Sistemas de Representação de Conhecimento e Raciocínio

MIEI - 3º ANO - 2º SEMESTRE
UNIVERSIDADE DO MINHO

TRABALHO PRÁTICO INDIVIDUAL

Hugo Moreira
A43148

Braga, 9 de Outubro de 2022

Resumo

O presente documento diz respeito ao trabalho prático proposto na unidade curricular de Sistemas de Representação de Conhecimento e Raciocınio da Universidade do Minho.

Este trabalho tem o objetivo de motivar para a resolução de problemas de pesquisa informada e não informada usando a linguagem de programação em lógica PROLOG.

Ao longo do presente relatório serão apresentadas as formas de conhecimento adotadas e as caracterısticas e funcionalidades do sistema, assim como também serão descritas as estratégias utilizadas para a concretização das mesmas.

De uma perspetiva geral, considero que a realização deste exercício foi relativamente bem sucedida, visto que penso ter cumprido com todos os requisitos mínimos propostos.

Conteúdo

1. Introdução

No âmbito da disciplina de Sistemas de Representação de Conhecimento e Raciocínio foi-nos proposta, a demonstração de funcionalidades subjacentes à resolução de problemas de pesquisa.

O sistema de representação de conhecimento e raciocínio desenvolvido deverá permitir:

- Representar conhecimento, como paragens e adjacentes;
- Representar métodos de resolução de problemas de pesquisa informada e não informada, como o caso de procura em profundidade ou largura e estrela ou gulosa, respetivamente. tipos estudados;
- Desenvolver um sistema que usufrua destes algoritmos para a resolução das capacidades que são pretendidas pelo sistema.

Este trabalho tem o objetivo de motivar para a resolução de problemas de pesquisa informada e não informada usando a linguagem de programação em lógica PROLOG.

2. Descrição do Trabalho e Análise de Resultados

2.1 Descrição do Sistema

Para o desenvolvimento do sistema de representação de conhecimento e raciocínio com capacidade para caracterizar um universo de transportes, foi considerado a seguinte representação:

- $paragem : \#gid, latitude, longitude, estado, tipo, publicidade, operadora, carreira, cod_rua, rua, freguesia$ $\{V, F\}$
- $adjacente : \#nodo1, nodo2, carreira \rightarrow \{V, F\}$

```
:-dynamic paragem/11
:-dynamic adjacente/3
```

2.2 Predicados

De seguida iremos expôr os predicados usados neste trabalho.

2.2.1 Predicado paragem/11

O predicado parragem tem como finalidade caraterizar uma entiade cuja se encontra situada numa rua em que podem passar várias carreiras, esta tem um estado da sua condição, se tem abrigo, publicidade, localização, etc:

```
paragem(79,-107011.55,-95214.57,Bom,fechado_dos_lados,yes,vimeca,[01],103,Rua_Damiaode_Gois,Algoes_Linda_a_Velha_e_Cruz_Quebrada_Dafundo).
paragem(593,-103777.02,-94637.67,Bom,sem_abrigo,no,vimeca,[01],300,Avenida_dos_Cavaleiros,Carnaxide_e_Queijas).
paragem(499,-103758.44,-94393.36,Bom,fechado_dos_lados,yes,vimeca,[01],300,Avenida_dos_Cavaleiros,Carnaxide_e_Queijas).
paragem(494,-106803.2,-96265.84,Bom,sem_abrigo,no,vimeca,[01],389,Rua_SaoJoaode_Deus,Algoes_Linda_a_Velha_e_Cruz_Quebrada_Dafundo).
paragem(480,-106757.3,-96240.22,Bom,sem_abrigo,no,vimeca,[01],389,Rua_SaoJoaode_Deus,Algoes_Linda_a_Velha_e_Cruz_Quebrada_Dafundo).
paragem(957,-106911.18264993647,-96261.15727273725,Bom,sem_abrigo,no,vimeca,[01],399,Escadinhas_da_Fonte_da_Maruja,Algoes_Linda_a_Velha_e_Cruz_Quebrada_Dafundo).
paragem(366,-106021.37,-96684.5,Bom,fechado_dos_lados,yes,vimeca,[01],411,Avenida_Dom_Pedro_V,Algoes_Linda_a_Velha_e_Cruz_Quebrada_Dafundo).
paragem(365,-106016.12,-96673.87,Bom,fechado_dos_lados,yes,vimeca,[01],411,Avenida_Dom_Pedro_V,Algoes_Linda_a_Velha_e_Cruz_Quebrada_Dafundo).
paragem(357,-105236.99,-96664.4,Bom,fechado_dos_lados,yes,vimeca,[01],1279,Avenida_Tomas_Ribeiro,Algoes_Linda_a_Velha_e_Cruz_Quebrada_Dafundo).
paragem(336,-105143.57,-96690.32,Bom,fechado_dos_lados,yes,vimeca,[01],1279,Avenida_Tomas_Ribeiro,Algoes_Linda_a_Velha_e_Cruz_Quebrada_Dafundo).
paragem(334,-105336.07,-96668.68,Bom,fechado_dos_lados,yes,vimeca,[01],1279,Avenida_Tomas_Ribeiro,Algoes_Linda_a_Velha_e_Cruz_Quebrada_Dafundo).
```

2.2.2 Predicado adjacente/3

O predicado adjacente tem como finalidade representar um caminho entre paragens, contendo também a informação relativamente à carreira que efetua essa transição de nodos/paragens:

```
adjacente(183,791,01).
adjacente(791,595,01).
adjacente(595,182,01).
adjacente(182,499,01).
adjacente(499,593,01).
adjacente(593,181,01).
adjacente(181,180,01).
```

2.3 Algoritmos de resolução

2.3.1 Procura em Profundidade

Embora possa existir diferentes tipos de algoritmos de pesquisa informada e não informada, como é o caso das pesquisas em largura ou profundidade e gulosa ou estrela. Decidi utilizar o algoritmo de pesquisa em profundidade.

Posto isto, quando se pretender efetuar a procura/existência de um caminho entre um ponto de origem e um ponto de destino, iremos utilizar o predicado resolvepp:

```
resolve_pp(Nodo, Dst, [Nodo|Caminho]) :-
    profundidadeprimeiro(Nodo, Dst, Caminho).


profundidadeprimeiro(Dst, Dst, [Caminho]).

profundidadeprimeiro(Nodo, Dst, [ProxNodo|Caminho]) :-
    adjacente(Nodo, ProxNodo,_),
    profundidadeprimeiro(ProxNodo, Dst, Caminho).
```

2.3.2 Filtrar por Operadoras

Existe a possibilidade de efetuar a pesquisa, utilizando apenas determinadas operadoras, é efetuada a verificação ao mesmo tempo que o caminho está a ser calculado:

```
resolve_operadoras(Nodo, Dst,L, [Nodo|Caminho]) :-
    operadoras(Nodo, Dst, L, Caminho).

operadoras(Dst, Dst, B, [Caminho]) :- findall((O),paragem(Dst,_,_,_,_,_,O,_,_,_,_),L),
                                      conf_list(B,L).

operadoras(Nodo, Dst, B, [ProxNodo|Caminho]) :-
    adjacente(Nodo, ProxNodo,_),
    findall((O),paragem(Nodo,_,_,_,_,_,O,_,_,_,_),L),
    conf_list(B,L),
    operadoras(ProxNodo, Dst, B, Caminho).
```

2.3.3 Filtrar excluindo determindas Operadoras

Semelhante ao pretendido no predicado referido anteriormente, apenas a funcionalidade é o oposto da anterior. Referindo isto a única alteração efetuada em relação ao predicado anterior foi mesmo a utilização do predicado nao para obter o oposto da anterior:

```
resolve_sem_operadoras(Nodo, Dst,L, [Nodo|Caminho]) :-
    sem_operadoras(Nodo, Dst, L, Caminho).

sem_operadoras(Dst, Dst, B, [Dst|Caminho]) :- findall((O),paragem(Dst,_,_,_,_,_,O,_,_,_,_),L),
                                              nao(conf_list(B,L)).

sem_operadoras(Nodo, Dst, B, [ProxNodo|Caminho]) :-
    adjacente(Nodo, ProxNodo,_),
    findall((O),paragem(Nodo,_,_,_,_,_,O,_,_,_,_),L),
    nao(conf_list(B,L)),
    sem_operadoras(ProxNodo, Dst, B, Caminho).
```

2.3.4 Calcular caminho determinando a paragem com maior número de carreiras

À medida que é efetuado o cálculo do caminho é feita a verificação e atualização do maior número de carreiras existentes nas paragens até então cálculadas:

```
resolve_maior(Nodo, Dst, [Nodo|Caminho], M) :-
    maior(Nodo, Dst, Caminho, M).


maior(Dst, Dst,[Caminho],0).

maior(Nodo, Dst, [ProxNodo|Caminho],P) :-
    adjacente(Nodo, ProxNodo,_),
    maior(ProxNodo, Dst, Caminho,M),
    findall((C),paragem(Nodo,_,_,_,_,_,_,C,_,_,_),L),
    comp(L,N),
    N>=M,
    P is N.

maior(Nodo, Dst, [ProxNodo|Caminho],P) :-
    adjacente(Nodo, ProxNodo,_),
    maior(ProxNodo, Dst, Caminho,M),
    findall((C),paragem(Nodo,_,_,_,_,_,_,C,_,_,_),L),
    comp(L,N),
    M>N,
    P is M.
```

2.3.5 Pesquisa com o menor número de paragens possíveis

O seguinte predicado efetua a procura de todos os caminhos possíveis, para dois determinados nodos. Calculando após o melhor caminho de todos os caminhos que deram como resultado a primeira pesquisa:

```
todos(A,B,L) :- findall((S),resolve_pp(A,B,S),L).

melhor(A,B,L,Custo) :- findall((S,C),(resolve_pp(A,B,S), length(S,C)),L), minimo(L,(S,Custo)).

minimo([(P,X)],(P,X)).
minimo([Px,X|L],(Py,Y)) :- minimo(L,(Py,Y)), X>Y.
minimo([Px,X|L],(Px,X)) :- minimo(L,(Py,Y)), X=<Y.
```

2.3.6 Procura caminhos apenas com paragens que contêm publicidade

Semelhante a predicados referidos anteriormente, este predicado efetua o calcula do caminho possível fazendo a verificação aquando do cálculo se a paragem contêm publicidade ou não:

```
resolve_pub(Nodo, Dst,L, [Nodo|Caminho]) :-
    pub(Nodo, Dst, L, Caminho).

pub(Dst, Dst, B, [Caminho]) :- findall((O),paragem(Dst,_,_,_,_,P,_,_,_,_,_),L),
                               pertence(B,L).


pub(Nodo, Dst, B, [ProxNodo|Caminho]) :-
    adjacente(Nodo, ProxNodo,_),
    findall((O),paragem(Nodo,_,_,_,_,P,_,_,_,_,_),L),
    pertence(B,L),
    pub(ProxNodo, Dst, B, Caminho).
```

2.3.7 Procura caminhos apenas com paragens abrigadas

Semelhante a predicados referidos anteriormente, este predicado efetua o calcula do caminho possível fazendo a verificação aquando do cálculo se a paragem contêm qualquer tiupo de abrigo:

```
resolve_abrigo(Nodo, Dst,L, [Nodo|Caminho]) :-
    abrigo(Nodo, Dst, L, Caminho).

abrigo(Dst, Dst, B, [Caminho]) :- findall((A),paragem(Dst,_,_,_,A,_,_,_,_,_,_),L),
                                  nao(pertence(B,L)).


abrigo(Nodo, Dst, B, [ProxNodo|Caminho]) :-
    adjacente(Nodo, ProxNodo,_),
    findall((A),paragem(Nodo,_,_,_,A,_,_,_,_,_,_),L),
    nao(pertence(B,L)),
    abrigo(ProxNodo, Dst, B, Caminho).
```

3. Conclusão

De uma perspetiva geral, considero que a realização deste exercıcio foi relativamente sucedida, visto que penso ter cumprido com quase todos os requisitos propostos.

Após finalizar este trabalho, posso afirmar que o nosso sistema poderia conter mais algoritmos de pesquisa, incluindo não informada que não chegou a ser implementada.

Este trabalho permitiu consolidar melhor os conceitos de adquiridos ao longo do semestre da cadeira de Sistemas de Representação de Conhecimento e Raciocínio, com o auxílio da linguagem de programação lógica PROLOG.