 67.125.

```
shortest_dfsAux(1, EmotionBool, M, N, Current, Dest, LA, Strength, Path):-
    (connection(Current, X, StrengthA, _, RelA, _);
    connection(X, Current, _, StrengthA, _, RelA)),
    emotion_checkSameEmotion(EmotionBool, Current, X),
    \+ member(X,LA),
    M1 is M + 1,
    shortest_dfsAux(1, EmotionBool, M1, N, X,Dest,[X|LA], Strength1, Path),
    getMulticriteria(StrengthA, RelA, FinalStrength),
    Strength is Strength1 + FinalStrength.
```



10. COMPARAÇÃO DOS 3 MÉTODOS COM VÁRIOS EXEMPLOS E USANDO A FUNÇÃO MULTICRITÉRIO

Depth First Search

Nº de Camadas Intermédias (sem nós de origem e destino)	Nº de soluções	Tempo para gerar a solução com menor nº de ligações
2	319,5	0.0s
3	411,0	0.0s
4	631	0.0s

A*

Nº de Camadas Intermédias (sem nós de origem e destino)	Nº de soluções	Tempo para gerar a solução com menor nº de ligações
2	345,5	0.0s
3	533	0.17661398794866523
4	N/A	Tempo excessivo (>3 mins)

Best First Search

Nº de Camadas Intermédias (sem	Nº de soluções	Tempo para gerar a solução com menor nº de ligações

nós de origem e destino)		
2	226,0	0.0s
3	404,0	0.0s
4	371,0	0.0S

Comparando os resultados dos algoritmos com os algoritmos em que so se considera a força de ligação, observamos semelhanças, logo podemos concluir que o multicritério não afeta a eficiência.

11. CONCLUSÕES

A introdução dos novos algoritmos Best First e A*

- No geral estes algoritmos revelam-se mais eficientes que a pesquisa em profundidade.
- É possível que o A* não encontre a melhor solução em termos de custo
- permitiu atingir resultados num tempo inferior, em comparação com os algoritmos anteriores
- No entanto não houve grandes alterações no tempo de resposta, comparando com os algoritmos anteriores, pelo que não tem praticamente impacto nenhum