Instituto Politécnico de Viseu

Escola Superior de Tecnologia e Gestão de Viseu

Departamento de Informática

Relatório do Trabalho Final

Gestão de Campeonato de Futebol

Inteligência Artificial

Orientadores:

Filipe Pinto

Realizado por:

Ana Catarina Reis, 14666

David Silva, 14016

Filipe Ferreira, 14017

Viseu, 2018

Instituto Politécnico de Viseu

Escola Superior de Tecnologia e Gestão de Viseu

Departamento de Informática

Relatório do Trabalho Final Licenciatura em Engenharia Informática

2017/2018

Inteligência Artificial

Orientadores:

Filipe Pinto

Realizado por:

Ana Catarina Reis, 14666

David Silva, 14016

Filipe Ferreira, 14017

Viseu, 2018

Índice

Índice de Figuras

1. Introdução

O presente projeto foi desenvolvido no âmbito da unidade curricular de Inteligência Artificial, da Licenciatura de Engenharia Informática, na Escola Superior de Tecnologia e Gestão de Viseu.

Consiste no desenvolvimento, em *Python*, de um agente resolvedor de problemas com a capacidade de procurar um caminho entre duas cidades diferentes, sendo que este caminho deve ser desenvolvido de acordo com diferentes estratégias.

No segundo capítulo do presente relatório, irá ser feito uma breve apresentação e do que trata a tecnologia *Python*, enquanto que, no capítulo 3, irá ser explicado o que se fez na fase de desenvolvimento do projeto.

Por fim, no capítulo 4, irão ser apresentadas as conclusões tiradas após a realização do todo o trabalho.

2. Python

Neste capítulo irá ser feito uma apresentação da *tecnologia Python*, utilizada ao longo do desenvolvimento do projeto/trabalho em questão.

A linguagem de programação *Python* foi criada em 1991, por *Guido van Rossum*, tendo como objetivos principais a produtividade e a legibilidade, ou seja, foi criada de modo a produzir um bom código, sendo também fácil de manter de uma forma rápida.

Tem características como o uso reduzido de caracteres especiais e de palavras chaves voltadas para a compilação, a marcação por blocos, o uso de gerenciador automático de memória, o suporte de diversos padrões de programação e a biblioteca padrão extensa.

É uma linguagem multiplataforma, sendo isto uma das suas principais vantagens, pois programas escritos numa plataforma poderão, na sua maioria e sem problemas, ser executados noutras diferentes.

Figura 1 - *Python*

3. Desenvolvimento do Projeto

Neste capítulo do trabalho irá ser explicado a sua estrutura e como foi desenvolvido. Foi estruturado em dois parâmetros: o tratamento das cidades e o tratamento dos métodos de procura em si.

3.1. Cidades

Neste parâmetro, foi criada uma classe Cidades onde se desenvolveram 3 dicionários, sendo eles: *cidadesLigadas*, *linhaReta* e *distanciaReta*.

O dicionário *cidadesLigadas* vai, posteriormente, ser utilizado nos métodos de procura Heurísticos (Informados), enquanto que o dicionário *linhaReta* vai ser utilizado nos métodos de procura cegos (não informados).

3.3.1. Dicionário *cidadesLigadas*

A Figura 2 apresenta o desenvolvimento do dicionário *cidadesLigadas* que se encontra dentro da função *Ligadas* (*def Ligadas (self)*).

Já dentro do dicionário em si (*cidadesLigadas = {}*), podem ver-se as várias linhas do mesmo, sendo que para cada uma delas, são indicadas as fronteiras do primeiro elemento da linha, isto é, por exemplo, para a cidade *Aveiro*, tem-se como fronteiras as cidades *Porto, Viseu, Coimbra e Leiria*, para a cidade *Évora*, tem-se como fronteiras as cidades *Lisboa, Santarém Portalegre, Setúbal, Beja* e *Castelo Branco*.

Aqui, para cada fronteira tem-se também a distáncia (peso) da primeira cidade de cada linha até à própria, ou seja, por exemplo, a distância (peso) entre *Aveiro* e *Porto* é 68, entre *Aveiro* e *Viseu* é 95, entre *Aveiro* e Coimbra é 68 e entre *Aveiro* e *Leiria* é 115.

```
def Ligadas(self):
    cidadesLigadas = {
        'Aveiro': {'Porto':68, 'Viseu': 95, 'Coimbra':68, 'Leiria': 115},
        'Braga': {'Viana do Castelo': 48, 'Vila Real': 106, 'Porto':53},
        'Braganca': {'Vila Real': 137, 'Guarda':202},
        'Beja': {'Evora': 78, 'Faro':152, 'Setubal':142},
        'Castelo Branco': {'Coimbra':159,'Guarda':106,'Portalegre':80,'Evora':203},
        'Coimbra': {'Viseu':96,'Leiria':67,'Aveiro':68,'Castelo Branco':159},
        'Evora': {'Lisboa':150,'Santarem':117,'Portalegre':131,'Setubal':103,'Beja':78,'Castelo Branco':203},
        'Faro':  {'Setubal': 249,'Lisboa':299,'Beja':152},
        'Guarda': {'Vila Real':157,'Viseu':85,'Braganca':202,'Castelo Branco':106},
        'Leiria': {'Lisboa':129,'Santarem':70,'Aveiro':115,'Coimbra':67},
        'Lisboa': {'Santarem':78,'Setubal':50,'Evora':150,'Faro':299,'Leiria':129},
        'Portalegre': {'Castelo Branco':80,'Evora':131},
        'Porto': {'Viana do Castelo':71,'Vila Real':116,'Viseu':133, 'Aveiro': 68,'Braga':53},
        'Santarem': {'Evora':117,'Leiria':70,'Lisboa':78},
        'Setubal': {'Beja':142,'Evora':103,'Faro':249,'Lisboa':50},
        'Viana do Castelo': {'Braga':48,'Porto':71},
        'Vila Real': {'Viseu':110,'Braga':106,'Braganca':137, 'Guarda':157,'Porto':116},
        'Viseu': {'Aveiro':95,'Coimbra':96,'Guarda':85,'Porto':133,'Vila Real':110}
    }
    return cidadesLigadas
```

Figura 2 - *cidadesLigadas*

3.3.2. Dicionário *linhaReta*

Na Figura 3, pode ver-se o desenvolvimento do dicionário *cidadesLigadas* que se encontra dentro da função *Ligadas* (*def Ligadas (self)*). Sendo que, funciona da mesma forma que o dicionário explicado anteriormente, tendo como única diferença a ausência dos pesos, pois não são necessários nos métodos cegos, onde este dicionário irá ser utilizado, como já foi referido.

```
    def Reta(self):
        linhaReta = {
            'Aveiro': ['Porto', 'Viseu', 'Coimbra', 'Leiria'],
            'Braga': ['Viana do Castelo', 'Vila Real', 'Porto'],
            'Braganca': ['Vila Real', 'Guarda'],
            'Beja': ['Evora', 'Faro', 'Setubal'],
            'Castelo Branco': ['Coimbra', 'Guarda', 'Portalegre', 'Evora'],
            'Coimbra': ['Viseu', 'Leiria', 'Aveiro', 'Castelo Branco'],
            'Evora': ['Lisboa', 'Santarem', 'Portalegre', 'Setubal', 'Beja', 'Castelo Branco'],
            'Faro': ['Setubal', 'Lisboa', 'Beja'],
            'Guarda': ['Vila Real', 'Viseu', 'Braganca', 'Castelo Branco'],
            'Leiria': ['Lisboa', 'Santarem', 'Aveiro', 'Coimbra'],
            'Lisboa': ['Santarem', 'Setubal', 'Evora', 'Faro', 'Leiria'],
            'Porto': ['Viana do Castelo', 'Vila Real', 'Viseu', 'Aveiro', 'Braga'],
            'Vila Real': ['Viseu', 'Braga', 'Braganca', 'Guarda', 'Porto'],
            'Viseu': ['Aveiro', 'Guarda', 'Porto', 'Vila Real', 'Coimbra'],
            'Setubal': ['Beja', 'Evora', 'Faro', 'Lisboa'],
            'Viana do Castelo': ['Braga', 'Porto'],
            'Santarem': ['Evora', 'Leiria', 'Lisboa'],
            'Portalegre': ['Castelo Branco', 'Evora']
        }
        return linhaReta
```

Figura 3 - *linhaReta*

3.3.3. Dicionário *distanciaReta*

Este dicionário (Figura 4), foi desenvolvido com o intuito de indicar a distância (peso) entre as cidades apresentadas e *Faro*, que é a cidade que é se assume como destino para os métodos de *Procura Sôfrega* e *A** (Heurísticos). Assim, pode dizer-se, por exemplo, que a distância (peso) entre *Aveiro* e *Faro* é de 366.

```
def DistanciaReta(self):
    distanciaReta = {
        'Aveiro': 366,
        'Braga': 454,
        'Bragança': 487,
        'Beja': 99,
        'Castelo Branco': 280,
        'Coimbra':319,
        'Evora': 157,
        'Faro': 0,
        'Guarda': 352,
        'Leiria': 278,
        'Lisboa': 195,
        'Portalegre': 228,
        'Porto': 418,
        'Santarem': 231,
        'Setubal': 168,
        'Viana do Castelo': 473,
        'Vila Real': 429,
        'Viseu': 363

    }
    return distanciaReta
```

Figura 4 - *distanciaReta*

3.2. Métodos de Procura

Neste parâmetro foi onde se desenvolveram os métodos de procura em si, sendo dois deles cegos (não informados) e os restantes dois heurísticos (informados). Irão ser explicados, primeiro, os métodos cegos (*profundidade primeiro* e *profundidade limitada*) e de seguida os métodos heurísticos (*procura sôfrega* e *A**).

3.2.1. Profundidade Primeiro

O método de procura *Profundidade Primeiro* tem como princípio a expansão do nó o mais profundo possível, sendo que não é considerado um método completo (encontra a solução quando esta existe), nem ótimo (encontra a solução de melhor qualidade aquando da existência de várias).

O código fonte do método de Profundidade Primeiro encontra-se na Figura 5.

```
def pesquisa_profundidade(grafo, inicio, fim):
    fronteira = []
    ant = []
    fronteira.append(inicio)
    visitados = []

    i = 1
    print('Nó inicial escolhido: '+ inicio)

    while True:
        print('Iteração - ', i)
        no_atual = fronteira.pop(0) #remove o primeiro da lista


        visitados.append(no_atual) #adiciona aos visitados
        print('Nó atual: ' + no_atual) # printa o no atual
        if(fronteira is not fronteira):
            ant.append(fronteira) #guarda todos aqueles que ainda nao foram explorados


        if no_atual == fim:
            print('*'*100)
            print('Chegamos a ' + fim)
            print('Locais Explorados: ')
            print(visitados)
            print('Total iterações: ', i)
            return
        else:
            print('Nó escolhido para abrir ' + no_atual)

        f = 0
        for no in grafo[no_atual]:
            if no not in visitados:
                fronteira.insert(f,no)
                f = f + 1

        print('-'*100)
        print('Na lista para visitar --> ', fronteira)
        i = i+1
```

Figura 5 - Profundidade Primeiro

O *output* deste método encontra-se no Anexo A.

3.2.2. Procura Sôfrega

O método de procura sôfrega tem como objetivo minimizar o custo estimado até se atingir o destino pretendido, assim, expande-se o nó mais próximo do destino pretendido primeiro. É possível estimar o custo com este método, mas não é possível determina-lo de modo preciso. O algoritmo escolhe o menor valor entre o nó atual e nó objetivo (Faro).

Não é um método ótimo, nem completo.

O código fonte está apresentado na Figura 6 e o *output* é o Anexo B.

```
def retorna_peso(no_inicio, no_para, tamanho):
    return tamanho.get((no_inicio, no_para)) if tamanho else 1


def procura_sofrega(graph, inicio, fim,reta):
    fronteira = PriorityQueue()
    fronteira.put((0,inicio))
    print('iteração 0')
    print(inicio)
    pos = 1
    valor = False
    while True:

        ucs_w, no_atual = fronteira.get()
        if valor:
            ucs_w = retorna_peso(no_atual, fim, reta)

        if no_atual == fim:
            print('-'*100)
            print('No escolhido ' + fim)
            print('Chegamos ao fim')
            return

        print('-'*100)
        print('Iteração '+str(pos))
        for no in graph[no_atual]:
            fronteira.put((retorna_peso(no, fim, reta), no))
        print(sorted(fronteira.queue))
        if fronteira.queue[0][1] is not fim:
            print('Para abrir ---> ' + str(fronteira.queue[0][1] + '.'))
        pos = pos+1
        valor = True
```

Figura 6 - Procura Sôfrega

3.2.3. A*

O método *A** é possível ser criado através da combinação do método de procura sôfrega e do método de custo uniforme, assim obtem-se um método completo e ótimo, usufruindo das vantagens de cada um dos algoritmos anteriores. Tem como principal objetivo tentar expandir o nó que pertence ao caminho com um menor custo associado.

Baseia-se em *f(n) = g(n) + h(n)*, sendo *f(n)* o custo estimado da solução mais barata passado por *n*, *g(n)* o custo do caminho e *h(n)* o custo estimado para o objetivo. Assim a melhor solução passa por escolher primeiro o nó com o valor mais baixo de *f*.

O código fonte desenvolvido encontra-se na Figura 7 e o seu *output* encontra-se nos anexos (Anexo C).

```
def a_star(graph, inicio, fim, peso, reta):
    fronteira = PriorityQueue()
    fronteira.put((0,inicio))
    print('iteração 0')
    print(inicio)
    pos = 1
    valor = False
    while True:

        ucs_w, no_atual = fronteira.get()
        if valor:
            ucs_w = ucs_w - retorna_peso(no_atual, fim, reta)

        if no_atual == fim:
            print('-'*100)
            print('No escolhido ' + fim)
            print('Chegamos ao fim')
            return

        print('-'*100)
        print('Iteração '+str(pos))
        for no in graph[no_atual]:
            fronteira.put((ucs_w + retorna_peso(no_atual, no, peso)+retorna_peso(no, fim, reta), no))
        print(sorted(fronteira.queue))
        if fronteira.queue[0][1] is not fim:
            print('Para abrir ---> ' + str(fronteira.queue[0][1] + '.'))
        pos = pos+1
        valor = True
```

Figura 7 - Método *A**

3.3. Menu

O código fonte do menu da aplicação encontra-se na Figura 8, enquanto que a sua apresentação é o Anexo D.

```
def menu(nome):
    sair = True
    while sair:
        Inicio = None
        Fim = None

        print(20*'^' + ' Opções ' + '^'*20)
        print('Opção 1: Pesquisa em Largura')
        print('Opção 2: Pesquisa em profundidade Limitada')
        print('Opção 3: Procura sôfrega (Faro)')
        print('Opção 4: A* (FARO)')
        print('Opcao: exit')
        opcao = input(nome + ' escolha uma opção: ')

        if opcao == "1":
            clear()
            print('Algoritmo - Pesquisa em Profundidade Primeiro')
            Inicio = input('Escolha o local de inicio: ')
            Fim = input('Escolha o destino: ')
            print(cidadesLigadas)
            pesquisa_profundidade(cidadesLigadas, Inicio, Fim)

        elif opcao == "2":
            clear()
            print('Algoritmo - Pesquisa em Profundidade Limitada')
            Inicio = input('Escolha o local de inicio: ')
            Fim = input('Escolha o destino: ')
            #pesquisa_em_profundidade_limitada(cidadesLigadas,Inicio,Fim,Nivel)
        elif opcao == "3":
            clear()
            print('Algoritmo - Custo uniforme ')
            Inicio = input('Escolha o local de inicio: ')
            print('Destino predefenido [Faro] ')
            Fim = 'Faro'
            custo_uniforme(cidadeReta,Inicio, Fim,pesos)
        elif opcao == "4":
            clear()
            print('Algoritmo - A* ')
            Inicio = input('Escolha o local de inicio: ')
            print('Destino predefenido [Faro] ')
            Fim = 'Faro'
            a_star(cidadesLigadas,Inicio, Fim, pesos,cidadeRetaFaro)

        elif opcao == "exit":
            clear()

            sair = False
        else:
            clear()
            print('Opcao errada escreva um numero de 1 a 4 ')
            print('Você escreveu: [' + opcao + ']')
```

Figura 8 - Menu

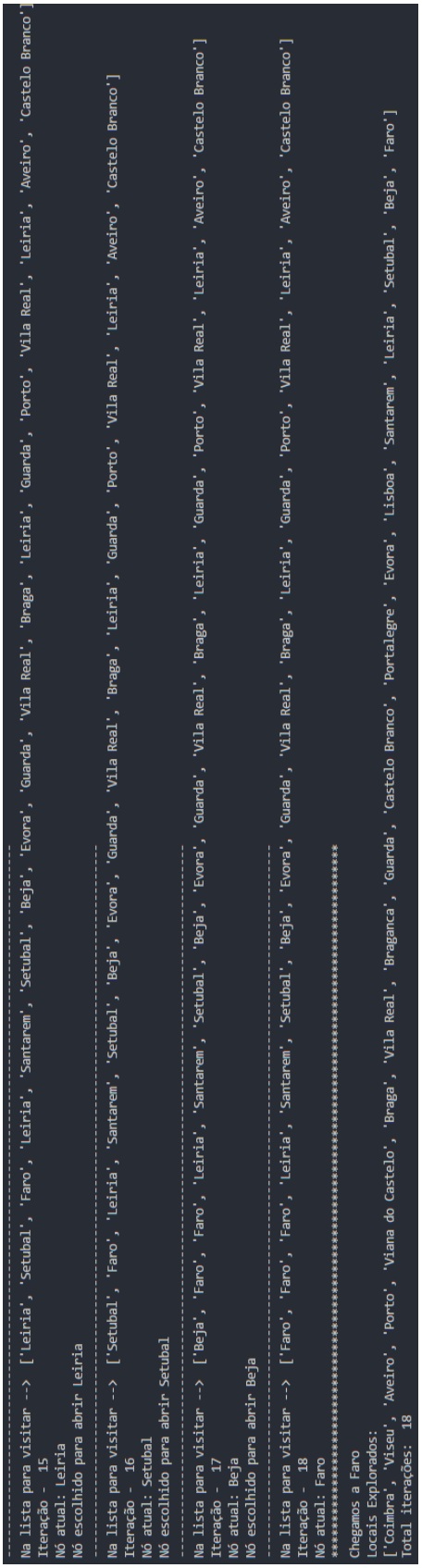
4. Conclusão

Como mencionado na introdução, neste capítulo será feita uma retrospetiva do trabalho realizado.

As principais dificuldades sentidas ao longo da realização do projeto foram no método de procura em profundidade limitada, acabando mesmo por não ser desenvolvido, sendo que todos os restantes foram desenvolvidos com sucesso, recorrendo a algumas pesquisas.

Em suma, conclui-se que os objetivos foram atingidos com sucesso, com entreajuda dos elementos do grupo.

Anexos



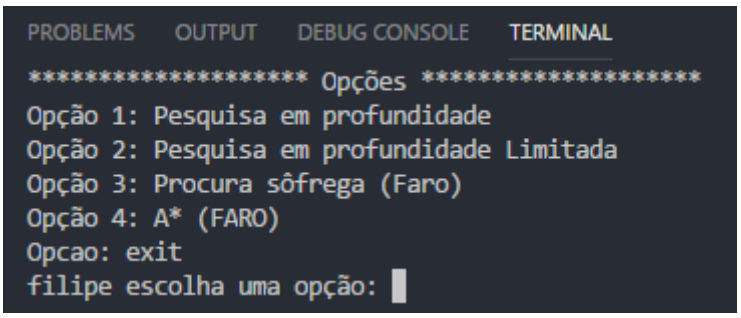
Anexo A - *Output* Procura Primeiro

```
Destino predefenido [Faro]
iteração 0
Coimbra
--------------------------------------------------------------------------------
Iteração 1
[('278, 'Leiria'), (280, 'Castelo Branco'), (363, 'Viseu'), (366, 'Aveiro')]
Para abrir ---> Leiria.
--------------------------------------------------------------------------------
Iteração 2
[(195, 'Lisboa'), (231, 'Santarem'), (280, 'Castelo Branco'), (319, 'Coimbra'), (363, 'Viseu'), (366, 'Aveiro'), (366, 'Aveiro')]
Para abrir ---> Lisboa.
--------------------------------------------------------------------------------
Iteração 3
[(0, 'Faro'), (157, 'Evora'), (168, 'Setubal'), (231, 'Santarem'), (231, 'Santarem'), (278, 'Leiria'), (280, 'Castelo Branco'), (319, 'Coimbra'), (363, 'Viseu'), (366, 'Aveiro'), (366, 'Aveiro')]
--------------------------------------------------------------------------------
No escolhido Faro
```

Anexo B - *Output* Procura Sôfrega

```
--------------------------------------------------------------------------------
Iteração 17
[(481, 'Leiria'), (484, 'Faro'), (484, 'Santarem'), (485, 'Leiria'), (487, 'Beja'), (491, 'Lisboa'), (495, 'Faro'), (495, 'Faro'), (502, 'Santarem'), (503, 'Evora'), (505, 'Santarem'), (506, 'Beja'), (506, 'Evora'), (507, '
Lisboa'), (510, 'Lisboa'), (511, 'Coimbra'), (514, 'Faro'), (519, 'Evora'), (522, 'Evora'), (524, 'Santarem'), (525, 'Evora'), (525, 'Lisboa'), (525, 'Setubal'), (526, 'Viseu'), (527, 'Evora'), (533, 'Guarda'
), (548, 'Aveiro'), (554, 'Porto'), (557, 'Aveiro'), (567, 'Evora'), (568, 'Aveiro'), (569, 'Coimbra'), (570, 'Aveiro'), (573, 'Castelo Branco'), (575, 'Castelo Branco'), (587, 'Coimbra'), (593, 'Viseu'), (595, 'Viseu'), (5
99, 'Castelo Branco'), (599, 'Lisboa'), (602, 'Santarem'), (603, 'Leiria'), (613, 'Portalegre'), (617, 'Guarda'), (622, 'Leiria'), (635, 'Vila Real'), (637, 'Coimbra'), (642, 'Setubal'), (647, 'Porto'), (664, 'Aveiro'), (68
2, 'Aveiro'), (737, 'Castelo Branco')]
Para abrir ---> Leiria.
--------------------------------------------------------------------------------
Iteração 18
[(484, 'Faro'), (484, 'Santarem'), (485, 'Leiria'), (487, 'Beja'), (491, 'Lisboa'), (495, 'Faro'), (495, 'Faro'), (502, 'Santarem'), (503, 'Evora'), (504, 'Santarem'), (505, 'Santarem'), (506, 'Beja'), (506, 'Evora'), (507,
 'Lisboa'), (510, 'Lisboa'), (511, 'Coimbra'), (514, 'Faro'), (514, 'Faro'), (519, 'Evora'), (522, 'Evora'), (524, 'Santarem'), (525, 'Evora'), (525, 'Lisboa'), (525, 'Setubal'), (526, 'Viseu'), (527, 'Evora'), (527, 'Lisbo
a'), (533, 'Guarda'), (548, 'Aveiro'), (554, 'Porto'), (557, 'Aveiro'), (567, 'Evora'), (568, 'Aveiro'), (569, 'Coimbra'), (570, 'Aveiro'), (573, 'Castelo Branco'), (575, 'Castelo Branco'), (587, 'Coimbra'), (589, 'Coimbra'
), (593, 'Viseu'), (595, 'Viseu'), (599, 'Castelo Branco'), (599, 'Lisboa'), (602, 'Santarem'), (603, 'Leiria'), (613, 'Portalegre'), (617, 'Guarda'), (622, 'Leiria'), (635, 'Vila Real'), (637, 'Coimbra'), (642, 'Setubal'),
 (647, 'Porto'), (664, 'Aveiro'), (682, 'Aveiro'), (684, 'Aveiro'), (737, 'Castelo Branco')]
--------------------------------------------------------------------------------
No escolhido Faro
Chegamos ao fim
```

Anexo C - *Output* do *A**



Anexo D - Menu da aplicação