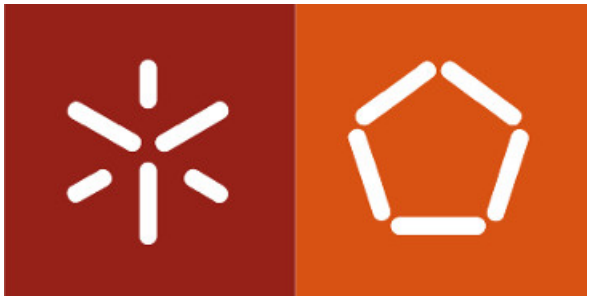

Universidade do Minho

Mestrado Integrado em Engenharia Informática

Sistemas de Representação de Conhecimento e Raciocínio

Trabalho Individual - Métodos de Resolução de Problemas e de Procura

---

Shahzod Yusupov - A82617

July 14, 2020

Resumo

O objetivo deste trabalho passa pela utilização de Programação em Lógica, usando a linguagem de programação PROLOG, no âmbito de métodos de Resolução de Problemas e no desenvolvimento de algoritmos de pesquisa.

O tema centra-se no mapa de Portugal, mais concretamente nas suas cidades e é proposto a criaçåo de um sistema, utilizando algortimos de pesquisa, que permita obter informações acerca dos caminhos e ligações existentes entre as mesmas.

Para a realização deste trabalho, foi necessário, numa primeira fase, a criação das ligações entra as cidades existentes seguindo um critério que será explicado posteriormente e só depois a implementação dos algortimos de pesquisa informada e não-informada, lecionados na Unidade Curricular.

CONTENTS

1 Introdução

Os algortimos de procura são bastante utilizados em várias áreas para a resolução de determinados problemas, e a Inteligência Artificial é uma delas. Este tipo de problemas, designados de problemas de procura, são normalmente resolvidos por passos apresentando vários estados ao longo da sua resolução, passos esses que dependem do algortimo a ser aplicado.

Assim, de forma a estudar/explorar esta problemática, é proposto a implementação de um sistema que se baseia neste tema, tendo como principal foco as cidades de Portugal e as ligações entre estas.

2 Preliminares

Previamente ao início do desenvolvimento do exercício em questão, foi necessário o estudo dos conteúdos que nos permitem, a implementação de um sistema que vá de encontro ao que é pedido.

Para além disso, foi necessário um pré-processamento dos dados fornecidos, visto que estes não estavam prontos a ser utilizados e a criação de uma Base de Conhecimento.

Assim é importante realçar alguns conceitos importantes dentro desta temática de modo a percebermos depois o funcionamento dos algortimos

- **Estado Inicial** tal como o nome diz, corresponde à descrição inicial do agente, ou seja, nesta temática podemos considerar a cidade onde queremos que o algoritmo tenha início.
- **Estado objetivo** estado que queremos alcançar após a implementação do algortimo, ou seja, nesta temática corresponde à cidade onde queremos chegar.
- **Ações** correspondem ao conjunto de ações que se podem tomar em cada estado.
- **Modelo d e Transição** Que estado resulta da execução de uma determinada ação em um determinado estado?
- **Custo do Caminho** função que determina o custo associado ao caminho

Em relação ao desempenho dos algortimo de pesquisa, este pode ser avaliado através dos seguintes critérios:

- **Completude** Está garantido que encontra a solução?
- **Complexidade no Tempo** Quanto tempo demora a encontrar a solução?
- **Complexidade no Espaço** Quanta memória necessita para fazer a pesquisa?
- **Otimalidade** Encontra a melhor solução?

3 Descrição do Trabalho e Análise de Resultados

Com o intuito de uma melhor exposição sobre o trabalho realizado, este mesmo foi dividdiO em partes, representadas da seguinte forma:

- Descrição da Base de Conhecimento , as estruturas usadas e o processamento dos dados do ficheiro fornecido;
- Descrição dos algoritmos desenvolvidos para a resolução dos problemas;
- Aplicação dos algoritmos e explicação do raciocínio por detrás de cada um
- Conclusões obtidas e observações feitas sobre o trabalho desenvolvido

3.1 Base de Conhecimento

Tendo em conta que o sistema representação de conhecimento, que estamos a analisar, consiste nas cidades de Portugal e as ligações entre estas, podemos caracterizar utilizando duas fontes de conhecimento, sendo elas: **cidade** e **g** que representa um grafo.

De seguida é apresentada a definição de cada um destes predicados.

- **cidade:** ID, city, lat, lng, admin, capital, monumento, hotel, loja, ligacoes -> {V,F,D}

  A cidade é caraterizado por um identificador único (ID), um nome(city), as suas coordenadas geográficas(lat, lng), a cidade que a administra(admin), se é capital ou não (capital), se possui monumentos(monumentos), se tem hoteis (hotel),as lojas que tem (loja) e as ligações com as outras cidades(ligacoes).

- **g: (graph)** -> {V,F,D}

  O grafo representa as relações entre as várias cidades, sendo que neste problema em específico interessa representar a possibilidade de viajar de uma cidade

para outra. Tal como lecionado nas aulas da Unidade Curricular, foi usado a entidade graph para representar os grafos conectados, sendo que possui na sua constituição 2 componentes que o caracterizam:os nodos (que correspondem às cidades) e as arestas (que correspondem ás ligações entre as cidades), esta última correspondendo a uma entidade arco que possui o nodo origem e nodo destino da ligação.

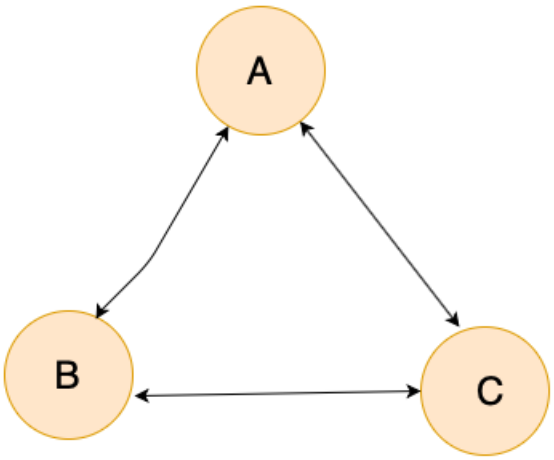

Figure 1: Exemplo de representação de um grafo

Assim, pegando nesta representação gráfica podemos passá-la para notação prolog: **graph([A,B,C], [arco(A,B), arco(B,A), arco(A,C), arco(C,A), arco (B,C), arco(C,B)])**, em que a primeira componente de graph corresponde à lista de nodos existentes e a segunda componente à lista de arestas existentes.

3.1.1 Pré-Processamento dos Dados

Foi fornecido um ficheiro xsls com todas as cidades, porém este ficheiro encontrava-se incompleto, tendo sido adicionado, através de uma script em Python utilizando a biblioteca Pandas, alguns campos que complementaram o predicado cidade.

Assim para além dos campos já presentes foram adicionados os campos : monumento,loja,hotel e ligacoes, tornando assim o dataset pronto a ser utilizado:

```
%%cidade(id,cidade,lat,long,distrito,capital,monumento,hotel,ligaçces)

cidade(1,'Lisbon',38.716667,-9.133333,'Lisboa','primary','Yes','Yes','Lidl',['Leiria', 'Santarem', 'Setubal']).
cidade(2,'Picotos',41.192402,-8.619816,'Porto','minor','Yes','Yes','Lidl',['Braga', 'Vila Real', 'Aveiro', 'Viseu']).
cidade(3,'Braga',41.550323,-8.420052,'Braga','admin','Yes','Yes','Lidl',['Porto', 'Viana do Castelo', 'Vila Real']).
cidade(5,'Setubal',38.533333,-8.900000,'Setubal','admin','Yes','Yes','Lidl',['Evora', 'Beja', 'Santarem', 'Lisboa']).
cidade(6,'Copeira',40.176915,-8.424018,'Coimbra','minor','Yes','Yes','Lidl',['Aveiro', 'Viseu', 'Leiria', 'Castelo Branco', 'Guarda']).
cidade(8,'Portimao',37.136636,-8.539796,'Faro','minor','Yes','Yes','Minipreco',['Beja']).
cidade(9,'Evora',38.566667,-7.900000,'Evora','admin','Yes','Yes','Minipreco',['Setubal', 'Santarem', 'Portalegre', 'Beja']).
cidade(10,'Aveiro',40.644269,-8.645535,'Aveiro','admin','Yes','Yes','Minipreco',['Viseu', 'Porto', 'Coimbra']).
cidade(11,'Leiria',39.747724,-8.804995,'Leiria','admin','Yes','Yes','Minipreco',['Coimbra', 'Castelo Branco', 'Santarem', 'Lisboa']).
cidade(12,'Faro',37.019367,-7.932229,'Faro','admin','Yes','Yes','Minipreco',['Beja']).
cidade(13,'Beja',38.015064,-7.863227,'Beja','admin','Yes','Yes','Minipreco',['Setubal', 'Evora', 'Faro']).
```

Figure 2: Excerto do dataset

Em relação às ligações entre as cidades, foi desenvolvido uma script,também em Python em que cada cidade minor só tem ligação direta com a sua capital, neste caso cidade admin e cada cidade admin tem ligação com os seus concelhos (cidades minor) e com as outras cidades admin.

4 Tipos de Algoritmos de Procura

De modo a resolver este trabalho é necessário ter em conta os vários tipos de algoritmo de procura existentes e ter conhecimento acerca do seu modo de funcionamento.

4.1 Pesquisa Não-Informada

Estes tipo de algoritmos são os mais conhecidos e talvez mais fáceis de perceber, caracterizando-se pelo facto do agente apenas ter informações sobre o nodo objetivo após alcançá-lo. Assim em cada estado o agente tem de ter em consideração todas as opções possíveis para alcançar o estado seguinte, usando apenas as indormações disponíveis na definição do problema.

4.1.1 Breadth- first search

Neste tipo de pesquisa todos os nós são expandidos primeiro, sendo uma pesquisa muito sistemática.

Normalmente demora muito tempo e sobretudo muito espaço, sendo que no geral só pequenos problemas podem ser resolvidos assim.

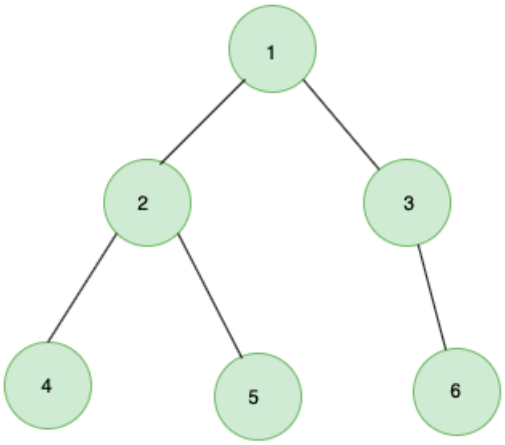

Figure 3: Breadth-first search

4.1.2 Depth- first search

Por outro lado este tipo de pesquisa consiste em expandir sempre um dos nós mais profundos da árvore, sendo necessário muito pouca memória tornando-se um bom algoritmo para problemas com muitas soluções.

Caso encontre um nodo que nao permita expansão e o nodo objetivo nao tiver sido alcançado, é efetuado um backtrack e a procura continua da mesma forma .

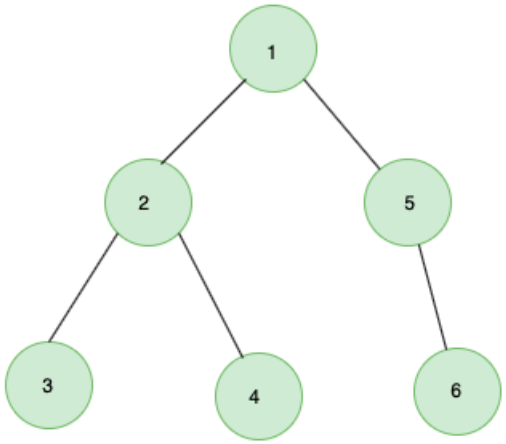

Figure 4: Depth-first search

4.2 Pesquisa Informada

Com o intuito de comparar com os algoritmos anteriores foram desenvolvidos algoritmos baseados em pesquisa informada. Estes são caracterizados por utilizarem informação do problema para evitar que o algoritmo de pesquisa fique " perdido vagueando no escuro".

Estes algoritmos usam heurísticas, que são estimativas de cada nodo ao nodo objetivo, sendo neste caso concreto a distância entre estes.

4.2.1 Pesquisa Gulosa

Caracterizado por expandir sempre o nodo que parece estar mais perto da solução, ou seja, nodo cuja estimativa é a mais curta em relação ao objetivo.

Este algoritmo mantém todos os nós na memória, tornando-se pesado e nem sempre encontra a solução ótima.

4.2.2 Pesquia A*

Caracterizado por combinar o custo desde o nodo inicial até ao atual mais a estimativa do custo do nodo atual ao nodo objetivo para a escolha do nodo a expandir.

Apesar também manter todos os nós em memória, este algortimo é ótimo e completo, escolhendo uma heurística admissível.

5 Aplicação dos Algortimos na resolução das queries

Assim sendo, as queries que foram propostas nesta temática são:

- Calcular um trajeto possível entre duas cidades;
- Selecionar apenas cidades, com uma determinada caraterística, para um determinado trajeto;
- Excluir uma ou mais caracteristicas de cidades para um percurso;
- Identificar num determinado percurso qual a cidade com o maior número de ligações;
- Escolher o menor percurso (usando o critério do menor número de cidades percorridas);
- Escolher o percurso mais rápido (usando o critério da distância);
- Escolher um percurso que passe apenas por cidades "minor";
- Escolher uma ou mais cidades intermédias por onde o percurso deverá obrigatoriamente passar.

De seguida irá ser explicado para cada query o raciocínio por detrás da sua resolução, bem como apresentação do resultado após a sua aplicação.

5.1 Calcular um trajeto entre dois pontos

Sendo esta query simples e sem restrições quanto aos caminhos, irão ser utilizados os algoritmos desenvolvidos nas aulas da Unidade Curricular, com adaptações de maneira a enquadrar no problema proposto. Assim após implementar os algoritmos os resultados foram os seguintes:

```
| ?- g(G), resolve_dfs(66,40,G,C), total_dist(C,D).
G = graph([1,2,3,4,5,6,7,8,9|...],[arco(148,66),arco(32,225),arco(5,157),arco(13,81),arco(241,15),arc
o(32,123),arco(285,11),arco(66,99),arco(...)|...]),
C = [66,11,17,9,21,15,16,14,18,10|...],
D = 0.009384789828834778 ?
yes
```

Figure 5: Depth-first search

```
| ?- g(G),resolve_bfs(66,40,G,C), total_dist(C,D).
G = graph([1,2,3,4,5,6,7,8,9|...],[arco(148,_A),arco(32,225),arco(5,157),arco(13,81),arco(241,15),arc
o(32,123),arco(285,11),arco(_A,99),arco(...)|...]),
C = [66,10,32,40],
D = 0.0012612052415451366 ?
yes
```

Figure 6: Breadth-first search

```
| ?- g(G), resolve_gulosa(66, 40, G, C).
G = graph([1,2,3,4,5,6,7,8,9|...],[arco(148,_A),arco(32,225),arco(5,157),arco(13,81),arco(241,15),arc
o(32,123),arco(285,11),arco(_A,99),arco(...)|...]),
C = [66,10,32,40]/0.0012612052415451366 ?
yes
```

Figure 7: Pequisa Gulosa

```
| ?- g(G), resolve_aestrela(66, 40, G, C).
G = graph([1,2,3,4,5,6,7,8,9|...],[arco(148,_A),arco(32,225),arco(5,157),arco(13,81),arco(241,15),arc
o(32,123),arco(285,11),arco(_A,99),arco(...)|...]),
C = [66,10,32,40]/0.0012612052415451366 ?
yes
```

Figure 8: Pesquisa A*

5.2 Selecionar apenas cidades, com uma determinada caraterística, para um determinado trajeto

Para esta query fora utilizados diferentes funções, que apenas diferem das anteriores na forma como são escolhidos os nodos adjacentes, dependendo da característica a avaliar. Assim, foi desenvolvido para escolher um trajeto que contenha cidades com determinadas lojas e também para trajetos cujas cidades apresentem ou não monumentos.

```
| ?- g(G),resolve_dfs_loja(16,17,G,['Minipreco'],C).
G = graph([1,2,3,4,5,6,7,8,9|...],[arco(148,16),arco(32,225),arco(5,157),arco(13,81),arco(241,15),arc
o(32,123),arco(285,11),arco(16,99),arco(...)|...]),
C = [16,11,17] ?
yes
```

```
| ?- g(G),resolve_dfs_loja(16,17,G,['Continente'],C).
no
```

Podemos verificar que no trajeto da cidade 16 para a cidade 17 é possível encontrar cidades que tenham a loja Minipreco ,porém já para a loja Continente não , significando isto que não existe nenhum trajeto da cidade 16 para a cidade 17 em que todas as cidades tenham a loja Continente.

Já em relação aos monumentos os resultados obtidos foram os seguintes:

```
| ?- g(G), resolve_dfs_monumento(66,40,G,S).
G = graph([1,2,3,4,5,6,7,8,9|...],[arco(148,66),arco(32,225),arco(5,157),arco(13,81),arco(241,15),arc
o(32,123),arco(285,11),arco(66,99),arco(...)|...]),
S = [66,11,17,9,21,15,16,227,18,10|...] ?
yes
```

Em que no trajeto da cidade 66 para a cidade 40 todas aquelas cidades intermédias possuem monumentos.

5.3 Excluir uma ou mais caracteristicas de cidades para um percurso

Este algoritmo é muito semelhante ao anterior, mudando aenas a forma como são escolhidos os nodos adjacentes, desta vez excluindo as caracteristicas anteriores.

```
| ?- g(G), resolve_bfs_s_lojas(66, 40, G,['Minipreco'],C).
G = graph([1,2,3,4,5,6,7,8,9|...],[arco(148,_A),arco(32,225),arco(5,157),arco(13,81),arco(241,15),arc
o(32,123),arco(285,11),arco(_A,99),arco(...)|...]),
C = [66,99,40] ?
yes
| ?- g(G), resolve_bfs_s_lojas(66, 40, G,['Continente'],C).
G = graph([1,2,3,4,5,6,7,8,9|...],[arco(148,_A),arco(32,225),arco(5,157),arco(13,81),arco(241,15),arc
o(32,123),arco(285,11),arco(_A,99),arco(...)|...]),
C = [66,11,40] ?
yes
| ?-
```

Figure 9: Breadth-first search

```
| ?- g(G), resolve_dfs_s_loja(66, 40, G,['Continente'],C).
G = graph([1,2,3,4,5,6,7,8,9|...],[arco(148,66),arco(32,225),arco(5,157),arco(13,81),arco(241,15),arc
o(32,123),arco(285,11),arco(66,99),arco(...)|...]),
C = [66,11,17,9,21,15,16,14,18,10|...] ?
yes
| ?- g(G), resolve_dfs_s_loja(66, 40, G,['Continente','Minipreco'],C).
no
| ?- g(G), resolve_dfs_s_loja(66, 40, G,['Minipreco'],C).
```

Figure 10: Depth-first search

5.4 Identificar num determinado percurso qual a cidade com o maior número de ligações;

O raciocínio por detrás deste algoritmo foi guardar o número de ligacoes de cada cidade presente no trajeto escolhido e no fim ordená-lo por (numero de ligacoes, cidade).

```
| ?- top_N_cidades([66,10,32,40],R,2).
R = [4-40,4-32] ?
yes
| ?- top_N_cidades([66,10,32,40],R,4).
R = [4-40,4-32,3-66,3-10] ?
yes
```

5.5 Escolher o menor percurso (usando o critério do menor número de cidades percorridas)

Nesta query é pedido o percurso com menor número de cidades percorridas e para tal a solução passa pela utilização da função findall para encontrar todas as soluções possíveis de caminhos entre 2 nodos,associando um custo a cada (que é o número de cidades que o caminho contém) e devolvendo o caminho com o menor custo.

```
| ?- g(G), minimo_cidades(66,40, G, S, C).
G = graph([1,2,3,4,5,6,7,8,9|...],[arco(148,_A),arco(32,225),arco(5,157),arco(13,81),arco(241,15),arc
o(32,123),arco(285,11),arco(_A,99),arco(...)|...]),
S = [66,10,32,40],
C = 4 ?
yes
```

Como podemos verificar da cidade 66 para a cidade 40 o menor trajeto é composto por 4 cidades e isto é facil de confirmar pois ambas as cidades 66 e 40 são cidades

minor, logo apenas têm ligação com a sua capital, portanto para chegar à cidade 40 é necessário primeiro passar pela capital da cidade 66 (10) e ainda pela capital da cidade 40 (32), chegando depois à cidade 40.

5.6 Escolher o percurso mais rápido (usando o critério da distância)

Muito semelhante ao algortimo anterior em que apenas muda o critério para a escolha do menor percurso, sendo neste caso a distância.

```
| ?- g(G), menor_dist(66,40,G,S,C).
G = graph([1,2,3,4,5,6,7,8,9|...],[arco(148,_A),arco(32,225),arco(5,157),arco(13,81),arco(241,15),arco(32,123),arco(285,11),
arco(_A,99),arco(...)|...]),
S = [66,10,32,40],
C = 0.0012612052415451366 ?
yes
```

5.7 Escolher um percurso que passe apenas por cidades "minor"

Esta query pede um percurso que passe apenas por cidades minor, porém da maneira como foram feitas as ligações nao é possivel um percurso assim, razão pela qual mudei "minor" para "admin" pois o algoritmo é o mesmo .

```
| ?- g(G), resolve_dfs_minor(3,10, G, C).
G = graph([1,2,3,4,5,6,7,8,9|...],[arco(148,3),arco(32,225),arco(5,157),arco(13,81),arco(241,15),arco(32,123),arco(285,11),a
rco(3,99),arco(...)|...]),
C = [3,11,17,9,21,15,16,14,18,10] ?
yes
```

Para mostrar que este algoritmo funciona , alterei no dataset a cidade 21 que era "admin" para "minor" e como resultado obtemos o seguinte percurso, em que a cidade 21 já nao faz parte deste.

```
| ?-  g(G), resolve_dfs_minor(3,10, G, C).
G = graph([1,2,3,4,5,6,7,8,9|...],[arco(148,3),arco(32,225),arco(5,157),arco(13,81),arco(241,15),arco(32,123),arco(285,11),a
rco(3,99),arco(...)|...]),
C = [3,11,17,15,16,14,18,10] ?
yes
```

```
| ?- g(G), resolve_bfs_minor(3,10,G,S).
G = graph([1,2,3,4,5,6,7,8,9|...],[arco(148,_A),arco(32,225),arco(5,157),arco(13,81),arco(241,15),arco(32,123),arco(285,11),
arco(_A,99),arco(...)|...]),
S = [3,32,10] ?
yes
```

6 Conclusão

O desenvolvimento deste trabalho prático permitiu consolidar os conhecimentos adquiridos ao longo do semestre e sobretudo pôr em prática os mesmos.

Em relação aos algoritmos desenvolvidos, já estava familiarizado com os do tipo pesquisa não informada, visto que já foram abordados em outras Unidades Curriculares , mas foi interessante implementá-los e também aos do tipo pesquisa informada neste contexto específico. Também a manipulação de grafos fez com que este tema se tornasse desafiante e de um modo geral gostei bastante de realizar este trabalho.

Em suma, penso que a maioria dos objetivos a atingir com a realização deste trabalho foram cumpridos, apesar de não ter conseguido completar todos os requisitos.

REFERENCES

[1] Russell and Norvig (2009), *A Modern Approach, 3rd edition*,

[2] Maria João Frade, *Lógica Computacional - PROLOG*, Departamento Informática, Univerisdade do Minho;

[3] César Analide, Paulo Novais, José Neves, *Sugestões para a Redacção de Relatórios Técnicos*, Departamento Informática, Universidade do Minho;