22-01-2022

Relatório ALGAV

Sprint C

Grupo:

- Helder Serralva 1181180
- Diogo Silva 1181178
- Diogo Ribeiro 1190508
- Ricardo Cardoso 1171388

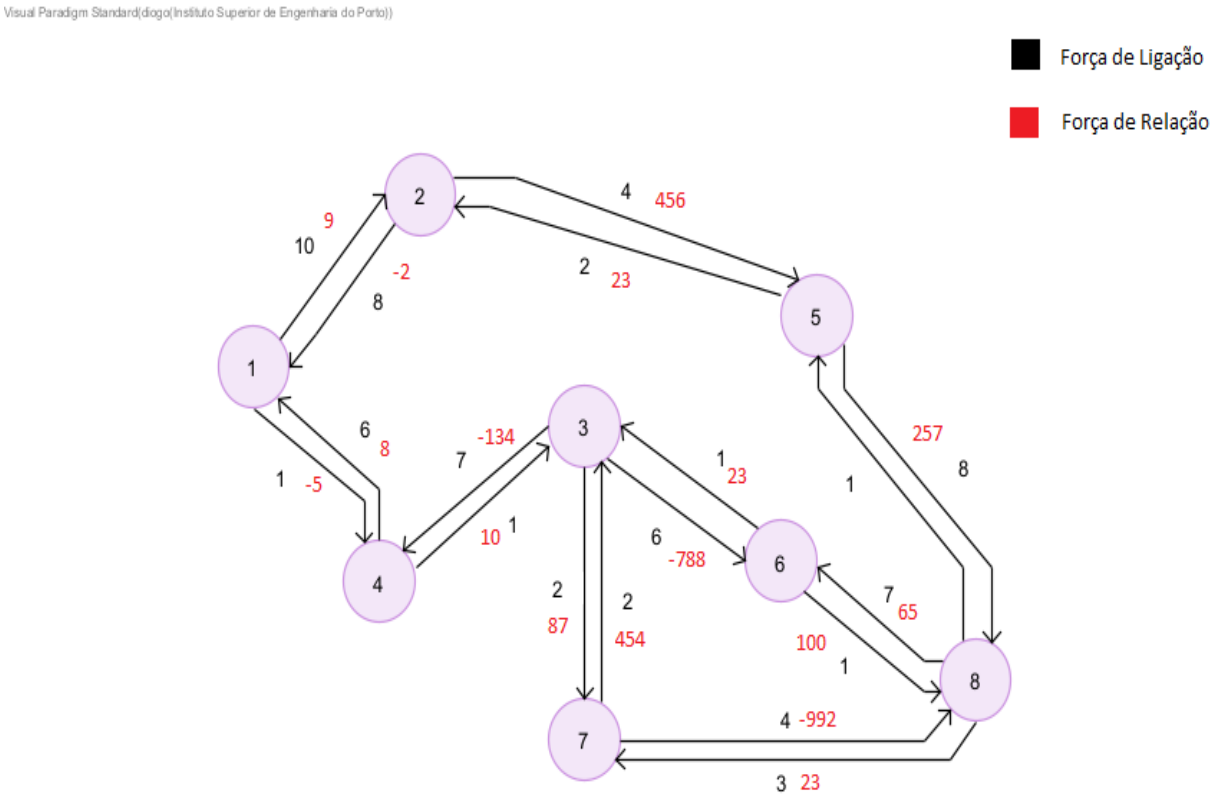
Turma 3DE

Índice

Índice

Criação de uma rede

Nesta secção será apresentada a rede criada para testar as funcionalidades implementadas neste sprint. Esta base de conhecimento já foi implementada no sprint passado e foi reaproveitada para este, introduzindo as forças de relação. Tendo em conta que é uma rede pequena, é fácil perceber as soluções esperadas para qualquer que seja o nó origem e destino.



Adaptação do A* ao problema da determinação do caminho mais forte (máximo de N ligações)

Era pretendido que adaptássemos o algoritmo dados nas teóricas implementado para outro contexto para o problema da determinação do caminho mais forte. Para além disso, deveria também ser possível estabelecer um limite máximo de ligações, uma vez que esta pesquisa pode ser muito extensa.

Para começar começamos por implementar uma verificação da existência do nó destino, isto porque caso o mesmo não exista, a pesquisa vai tentar encontrar todas as soluções possíveis e só depois retornará false. Desta forma, caso o nó destino não exista, o algoritmo devolve de imediato a resposta. A pesquisa em si "aStar2" só será então feita se esta condição se verificar.

```
aStar(Orig,Dest,MaxLigacoes, Cam,Custo):-
    no(Dest,_), % Nao pesquisar caso o no destino nao exista.
    forcasOrdenadas(Forcas), Ignorar por enquanto (usado para a estimativa)
    aStar2(Dest,[(_,0,[Orig])],MaxLigacoes,Cam,Custo, Forcas).
```

De seguida, foi introduzida então a opção de limitar o número de ligações da solução. O que algoritmo começa por fazer é verificar se o primeiro nó da melhor solução até ao momento é igual ao nó destino, e no caso de ser é feita a inversão dessa lista de caminhos e apresentada a solução. Caso isto não aconteça, vai á primeira possível solução (mais forte até ao momento) e é determinado o número de nós da mesma. Caso a possível solução já ultrapasse o número máximo de ligações, a mesma é ignorada e passa-se a ter em consideração a segunda mais forte (próximo elemento da lista de possíveis soluções).

```
aStar2(Dest,[(_,Custo,[Dest|T])|_],_,Cam,Custo, _):-
    reverse([Dest|T],Cam),!.

aStar2(Dest,[(_,_,LA)|Outros],MaxLigacoes,Cam,Custo, Forcas):-
     length(LA, Tam),
     Tam > MaxLigacoes,
     aStar2(Dest, Outros, MaxLigacoes, Cam,Custo, Forcas),!.
```

A solução atual (que já ultrapassa o limite de ligações é ignorada

Se nenhuma destas condições acima se verificar, então o algoritmo irá calcular os possíveis novos destinos a partir do primeiro nó da melhor solução até ao momento e adicioná-los á lista de possíveis soluções, assim como o seu custo, que neste momento é apenas a força de ligação entre os dois nós. Após adicionar os novos caminhos possíveis, a lista de possíveis soluções volta a ser ordenada de forma descendente de acordo com o custo (primeiro parâmetro da estrutura que vai ser ordenada). Isto acontece então até que o primeiro elemento da primeira solução possível seja igual ao nó destino.

```
aStar2(Dest,[(_,Ca,LA)|Outros],MaxLigacoes,Cam,Custo, Forcas):-
    length(LA, Tam),
    Tam =< MaxLigacoes,
    LA=[Act|_],
    findall((CEX,CaX,[X|LA]),
            (Dest\==Act,(ligacao(Act,X,CustoX,_,FRX,_);ligacao(X,Act,_,CustoX,_,FRX)),\+ member(X,LA),
             append([X], LA, T),
             removerForcasUsadas(T, Forcas, ForcasRestantes),
             NiveisRestantes is MaxLigacoes - Tam,
             CaX is CustoX + Ca, estimativa(ForcasRestantes,NiveisRestantes,EstX),
             CEX is CaX +EstX),Novos),

    append(Outros,Novos,Todos),
    sort(0, @>=, Todos,TodosOrd),
    aStar2(Dest,TodosOrd,MaxLigacoes,Cam,Custo, Forcas).

estimativa(_,_,0).
```

Apenas necessário para o calculo das estimativas, ainda não apresentado aqui.

Exemplos:

- Caminho do nó 3 para o 8. (Primeiro exemplo, limitando a 3 ligações no máximo, segundo exemplo, limitando a 7).

```
?- aStar(3,8, 3, Cam, Custo).
Cam = [3, 6, 8],
Custo = 7.

?- aStar(3,8, 7, Cam, Custo).
Cam = [3, 4, 1, 2, 5, 8],
Custo = 35.
```

Estimativa

Para o cálculo da estimativa, seguimos a estratégia sugerida no pdf de apoio ao sprint, ou seja, ter uma lista ordenada com as forças de todos as ligações (com repetidos), sendo que a estimativa seria a soma das primeiras N forças, ainda não usadas (sendo N o número de níveis até ao nível máximo).

Primeiro começamos por desenvolver um método que percorresse todas as ligações e devolvesse então essa lista ordenada com todas as forças encontradas, sendo que maior força é o primeiro elemento e a menor o último.

```
:- dynamic forcas/1.

forcasOrdenadas(Res):-
                    (percorrerLigacoes();true),
                    findall(F, forcas(F), Lf),
                    retractall(forcas(_)),
                    sort(0, @>=, Lf, Res).

percorrerLigacoes():-
                    ligacao(_,_,F1,F2,_,_ ),
                    asserta(forcas(F1)),
                    asserta(forcas(F2)),
                    fail.
```

Como vimos na secção anterior, este método é só chamado uma vez no início do algoritmo, após verificar que o nó destino existe, e o resultado é passado por parâmetro no campo "Forcas" para que possa ser usado durante o processo. Para o cálculo das estimativas de um caminho, era necessário considerar as forças que ainda não foram usadas. Foi implementado o predicado "removerForcasUsadas", que recebe 2 parâmetros, o primeiro é o caminho a considerar, e o segundo a lista de todas as forças existentes. Para cada ligação, é encontrada a força e removida da lista das forças totais. Isto acontece até que sejam verificadas todas as ligações do possível caminho.

```
removerForcasUsadas([_], F,F):-!.
removerForcasUsadas([NoAtual|Caminho], TodasForcas, R):-
                Caminho=[ProximoNo|_],
                (ligacao(NoAtual, ProximoNo, _,Forca,_,_);ligacao(ProximoNo, NoAtual,Forca,_, _,_)),!,
                deleteFirst(Forca, TodasForcas, TF),
                removerForcasUsadas(Caminho, TF, R).
```

A estimativa poderia então ser calculada, como foi dito, pela soma das N primeiras forças restantes, para isso, foi implementado o predicado" somaDosPrimeirosN". Este predicado soma todas as primeiras N forças ou todas as forças mesmo, caso o N seja maior que o número total de forças restantes.

```
estimativa(ForcasRestantes,NiveisRestantes,E):-
                somaDosPrimeirosN(ForcasRestantes, NiveisRestantes, E).


somaDosPrimeirosN([],_,0):-!.
somaDosPrimeirosN(_, 0, 0):-!.
somaDosPrimeirosN([F1|F], N, E):-
                N1 is N - 1,
                somaDosPrimeirosN(F, N1, T),
                E is T + F1.
```

Adaptação do Best First ao problema da determinação do caminho mais forte (máximo de N ligações)

Assim como para A*, foi pedido que adaptássemos este algoritmo, dado nas teóricas, ao contexto do projeto e que existisse a opção de limitar o número de ligações da solução encontrada.

As alterações que fizemos a este algoritmo foram muito parecidas ao que já foi descrito para o A*, em termos de limitar as ligações, a estratégia usada foi exatamente a mesma e também foi implementada a verificação da existência do nó destino. Sendo assim, tudo o que foi feito neste algoritmo foi substituir o edge existente pela ligação do nosso contexto, e o custo é determinado a partir dessa ligação também.

```
bestfs12(Dest,LLA,MaxLigacoes,Cam,Custo):-
        member1(LA,LLA,LLA1),
        length(LA, Tam),
        Tam =< MaxLigacoes,
        LA=[Act|_],
        ((Act==Dest,!,bestfs12(Dest,[LA|LLA1],MaxLigacoes,Cam,Custo))
         ;
         (
          findall((CX,[X|LA]),((ligacao(Act,X,CX,_,_,_);ligacao(X,Act,_,CX,_,_)),
          \+member(X,LA)),Novos),
          Novos\==[],!,
          sort(0,@>=,Novos,NovosOrd),
          retira_custos(NovosOrd,NovosOrd1),
          append(NovosOrd1,LLA1,LLA2),
          bestfs12(Dest,LLA2,MaxLigacoes,Cam,Custo)
         )).

member1(LA,[LA|LAA],LAA).
member1(LA,[_|LAA],LAA1):-member1(LA,LAA,LAA1).

retira_custos([],[]).
retira_custos([(_,LA)|L],[LA|L1]):-retira_custos(L,L1).

calcula_custo([Act,X],C):-!,(ligacao(Act,X,C,_,_,_);ligacao(X,Act,_,C,_,_)).
calcula_custo([Act,X|L],S):-calcula_custo([X|L],S1),
                            (ligacao(Act,X,C,_,_,_);ligacao(X,Act,_,C,_,_)),S is S1+C.
```

Como podemos ver na imagem acima, toda a pesquisa de um novo caminho a partir do atual só é feita caso o número de ligações ainda seja inferior ao máximo estabelecido.

Exemplos:

- Caminho do nó 3 para o 8. (Primeiro exemplo, limitando a 3 ligações no máximo, segundo exemplo, limitando a 7, terceiro exemplo, limitando a 1, que retorna false uma vez o nós 3 e 8 não tem uma ligação direta).

```
?- bestfs(3,8,3,Cam,Custo).
Cam = [3, 6, 8],
Custo = 7.

?- bestfs(3,8,7,Cam,Custo).
Cam = [3, 4, 1, 2, 5, 8],
Custo = 35.

?- bestfs(3,8,1,Cam,Custo).
false.
```

Adaptação do Primeiro em profundidade ao problema da determinação do caminho mais forte (máximo de N ligações)

Tal como nos algoritmos acima descritos, a adaptação deste também foi bastante simples. Apenas determinamos o tamanho do caminho até ao momento e só é permitida a continuação da pesquisa para esse caminho caso esse tamanho não seja superior ao limite máximo de ligações. Como o dfs devolve mais do que uma solução (através do backtracking), as soluções que não verifiquem essa condição são ignoradas.

```
dfs(Orig,Dest,Cam, FCam, MForca, MaxLigacoes):-
                dfs2(Orig,Dest,[Orig],0,101, Cam, FCam, MForca, MaxLigacoes).
dfs2(Dest,Dest,LA,FA,MF, Cam, FA, MF,_):-!,reverse(LA,Cam).
dfs2(Act,Dest,LA,FA,MF, Cam, FCam, MForca, ML):-
                length(LA, Tam),
                Tam =< ML,
                (ligacao(Act,NX,F1,F2,_,_);ligacao(NX,Act,F2,F1,_,_)),
                menor_forca(F1,F2,MF,Menor),
                \+ member(NX,LA),
                FT is FA + F1,
                dfs2(NX,Dest,[NX|LA],FT,Menor, Cam, FCam, MForca, ML).
```

Vejamos alguns exemplos:

- Caminho do nó 3 para o 8. (Primeiro exemplo, limitando a 3 ligações no máximo, segundo exemplo, limitando a 7, terceiro exemplo, limitando a 1, que retorna false uma vez o nós 3 e 8 não tem uma ligação direta).

```
?- dfs(3,8,Cam,Custo,_,3).
Cam = [3, 7, 8],
Custo = 6 ;
Cam = [3, 6, 8],
Custo = 7 ;
false.

?- dfs(3,8,Cam,Custo,_,7).
Cam = [3, 4, 1, 2, 5, 8],
Custo = 35 ;
Cam = [3, 7, 8],
Custo = 6 ;
Cam = [3, 6, 8],
Custo = 7 ;
false.

?- dfs(3,8,Cam,Custo,_,1).
false.
```

Comparação dos 3 algoritmos

Nesta secção será apresenta uma breve comparação entre os 3 algoritmos, comparando tempos e soluções obtidas por cada um. De notar, que nesta comparação só as forças de ligação são tidas em conta.

Origem	Destino	Nr Ligações	Nivel Corte	Resultados			
				Algoritmo	Caminho	Custo	Tempo
1	8	23	4	Mais Forte (DFS)	1,3,4,10,8	48	0.001035928726196289
				A*	1,4,10,8	35	0.001008033752441406
				BestFirst	1,2,6,7,8	24	0.0
			8	Mais Forte (DFS)	1, 2, 5, 3, 4, 7, 9, 10, 8	96	0.003988027572631836
				A*	1, 3, 4, 10, 8	48	0.001017093658447265
				BestFirst	1, 2, 6, 4, 10, 8	56	0.0
		26	4	Mais Forte (DFS)	1, 7, 9, 10, 8	70	0.000998020172119140
				A*	1, 7, 9, 10, 8	70	0.000981092453002929
				BestFirst	1, 7, 6, 2, 3, 5, 8	65	0.0
			10	Mais Forte (DFS)	1, 4, 6, 2, 3, 5, 7, 9, 10,8	103	0.007016897201538086
				A*	1, 7, 9, 10, 8	70	0.001019001007080078
				BestFirst	1, 7, 6, 2, 3, 5, 8	65	0.0
		30	6	Mais Forte (DFS)	1, 7, 5, 3, 9, 10, 8	96	0.013998985290527344
				A*	1, 7, 9, 10, 8	70	0.001014947891235351
				BestFirst	1, 7, 6, 2, 8	55	0.0
			12	Mais Forte (DFS)	1, 7, 4, 2, 6, 5, 3, 9, 10,8	127	0.044025182723999021
				A*	1,5,3,9,10,8	82	0.001015901565551757
				BestFirst	1, 7, 6, 2, 8	55	0.0

A primeira conclusão que podemos tirar daqui é que sem sombra de dúvida o algoritmo best first é o mais apropriado para nos dar uma solução no melhor tempo possível. Contudo, podemos reparar que em termos de resultados também foi o algoritmo que mais vezes nos deu a solução com menor custo de entre os 3, o que neste caso não é o pretendido, o custo representa a força de ligação, e o que nos interessa é o caminho com maior força de ligação possível. A nível de resultados, podemos concluir que o algoritmo implementado no sprint anterior nos garante o caminho com a maior força possível, contudo o tempo de solução é visivelmente afetado pelo nível de ligações e pelo nível de corte (máximo de ligações), sendo que este último fator parece ter um peso maior nesse aspeto. Já o A* é quase como se fosse um intermédio destes dois, dá-nos uma solução melhor do que a do Best First mas é pior em termos de tempo, dá-nos uma solução pior do que a do Mais Forte (DFS) mas compensa em questões de tempo.

Desta forma podemos concluir que o algoritmo a ser usado depende muito daquilo que se pretenda fazer. Se o pretendido for encontrar um caminho apenas de um ponto a outro sem a que força de ligação tenha necessariamente de ser a maior, o Best First é sem dúvida o algoritmo a ser usado. Se a força de ligação importar, podemos considerar usar o DFS para tentar encontrar um caminho se os dois pontos estiverem a poucos nós de distância um do outro, caso sejam nós distantes um do outro, o uso do A* terá as suas vantagens sobre este.

Função Multicritério

Uma vez que a força de ligação não tem o mesmo peso que as forças de relação, ou seja, não podem ser simplesmente somadas de igual forma, implementamos esta função multicritério que relaciona as duas de maneira igual e devolve o resultado da combinação das duas.

Tendo em conta que a força de relação é a diferença entre likes e dislikes é impossível saber o intervalo de valores que esta força pode tomar. Desta forma, seguimos a estratégia sugerida no pdf de apoio e aplicamos os limites de –200 a 200. Assim, qualquer força com valor inferior a –200 passa a ter o valor –200, qualquer força com o valor superior a 200 fica com o valor 200, as restantes forças ficam com o valor original.

```
aplicarLimitesNaForcaRelacao(ForcaRelacao,-200):-
                ForcaRelacao < -200,!.

aplicarLimitesNaForcaRelacao(ForcaRelacao,200):-
                ForcaRelacao > 200,!.

aplicarLimitesNaForcaRelacao(ForcaRelacao, ForcaRelacao).
```

Com a força de relação assim tratada, sabemos que esta só pode variar de –200 a 200, ou seja, pode assumir um intervalo de 400 valores. Para que seja mais fácil o cálculo, somamos a esta força o valor 200 para garantir que só lidamos com valores positivos, assim, o intervalo fica de 0 a 400.

Por fim, faltaria apenas calcular a percentagem de cada força tendo em conta o intervalo de valores de cada tipo de força. A força de ligação varia de 0 a 100, então não foi preciso qualquer cálculo para esta uma vez que já está representada em percentagem. Já para a força de relação tivemos que a dividir pelo número de valores que esta podia assumir (400 – de –200 a 200) e de seguida multiplicar por 100 para nos dar o valor da percentagem. Daqui faltava somar 50% do valor de cada força, daí a divisão por 2, e teríamos o nosso resultado.

```
funcaoMulticriterio(ForcaLigacao, ForcaRelacao, Resultado):-
                aplicarLimitesNaForcaRelacao(ForcaRelacao, FR),
                F is FR + 200,
                PFR is (F * 100 / 400) /2,
                PFL is ForcaLigacao / 2,
                Resultado is PFR + PFL.
```

Podemos testar esta função comparando os resultados obtidos com a tabela do PDF de apoio.

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

```
?- funcaoMulticriterio(0,-245,R).
R = 0.

?- funcaoMulticriterio(0,0,R).
R = 25.

?- funcaoMulticriterio(0,456,R).
R = 50.

?- funcaoMulticriterio(50,-2323,R).
R = 25.

?- funcaoMulticriterio(50,0,R).
R = 50.

?- funcaoMulticriterio(50,4324234,R).
R = 75.

?- funcaoMulticriterio(100,-200,R).
R = 50.

?- funcaoMulticriterio(100,0,R).
R = 75.

?- funcaoMulticriterio(100,200,R).
R = 100.
```

Adaptação dos 3 algoritmos para usar a função Multicritério

Depois de implementada a função multicritério, faltava apenas adaptar os nossos algoritmos para fazerem uso da mesma.

Em relação ao dfs, as alterações foram relativamente pequenas, ao encontrar a próxima ligação, em vez de ir buscar apenas a força de ligação FL, era também retirado a força de relação FR, e a soma do custo era agora feito somando o resultado da função multicritério F1.

```
dfs2(Act,Dest,LA,FA,MF, Cam, FCam, MForca, ML):-
                length(LA, Tam),
                Tam =< ML,
                (ligacao(Act,NX,FL1,FL2,FR1,FR2);ligacao(NX,Act,FL2,FL1,FR2,FR1)),
                funcaoMulticriterio(FL1, FR1, F1),
                funcaoMulticriterio(FL2, FR2, F2),
                menor_forca(F1,F2,MF,Menor),
                \+ member(NX,LA),
                FT is FA + F1,
                dfs2(NX,Dest,[NX|LA],FT,Menor, Cam, FCam, MForca, ML).
```

No que diz respeito ao A*, a mesma coisa, na altura de obter a próxima ligação tínhamos também em consideração a força de relação FRX e a função multicritério devolverá a combinação das duas forças, o CustoX. Como podemos ver, é este CustoX que é somado ao custo anterior.

```
aStar2(Dest,[(_,Ca,LA)|Outros],MaxLigacoes,Cam,Custo, Forcas):-
    length(LA, Tam),
    Tam =< MaxLigacoes,
    LA=[Act|_],
    findall((CEX,CaX,[X|LA]),
            (Dest\==Act,(ligacao(Act,X,FLX,_,FRX,_);ligacao(X,Act,_,FLX,_,FRX)),\+ member(X,LA),
             append([X], LA, T),
             removerForcasUsadas(T, Forcas, ForcasRestantes),
             NiveisRestantes is MaxLigacoes - Tam,
             funcaoMulticriterio(FLX, FRX, CustoX),
             CaX is CustoX + Ca, estimativa(ForcasRestantes,NiveisRestantes,EstX),
             CEX is CaX +EstX),Novos),

    append(Outros,Novos,Todos),
    sort(0, @>=, Todos,TodosOrd),
    aStar2(Dest,TodosOrd,MaxLigacoes,Cam,Custo, Forcas).
```

Para o Best First, sabemos que é usada uma estrutura que tem no primeiro parâmetro o custo e no caminho a considerar. Aquilo que foi alterado, foi a forma de obter esse custo. Tal como nos outros algoritmos, a força de relação FR passou a ser também retirada da ligação e a função multicritério é aplicada tendo em conta essa força e a de ligação FL. O resultado é guardado na variável CX, que como podemos ver é a variável responsável por guardar o custo do caminho.

```
bestfs12(Dest,LLA,MaxLigacoes,Cam,Custo):-
        member1(LA,LLA,LLA1),
        length(LA, Tam),
        Tam =< MaxLigacoes,
        LA=[Act|_],
        ((Act==Dest,!,bestfs12(Dest,[LA|LLA1],MaxLigacoes,Cam,Custo))
         ;
         (
          findall((CX,[X|LA]),((ligacao(Act,X,FL,_,FR,_);ligacao(X,Act,_,FL,_,FR)),
          \+member(X,LA),funcaoMulticriterio(FL,FR,CX)),Novos),
          Novos\==[],!,
          sort(0,@>=,Novos,NovosOrd),
          retira_custos(NovosOrd,NovosOrd1),
          append(NovosOrd1,LLA1,LLA2),
          bestfs12(Dest,LLA2,MaxLigacoes,Cam,Custo)
         )).
```

Vejamos os resultados para os exemplos testados anteriormente sem a implementação desta função multicritério:

```
?- dfs(3,8,Cam,Custo,_,3).
Cam = [3, 7, 8],
Custo = 38.875 ;
Cam = [3, 6, 8],
Custo = 41.0 ;
false.

?- aStar(3,8,3,Cam,Custo).
Cam = [3, 6, 8],
Custo = 41.0.

?- bestfs(3,8,3,Cam,Custo).
Cam = [3, 7, 8],
Custo = 38.875.
```

Podemos ver que, para além dos custos serem agora diferentes, o resultado que o bestfs nos dá é diferente do que nos deu na altura em que as forças de relações ainda não eram tidas em conta.

Comparação dos 3 algoritmos (tendo em conta a função multicritério)

Para esta comparação vamos usar os mesmos exemplos que foram usados para a comparação sem considerar as forças de relação.

Origem	Destino	Nr Ligações	Nivel Corte	Resultados							
				Algoritmo	Caminho	Custo	Tempo		Caminho	Custo	Tempo
1	8	23	4	Mais Forte (DFS)	1,3,4,10,8	48	0.001035928726196289	Função Multricritério	1, 3, 2, 5, 8	189.875	0.001012086868286132
				A*	1,4,10,8	35	0.001008033752441406		1, 3, 5, 8	149.75	0.001023054122924804
				BestFirst	1,2,6,7,8	24	0.0		1, 3, 5, 8	149.75	0.0
			8	Mais Forte (DFS)	1, 2, 5, 3, 4, 7, 9, 10, 8	96	0.003988027572631836		1, 3, 7, 10, 4, 6, 2, 5, 8	323.125	0.007992029190063477
				A*	1, 3, 4, 10, 8	48	0.001017093658447265		1, 3, 5, 8	149.75	0.000997066497802734
				BestFirst	1, 2, 6, 4, 10, 8	56	0.0		1, 3, 5, 8	149.75	0.0
		26	4	Mais Forte (DFS)	1, 7, 9, 10, 8	70	0.000998020172119140		1, 3, 2, 5, 8	189.875	0.002040863037109375
				A*	1, 7, 9, 10, 8	70	0.000981092453002929		1, 3, 5, 8	149.75	0.001023054122924804
				BestFirst	1, 7, 6, 2, 3, 5, 8	65	0.0		1, 3, 5, 8	149.75	0.0
			10	Mais Forte (DFS)	1, 4, 6, 2, 3, 5, 7, 9, 10,8	103	0.007016897201538086		1, 3, 7, 9, 10, 4, 6, 2, 5,8	359.125	0.02198505401611328
				A*	1, 7, 9, 10, 8	70	0.001019001007080078		1, 3, 5, 8	149.75	0.00099921226501464
				BestFirst	1, 7, 6, 2, 3, 5, 8	65	0.0		1, 3, 5, 8	149.75	0.0
		30	6	Mais Forte (DFS)	1, 7, 5, 3, 9, 10, 8	96	0.013998985290527344		1, 3, 7, 4, 2, 5, 8	284.0	0.026026010513305664
				A*	1, 7, 9, 10, 8	70	0.001014947891235351		1, 3, 5, 8	149.75	0.000984907150268554
				BestFirst	1, 7, 6, 2, 8	55	0.0		1, 3, 5, 8	149.75	0.0
			12	Mais Forte (DFS)	1, 7, 4, 2, 6, 5, 3, 9, 10,8	127	0.044025182723999021		1, 3, 7, 9, 10, 4, 2, 6, 5,...,8	396.875	0.103024005889892582
				A*	1,5,3,9,10,8	82	0.001015901565551757		1, 3, 5, 8	149.75	0.000974893569946289
				BestFirst	1, 7, 6, 2, 8	55	0.0		1, 3, 5, 8	149.75	0.0

Podemos ver que os resultados foram em todos os casos diferentes, isto porque a força de relação foi tida em conta desta vez. Podemos também concluir que para o Best First e para o A* os tempos mantiveram-se praticamente iguais, com a particularidade de estes darem em todos os testes o mesmo resultado. No que diz respeito ao DFS, é possível reparar que os tempos aumentaram significativamente passando, em média, a demorar o dobro do tempo que demorava para obter um caminho sem ter em conta as forças de relação.

Conclusões

Ao longo do relatório temos referido o DFS como o que nos apresenta a melhor solução e o que demora mais tempo. Sabemos que isto apenas acontece porque o nosso algoritmo vai percorrer todos os resultados dados pelo DFS e aí sim garantir a melhor solução. Se tivéssemos em conta apenas a primeira solução dada pelo dfs então não existia qualquer garantia de ser o melhor caminho e o tempo seria muito inferior, tal como acontece nos outros dois algoritmos. Contudo, ao analisar as coisas desta forma, permite-nos uma melhor comparação entre as soluções obtidas, podendo perceber melhor o quão longe ficou uma determinada solução da melhor solução. A questão dos tempos entre os dois algoritmos e o DFS deve ser ignorada já que, como foi dito, não para na primeira solução (A* e Best First), mas só quando todas as possíveis foram percorridas (DFS – Mais Forte).

De qualquer das formas, uma vez que o DFS, como o nome indica, vai fazer uma pesquisa em profundidade, para redes de grande dimensão, este algoritmo não aparenta ser o mais apropriado, uma vez que se pode estender bastante, ou seja, este algoritmo deve ser usado em redes de pequena dimensão apenas. Se analisarmos as tabelas, podemos ver o impacto que aumentar o número máximo de ligações tem no tempo de resposta deste algoritmo.

Em relação aos outros dois algoritmos, podemos perceber que têm comportamentos muito parecidos, mesmo nos resultados que apresentam. O best first, para os exemplos que foram testados, revelou ser o algoritmo que demorou menos tempo a gerar uma solução. Contudo, em comparação com a solução do A* revelou ser, na maior parte dos casos, uma solução menos forte. O critério para usar um destes dois algoritmos deve depender daquilo que é mais importante, se se pretender o melhor tempo possível, o best first deverá ser o escolhido, no caso de se querer uma solução mais próxima da melhor possível, o A* apresenta melhor características.

Tendo em conta a implementação da função multicritério, é agora possível relacionar as forças de ligação com as forças de relação e assim obter um valor que reflete essa combinação. Com este valor, temos um cálculo do custo mais aproximado da realidade, e assim é possível gerar soluções mais próximas daquilo que se pretende.