**Universidade do Minho**
**Mestrado Integrado em Engenharia Informática**
**Scripting no Processamento de Linguagem Natural**

# *Python Interface for a Prolog Constraint Solver for Genealogy.*

João Gomes, A74033
Tiago Fraga, A74092

1 de Julho de 2019

# Conteúdo

# Capítulo 1

# Introdução

Com a realização do presente trabalho prático, pretende-se estudar, conceber e implementar uma interface de *constraint solvers* para problemas de genealogia.

Para o efeito, foi nos solicitado que a interface fosse desenvolvida utilizando a linguagem *Python* e os *constraint solvers* derivassem da linguagem *Prolog*. A partir deste momento, deparamo-nos com a primeira dificuldade do problema, ou seja, ligar e trabalhar duas linguagens bastante distintas.

O trabalho irá ser desenvolvido com o auxilio da ferramenta *Python Constraint*, cujo objetivo é oferecer soluções para problemas de satisfação de restrições em domínios finitos. Este tipo de problemas faz parte de uma classe de problemas que podem ser representados segundo variáveis seus domínios, bem como um conjunto de restrições que se pode aplicar a essas variáveis.

A forma de programar e resolver este tipo de problemas é ligeiramente diferente do que temos lidado nos últimos anos e de certa forma, altera o paradigma de programação de *Python*, uma vez que esta é uma linguagem imperativa, no entanto durante o desenvolvimento deste trabalho teve de ser usada de uma forma funcional, que ao contrario do imperativo onde implementamos todos os passos para chegar a um resultado final, apenas declaramos o objetivo e as suas restrições e o programa devolve todas as soluções possíveis para o fornecido.

Ao longo deste relatório iremos começar por abordar as componentes do pacote usado no desenvolvimento do projeto, bem como os algoritmos e funções implementadas. Em seguida, apresentaremos uma descrição sobre a interface desenvolvida e de que maneira se integra no trabalho. Após apresentarmos os resultados obtidos no fim do desenvolvimento do projeto, iremos enumerar algumas conclusões e medidas para um trabalho futuro sobre o tema.

# Capítulo 2

# Caso de Estudo

## 2.1 Domínio do Problema

O domínio estudado neste projeto é sobre genealogia. Foi-nos proposto que fosse abordado este tema de forma a detalhar a capacidade do nosso programa, face a problemas de restrições nesta área. Temos como objetivo construir uma interface capaz de retornar as soluções disponíveis para vários atributos sobre a árvore genealógica de uma família. Posteriormente, pretendemos adicionar a capacidade de o utilizador adicionar novas variáveis e restrições ao programa através da interface e consequentemente que este tenha a capacidade de devolver novas soluções depois de recalculadas com a nova informação.

## 2.2 Python Constraint

De forma a dar inicio ao desenvolvimento do projeto, tivemos de estudar que ferramenta era a mais adequada para o domínio em questão. Uma vez que nos foi pedido que desenvolvêssemos um programa em *Python* fomos à procura de ferramentas que simulassem o funcionamento do *Prolog* dentro do trabalho desenvolvido, bem como fossem capazes de adicionar ao mesmo as restrições que pretendíamos.

Assim sendo, encontramos o pacote **Python Constraint**. Esta ferramenta tem a capacidade de oferecer vários *solvers* de forma a dar ao programa em questão a capacidade de resolver problemas de restrições. Um problema de restrições é composto por variáveis e restrições sobre um determinado domínio, sendo neste caso um problema de genealogia.

Para dar o máximo uso da ferramenta, decidimos usar os vários *solvers* que esta oferece, sendo eles: o **Bactracking solver**, o **Recursive backtracking solver** e o **Minimum conflicts solver**.

Um algoritmo de *backtracking* caracteriza-se por ter a capacidade de resolver um dado problema

testando todos os caminhos até a uma possivel solução. Sempre que um caminho é testado e nao é encontrado uma solução o algoritmo volta atrás e testa outras alternativas, continuando deste modo até todos os caminhos serem testados ou até encontrar uma solução final.

Um algoritmo de *minimum conflictus* caracteriza-se por ser um algoritmo de procura heurístico. Dada uma atribuição inicial de valores a todas as variáveis de um problema de restrições, o algoritmo seleciona aleatoriamente uma variável do conjunto de variáveis com conflitos que violam uma ou mais restrições do problema. Em seguida, atribui a essa variável o valor que minimiza o número de conflitos. Se houver mais de um valor com um número mínimo de conflitos, ele escolhe um aleatoriamente. Esse processo de seleção de variável aleatória e atribuição de valor de conflito mínimo é iterado até que uma solução seja encontrada ou um número máximo de iterações pré-selecionado seja atingido.

Alem destas características, o pacote oferece um conjunto de funções que nos auxiliaram no momento de desenvolvimento do código do projecto. Estas funções irão ser descritas no próximo capitulo.

## 2.3 Funções Implementadas

### 2.3.1 Base de Conhecimento

De forma a desenvolver as funções para o código do programa decidimos dividir as mesmas por vários módulos.

A base de conhecimento do sistema encontra-se presente num ficheiro *json*. Este ficheiro contem a informação de vários indivíduos da árvore genealógica de uma possivel família. Assim sendo, cada individuo é caracterizado pelos atributos: nome, tipo (mãe, pai, filho ...), data de nascimento, data de falecimento, e um conjunto de relações de parentesco.

Apesar de o ficheiro inicial já conter um conjunto assinalável de indivíduos, o programa tem a capacidade de oferecer ao utilizador a possibilidade de inserir novas pessoas no sistema.

Esta funcionalidade é possivel através de um conjunto de funções que se situam no ficheiro *baseDeConhecimento.py*, que será anexado no fim do presente relatório.

```
def addFilho(filho):
file_open = open('teste.json','r')
data = json.load(file_open)
if filho in data:
    return False
else:
```

```
    data.append(filho)
    with open('teste.json', 'w') as outfile:
        json.dump(data, outfile)
    return True
```

Na função em cima, é possivel verificar o exemplo da inserção de um filho na base de conhecimento.

Assim que o utilizador entender que a base de conhecimento é suficiente para executar o programa, é necessário adicionar as variáveis do sistema. Recordamos que um problema de restrições é composto por variáreis de sistema e restrições sobre um determinado domínio. Esta primeira etapa, consistia em adicionar as variáveis de sistema.

```
 def carregarVariaveis(problem):
file_open = open('teste.json','r')
data = json.load(file_open)
listaFilho = []
listaPai = []
listaMae = []
listaAvoM = []
listaAvoF = []
for d in data:
    if d["tipo"] == "filho":
        aux = {
            "nome" : d["nome"],
            "dataNasc": d["dataNasc"],
            "dataMorte": d["dataMorte"]
        }
        listaFilho.append(aux)
    else:
        if d["tipo"] == "pai":
            aux = {
                "nome" : d["nome"],
                "dataNasc": d["dataNasc"],
                "dataMorte": d["dataMorte"],
                "ePai": d["ePai"]
            }
```

```
            listaPai.append(aux)
        else:
            if d["tipo"] == "mae":
                aux = {
                    "nome" : d["nome"],
                    "dataNasc": d["dataNasc"],
                    "dataMorte": d["dataMorte"],
                    "eMae": d["eMae"]
                }
                listaMae.append(aux)
            else:
                if d["tipo"] == "avo_m":
                    aux = {
                        "nome" : d["nome"],
                        "dataNasc": d["dataNasc"],
                        "dataMorte": d["dataMorte"],
                        "ePai": d["ePai"],
                        "eAvo": d["eAvo"]
                    }
                    listaAvoM.append(aux)
                else:
                    if d["tipo"] == "avo_f":
                        aux = {
                            "nome" : d["nome"],
                            "dataNasc": d["dataNasc"],
                            "dataMorte": d["dataMorte"],
                            "eMae": d["eMae"],
                            "eAvo": d["eAvo"]
                        }
                        listaAvoF.append(aux)
problem.addVariable("filho", listaFilho)
problem.addVariable("pai", listaPai)
problem.addVariable("mae", listaMae)
problem.addVariable("avo_m", listaAvoM)
problem.addVariable("avo_f",listaAvoF)
```

A função em cima é a responsável por carregar as variáveis de sistema. Todos os indivíduos presentes no ficheiro da base de conhecimento são carregados para o programa para seres usados no *solver*. Para executar este carregamento, utilizando o pacote *Python constraint*, usamos a função **addVariable** que este disponibiliza.

Estas restrições irão ser descritas no próximo sub-capitulo.

### 2.3.2 Restrições

As restrições usadas na realização deste projeto foram as seguintes:

- Restrição entre diferença de Datas de Nascimento entre Pai e Filho:

  Esta restrição diz que o pai só pode ter um filho caso tenha mais de doze anos, a data de nascimento do filho tem de ser superior a doze anos da data de nascimento do pai.

- Intervalo de Nascimento entre Mãe e Filho:

  Esta restrição tem uma pequena diferença da anterior, a data de nascimento do filho tem de estar contida no intervalo de doze anos da mãe e os seus cinquenta anos.

- Filho nasce antes da Data de falecimento:

  Esta restrição diz que um filho só pode nascer caso os pais tenham uma data de morte superior a data de nascimento dos filhos.

- Avo - Neto:

  Restrição para impedir que tenhamos resultados com avós e netos não relacionados.

- Pai - Filho:

  Igual a restrição anterior, impedir que tenhamos pais e filhos não relacionados.

- Mãe - Filho:

  Tal como as duas anteriores temos definida a restrição para a relação mãe e filho.

- Casados:

  Para impedir que apareça um conjunto de resultados com o avô materno e a avó paterna, ou pais distintos para filhos diferentes temos a restrição que diz que têm de ser casados para aparecer nos resultados.

## 2.4 Interface

A interface do programa que foi desenvolvida para fazer a ponte entre o utilizador e o sistema, está presente na linha de comandos assim que executamos o software.

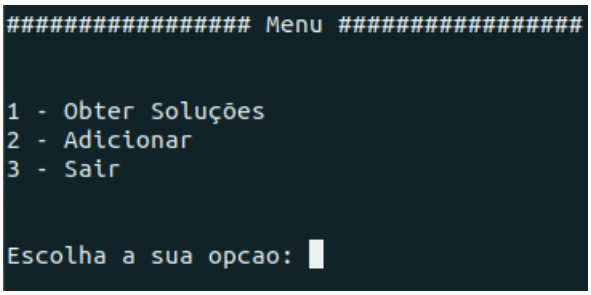


Figura 2.1: Menu Inicial.

No menu inicial existe a capacidade de adicionar novas variáveis à base de conhecimento, ou por outro lado obter as soluções com as variáveis e restrições existentes.

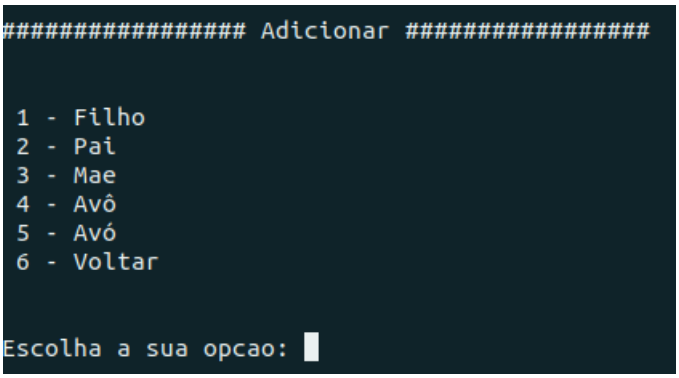


Figura 2.2: Menu para adicionar variáveis.

No menu para adicionar variáveis, é possível adicionar novos indivíduos ao sistema. Filho, pai, mae avô e avó são as opções disponíveis.

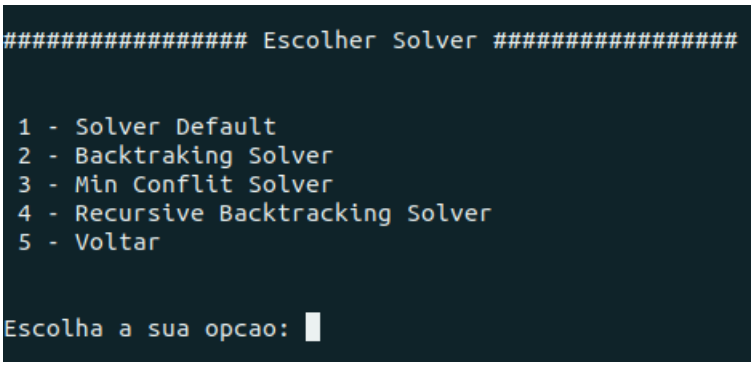


Figura 2.3: Menu para escolher o tipo de solver.

No menu para escolher o tipo de solver que irá ser usado para calcular as soluções do sistema apresentamos todas as opções estudadas.

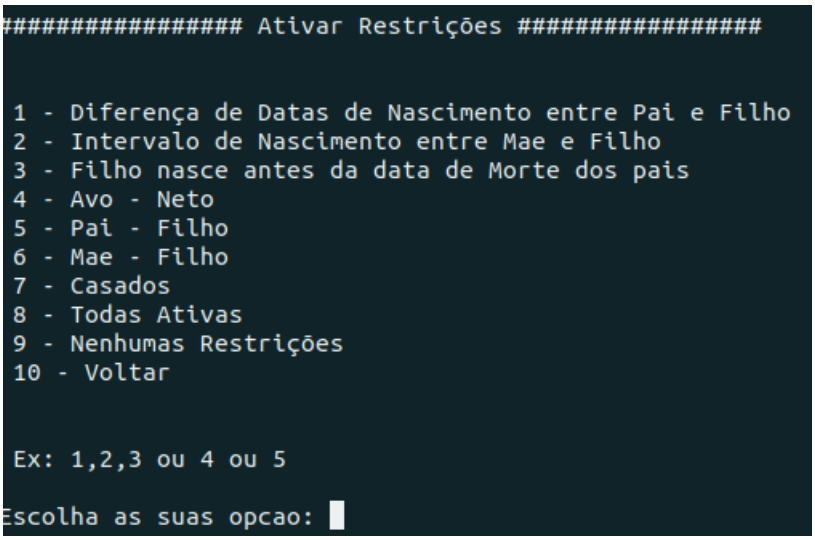


Figura 2.4: Menu para escolher o tipo de solver.

Por fim, damos a possibilidade ao utilizador de ativar as restrições que pretende, sendo que pode ativar todas, nenhuma ou alguma em especifico.

# Capítulo 3

# Resultados

Concluída a etapa anterior de analise do problema e modelação do mesmo, chegou o momento de fazer os testes ao programa de forma a interpretar os resultados obtidos.

Neste momento, o programa já possuía as variáveis e as restrições que pretendíamos para o problema, desta forma tivemos de fazer os testes para os vários tipos de *solvers* que iríamos usar. Fora usados 4 tipos de *solvers*. Para os testes que iriam ser efetuados decidimos que todas as restrições iriam ser ativadas, desta forma podemos ver como o programa se comporta na situação mais difícil.

O primeiro teste efetuado foi com o *solver* por defeito. Para este teste obtivemos os mesmos resultados que para os dois posteriores testes, ou seja para o solver com *bactracking* e para o solver com *bactracking* recursivo.

Para o solver *Min Conflit* apenas obtivemos um resultado.


Figura 3.1: Solver default.


Figura 3.2: Backtracking solver.


Figura 3.3: Recursive Backtracking solver.

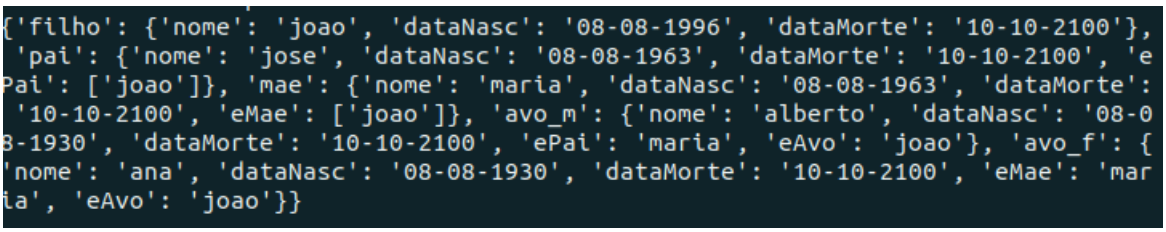


Figura 3.4: Min conflit solver.

# Capítulo 4

# Conclusões e Trabalho Futuro

Damos por concluído o trabalho prático, onde abordamos em forma de resumo o estudo sobre o problemas de restrições sobre genealogia.

Após concluirmos o nosso estudo, entendemos que para efetuar este programa em *Python*, como nos foi solicitado teríamos de usar o pacote *Python Constraint*.

Deste modo, verificamos num primeiro contacto com a ferramenta que este tipo de problemas não seriam os adequados a serem resolvidos com a mesma. Através de algumas pesquisas, entendemos que problemas do género do *Sudoku*, seriam problemas mais adequados a serem resolvidos com o módulo *Python Constraint*.

Outra dificuldade detetada logo no inicio e que nos levou a adotar o uso deste pacote, foi a longa distancia no tempo desde a ultima vez que trabalhos com a linguagem *PROLOG*, bem como a dificuldade que enfrentamos em importar módulos desta linguagem para *Python*.

Apesar das dificuldades, conseguimos obter resultados satisfatórios para os casos testados, embora a complexidade do problema não tivesse sido a desejada.

Aqui encontramos a primeira medida que achamos que poderá ser explorada no futuro, que será a complexidade do problema. De forma a continuar este projeto, pensamos que no futuro a dificuldade e complexidade das restrições pode e dever ser aumentada para ser possível obter um software mais complexo que consiga responder a perguntas mais variadas às que responde neste momento.

Em suma, apesar da falha em cima apontada, a ferramenta estudada é capaz de responder ao pedido o que nos leva a afirmar que é uma boa base para este trabalho continuar a ser desenvolvido num futuro próximo.

# Apêndice A

# Anexo

Apresentamos o código das funções implementadas.

## A.1 auxiliarFunctions.py

```
1  from constraint import *
2
3
4  def init():
5      problem = Problem()
6      return problem
7
8  def carregamento_por_ficheiro(file,p):
9      p.consult(file)
10
11 def getResults(problem):
12     return problem.getSolutions()
13
14 def addVariavel(nome,variavel,problem):
15     problem.addVariable(nome, [variavel])
16
17 def addConstraint(intervenientes,constraint,problem):
18     problem.addConstraint(constraint)
```

## A.2 baseDeConhecimento.py

```
1  from constraint import *
2  import constraintFunctions as const
3  import json
```

```
4
5  # Fun  es de Adicionar
6
7  def addFilho(filho):
8      file_open = open('teste.json','r')
9      data = json.load(file_open)
10     if filho in data:
11         return False
12     else:
13         data.append(filho)
14         with open('teste.json', 'w') as outfile:
15             json.dump(data, outfile)
16         return True
17
18 def addPai(pai):
19     file_open = open('teste.json','r')
20     data = json.load(file_open)
21     if pai in data:
22         return False
23     else:
24         data.append(pai)
25         with open('teste.json', 'w') as outfile:
26             json.dump(data, outfile)
27         return True
28
29 def addMae(mae):
30     file_open = open('teste.json','r')
31     data = json.load(file_open)
32     if mae in data:
33         return False
34     else:
35         data.append(mae)
36         with open('teste.json', 'w') as outfile:
37             json.dump(data, outfile)
38         return True
39
40 def addAvo_M(avo):
41     file_open = open('teste.json','r')
42     data = json.load(file_open)
43     if avo in data:
44         return False
45     else:
46         data.append(avo)
```

```
47          with open('teste.json', 'w') as outfile:
48              json.dump(data, outfile)
49          return True
50
51  def addAvo_F(avo_f):
52      file_open = open('teste.json','r')
53      data = json.load(file_open)
54      if avo_f in data:
55          return False
56      else:
57          data.append(avo_f)
58          with open('teste.json', 'w') as outfile:
59              json.dump(data, outfile)
60          return True
61
62
63
64  def carregarVariaveis(problem):
65      file_open = open('teste.json','r')
66      data = json.load(file_open)
67      listaFilho = []
68      listaPai = []
69      listaMae = []
70      listaAvoM = []
71      listaAvoF = []
72      for d in data:
73          if d["tipo"] == "filho":
74              aux = {
75                  "nome" : d["nome"],
76                  "dataNasc": d["dataNasc"],
77                  "dataMorte": d["dataMorte"]
78              }
79              listaFilho.append(aux)
80          else:
81              if d["tipo"] == "pai":
82                  aux = {
83                      "nome" : d["nome"],
84                      "dataNasc": d["dataNasc"],
85                      "dataMorte": d["dataMorte"],
86                      "ePai": d["ePai"]
87                  }
88                  listaPai.append(aux)
89              else:
```

```
90                  if d["tipo"] == "mae":
91                      aux = {
92                          "nome" : d["nome"],
93                          "dataNasc": d["dataNasc"],
94                          "dataMorte": d["dataMorte"],
95                          "eMae": d["eMae"]
96                      }
97                      listaMae.append(aux)
98                  else:
99                      if d["tipo"] == "avo_m":
100                         aux = {
101                             "nome" : d["nome"],
102                             "dataNasc": d["dataNasc"],
103                             "dataMorte": d["dataMorte"],
104                             "ePai": d["ePai"],
105                             "eAvo": d["eAvo"]
106                         }
107                         listaAvoM.append(aux)
108                     else:
109                         if d["tipo"] == "avo_f":
110                             aux = {
111                                 "nome" : d["nome"],
112                                 "dataNasc": d["dataNasc"],
113                                 "dataMorte": d["dataMorte"],
114                                 "eMae": d["eMae"],
115                                 "eAvo": d["eAvo"]
116                             }
117                             listaAvoF.append(aux)
118     problem.addVariable("filho", listaFilho)
119     problem.addVariable("pai", listaPai)
120     problem.addVariable("mae", listaMae)
121     problem.addVariable("avo_m", listaAvoM)
122     problem.addVariable("avo_f",listaAvoF)
123
124
125 # Solvers
126 def get_solver_default(flags):
127     problem = Problem()
128
129     carregarVariaveis(problem)
130
131     # 1 - Restri  o entre diferen a de Datas de Nascimento entre Pai e
        Filho
```

```
132     if 1 in flags or 8 in flags:
133         problem.addConstraint(const.dataNascimento_pai_filho,("pai","filho")
                )
134         problem.addConstraint(const.dataNascimento_avo_m_filho,("avo_m", "
                pai","mae"))
135
136     # 2 - Intervalo de Nascimento entre Mae e Filho
137     if 2 in flags or 8 in flags:
138         problem.addConstraint(const.dataNascimento_mae_filho,("mae", "filho"
                ))
139         problem.addConstraint(const.dataNascimento_avo_f_filho,("avo_f", "
                pai","mae"))
140
141     # 3 - Filho nasce antes da Data de Morte
142     if 3 in flags or 8 in flags:
143         problem.addConstraint(const.dataMorte_pai_filho,("pai","filho"))
144         problem.addConstraint(const.dataMorte_mae_filho,("mae","filho"))
145         problem.addConstraint(const.dataMorte_avo_m_filho,("avo_m","pai","
                mae"))
146         problem.addConstraint(const.dataMorte_avo_f_filho,("avo_f","pai","
                mae"))
147
148     # 4 - Avo - Neto
149     if 4 in flags or 8 in flags:
150         problem.addConstraint(const.avo_neto,("avo_m","avo_f","filho"))
151
152     # 5 - Pai - Filho
153     if 5 in flags or 8 in flags:
154         problem.addConstraint(const.pai_filho,("pai","filho"))
155         problem.addConstraint(const.avo_m_filho,("avo_m","pai","mae"))
156
157     # 6 - Mae - Filho
158     if 6 in flags or 8 in flags:
159         problem.addConstraint(const.mae_filho,("mae","filho"))
160         problem.addConstraint(const.avo_f_filho,("avo_f","pai","mae"))
161
162     # 7 - Casados
163     if 7 in flags or 8 in flags:
164         problem.addConstraint(const.casado_pai_mae,("pai","mae"))
165         problem.addConstraint(const.casado_avo_m_avo_f,("avo_m","avo_f"))
166
167     results = problem.getSolutions()
168     if results == []:
```

```
169             return 'N o foram obtidos resultados com estas restri   es'
170         else:
171             return results
172
173 def get_backtracking_solver(flags):
174     problem = Problem(BacktrackingSolver())
175
176     carregarVariaveis(problem)
177
178     # 1 - Restri   o entre diferen a de Datas de Nascimento entre Pai e
            Filho
179     if 1 in flags or 8 in flags:
180         problem.addConstraint(const.dataNascimento_pai_filho,("pai","filho")
                )
181         problem.addConstraint(const.dataNascimento_avo_m_filho,("avo_m", "
                pai","mae"))
182
183     # 2 - Intervalo de Nascimento entre Mae e Filho
184     if 2 in flags or 8 in flags:
185         problem.addConstraint(const.dataNascimento_mae_filho,("mae", "filho"
                ))
186         problem.addConstraint(const.dataNascimento_avo_f_filho,("avo_f", "
                pai","mae"))
187
188     # 3 - Filho nasce antes da Data de Morte
189     if 3 in flags or 8 in flags:
190         problem.addConstraint(const.dataMorte_pai_filho,("pai","filho"))
191         problem.addConstraint(const.dataMorte_mae_filho,("mae","filho"))
192         problem.addConstraint(const.dataMorte_avo_m_filho,("avo_m","pai","
                mae"))
193         problem.addConstraint(const.dataMorte_avo_f_filho,("avo_f","pai","
                mae"))
194
195     # 4 - Avo - Neto
196     if 4 in flags or 8 in flags:
197         problem.addConstraint(const.avo_neto,("avo_m","avo_f","filho"))
198
199     # 5 - Pai - Filho
200     if 5 in flags or 8 in flags:
201         problem.addConstraint(const.pai_filho,("pai","filho"))
202         problem.addConstraint(const.avo_m_filho,("avo_m","pai","mae"))
203
204     # 6 - Mae - Filho
```

```
205     if 6 in flags or 8 in flags:
206         problem.addConstraint(const.mae_filho,("mae","filho"))
207         problem.addConstraint(const.avo_f_filho,("avo_f","pai","mae"))
208 
209     # 7 - Casados
210     if 7 in flags or 8 in flags:
211         problem.addConstraint(const.casado_pai_mae,("pai","mae"))
212         problem.addConstraint(const.casado_avo_m_avo_f,("avo_m","avo_f"))
213 
214     results = problem.getSolutions()
215     if results == []:
216         return 'N o foram obtidos resultados com estas restri   es'
217     else:
218         return results
219 
220 def get_min_conflit_solver(flags):
221     problem = Problem(MinConflictsSolver())
222 
223     carregarVariaveis(problem)
224 
225     # 1 - Restri   o entre diferen a de Datas de Nascimento entre Pai e
            Filho
226     if 1 in flags or 8 in flags:
227         problem.addConstraint(const.dataNascimento_pai_filho,("pai","filho")
                )
228         problem.addConstraint(const.dataNascimento_avo_m_filho,("avo_m", "
                pai","mae"))
229 
230     # 2 - Intervalo de Nascimento entre Mae e Filho
231     if 2 in flags or 8 in flags:
232         problem.addConstraint(const.dataNascimento_mae_filho,("mae", "filho"
                ))
233         problem.addConstraint(const.dataNascimento_avo_f_filho,("avo_f", "
                pai","mae"))
234 
235     # 3 - Filho nasce antes da Data de Morte
236     if 3 in flags or 8 in flags:
237         problem.addConstraint(const.dataMorte_pai_filho,("pai","filho"))
238         problem.addConstraint(const.dataMorte_mae_filho,("mae","filho"))
239         problem.addConstraint(const.dataMorte_avo_m_filho,("avo_m","pai","
                mae"))
240         problem.addConstraint(const.dataMorte_avo_f_filho,("avo_f","pai","
                mae"))
```

```
241
242     # 4 - Avo - Neto
243     if 4 in flags or 8 in flags:
244         problem.addConstraint(const.avo_neto,("avo_m","avo_f","filho"))
245
246     # 5 - Pai - Filho
247     if 5 in flags or 8 in flags:
248         problem.addConstraint(const.pai_filho,("pai","filho"))
249         problem.addConstraint(const.avo_m_filho,("avo_m","pai","mae"))
250
251     # 6 - Mae - Filho
252     if 6 in flags or 8 in flags:
253         problem.addConstraint(const.mae_filho,("mae","filho"))
254         problem.addConstraint(const.avo_f_filho,("avo_f","pai","mae"))
255
256     # 7 - Casados
257     if 7 in flags or 8 in flags:
258         problem.addConstraint(const.casado_pai_mae,("pai","mae"))
259         problem.addConstraint(const.casado_avo_m_avo_f,("avo_m","avo_f"))
260
261     results = problem.getSolution()
262     if results == None:
263         return 'N o foram obtidos resultados com estas restri   es'
264     else:
265         return results
266
267 def get_recursive_backtraking_solver(flags):
268     problem = Problem(RecursiveBacktrackingSolver())
269
270     carregarVariaveis(problem)
271
272     # 1 - Restri   o entre diferen a de Datas de Nascimento entre Pai e
            Filho
273     if 1 in flags or 8 in flags:
274         problem.addConstraint(const.dataNascimento_pai_filho,("pai","filho")
                )
275         problem.addConstraint(const.dataNascimento_avo_m_filho,("avo_m", "
                pai","mae"))
276
277     # 2 - Intervalo de Nascimento entre Mae e Filho
278     if 2 in flags or 8 in flags:
279         problem.addConstraint(const.dataNascimento_mae_filho,("mae", "filho"
                ))
```

```
280             problem.addConstraint(const.dataNascimento_avo_f_filho,("avo_f", "
                    pai","mae"))
281
282         # 3 - Filho nasce antes da Data de Morte
283         if 3 in flags or 8 in flags:
284             problem.addConstraint(const.dataMorte_pai_filho,("pai","filho"))
285             problem.addConstraint(const.dataMorte_mae_filho,("mae","filho"))
286             problem.addConstraint(const.dataMorte_avo_m_filho,("avo_m","pai","
                    mae"))
287             problem.addConstraint(const.dataMorte_avo_f_filho,("avo_f","pai","
                    mae"))
288
289         # 4 - Avo - Neto
290         if 4 in flags or 8 in flags:
291             problem.addConstraint(const.avo_neto,("avo_m","avo_f","filho"))
292
293         # 5 - Pai - Filho
294         if 5 in flags or 8 in flags:
295             problem.addConstraint(const.pai_filho,("pai","filho"))
296             problem.addConstraint(const.avo_m_filho,("avo_m","pai","mae"))
297
298         # 6 - Mae - Filho
299         if 6 in flags or 8 in flags:
300             problem.addConstraint(const.mae_filho,("mae","filho"))
301             problem.addConstraint(const.avo_f_filho,("avo_f","pai","mae"))
302
303         # 7 - Casados
304         if 7 in flags or 8 in flags:
305             problem.addConstraint(const.casado_pai_mae,("pai","mae"))
306             problem.addConstraint(const.casado_avo_m_avo_f,("avo_m","avo_f"))
307
308         results = problem.getSolutions()
309         if results == []:
310             return 'N o foram obtidos resultados com estas restri   es'
311         else:
312             return results
```

## A.3 constraintFunctions.py

```
1
2
3 # 1 - Restri   o entre diferen a de Datas de Nascimento entre Pai e Filho
4
```

```
5  def dataNascimento_pai_filho(pai,filho):
6      if filho["nome"] in pai["ePai"]:
7          return int(filho["dataNasc"].split('-')[2]) > int(pai["dataNasc"].
               split('-')[2]) + 13
8      else:
9          return False
10
11 def dataNascimento_avo_m_filho(avo,pai,mae):
12     if pai["nome"] in avo["ePai"]:
13         return int(pai["dataNasc"].split('-')[2]) > int(avo["dataNasc"].
               split('-')[2]) + 13
14     else:
15         if mae["nome"] in avo["ePai"]:
16             return int(mae["dataNasc"].split('-')[2]) > int(avo["dataNasc"].
                   split('-')[2]) + 13
17         else:
18             return False
19
20 # 2 - Restri  o entre diferen a de Datas de Nascimento entre Mae e Filho
21 def dataNascimento_mae_filho(mae,filho):
22     if filho["nome"] in mae["eMae"]:
23          return int(mae["dataNasc"].split('-')[2])+12 <= int(filho["dataNasc
               "].split('-')[2]) and int(filho["dataNasc"].split('-')[2]) <=
               int(mae["dataNasc"].split('-')[2])+50
24     else:
25         return False
26
27 def dataNascimento_avo_f_filho(avo,pai,mae):
28     if pai["nome"] in avo["eMae"]:
29         return int(avo["dataNasc"].split('-')[2])+12 <= int(pai["dataNasc"].
               split('-')[2]) and int(pai["dataNasc"].split('-')[2]) <= int(avo[
               "dataNasc"].split('-')[2])+50
30     else:
31         if mae["nome"] in avo["eMae"]:
32             return int(avo["dataNasc"].split('-')[2])+12 <= int(mae["
                   dataNasc"].split('-')[2]) and int(mae["dataNasc"].split('-')
                   [2]) <= int(avo["dataNasc"].split('-')[2])+50
33         else:
34             return False
35
36 # 3 - N o pode estar morto quando o filho nasce
37
38 def dataMorte_pai_filho(pai,filho):
```

```
39      if filho["nome"] in pai["ePai"]:
40          return int(pai["dataMorte"].split('-')[2]) > int(filho["dataNasc"].
                split('-')[2])
41      else:
42          return False
43
44  def dataMorte_mae_filho(mae,filho):
45      if filho["nome"] in mae["eMae"]:
46          return int(mae["dataMorte"].split('-')[2]) > int(filho["dataNasc"].
                split('-')[2])
47      else:
48          return False
49
50  def dataMorte_avo_m_filho(avo,pai,mae):
51      if pai["nome"] in avo["ePai"]:
52          return int(avo["dataMorte"].split('-')[2]) > int(pai["dataNasc"].
                split('-')[2])
53      else:
54          if mae["nome"] in avo["ePai"]:
55              return int(avo["dataMorte"].split('-')[2]) > int(mae["dataNasc"
                    ].split('-')[2])
56          else:
57              return False
58
59  def dataMorte_avo_f_filho(avo,pai,mae):
60      if pai["nome"] in avo["eMae"]:
61          return int(avo["dataMorte"].split('-')[2]) > int(pai["dataNasc"].
                split('-')[2])
62      else:
63          if mae["nome"] in avo["eMae"]:
64              return int(avo["dataMorte"].split('-')[2]) > int(mae["dataNasc"
                    ].split('-')[2])
65          else:
66              return False
67
68
69  # 5 - Avo-Neto
70
71  def avo_neto(avo_m,avo_f,filho):
72      if filho["nome"] in avo_m["eAvo"]:
73          if filho["nome"] in avo_f["eAvo"]:
74              return True
75          else:
```

```
76              return False
77      else:
78          return False
79
80  # 6 - Pai - Filho
81
82  def pai_filho(pai,filho):
83      if filho["nome"] in pai["ePai"]:
84          return True
85      else:
86          return False
87
88  def avo_m_filho(avo,pai,mae):
89      if pai["nome"] in avo["ePai"]:
90          return True
91      else:
92          if mae["nome"] in avo["ePai"]:
93              return True
94          else:
95              return False
96
97  # 7 - Mae - Filho
98
99  def mae_filho(mae,filho):
100     if filho["nome"] in mae["eMae"]:
101         return True
102     else:
103         return False
104
105 def avo_f_filho(avo,pai,mae):
106     if pai["nome"] in avo["eMae"]:
107         return True
108     else:
109         if mae["nome"] in avo["eMae"]:
110             return True
111         else:
112             return False
113
114 # 8 - Casado Pai - Mae
115
116 def casado_pai_mae(pai,mae):
117     if pai["ePai"] == mae["eMae"]:
118         return True
```

```
119     else:
120         return False
121 
122 def casado_avo_m_avo_f(avo_m,avo_f):
123     if avo_m["ePai"] == avo_f["eMae"]:
124         return True
125     else:
126         return False
```

## A.4 interface.py

```
1  import os
2  
3  def showMenuPrincipal():
4      print('\n\n################## Menu ##################\n\n')
5      print('1 - Obter Solu  es')
6      print('2 - Adicionar')
7      print('3 - Sair\n\n')
8  
9  def showMenu_Inicializacao():
10     print('\n\n################## Escolher Solver ##################\n\n')
11     print(' 1 - Solver Default')
12     print(' 2 - Backtraking Solver')
13     print(' 3 - Min Conflit Solver')
14     print(' 4 - Recursive Backtracking Solver')
15     print(' 5 - Voltar\n\n')
16 
17 def showMenu_Constraints():
18     print('\n\n################## Ativar Restri  es ##################\n\n')
19     print(' 1 - Diferen a de Datas de Nascimento entre Pai e Filho')
20     print(' 2 - Intervalo de Nascimento entre Mae e Filho')
21     print(' 3 - Filho nasce antes da data de Morte dos pais')
22     print(' 4 - Avo - Neto')
23     print(' 5 - Pai - Filho')
24     print(' 6 - Mae - Filho')
25     print(' 7 - Casados')
26     print(' 8 - Todas Ativas')
27     print(' 9 - Nenhumas Restri  es')
28     print(' 10 - Voltar\n\n')
29     print(' Ex: 1,2,3 ou 4 ou 5\n')
30 
31 def showMenu_Adicionar():
32     print('\n\n################## Adicionar ##################\n\n')
```

```
33      print(' 1 - Filho')
34      print(' 2 - Pai')
35      print(' 3 - Mae')
36      print(' 4 - Av ')
37      print(' 5 - Av ')
38      print(' 6 - Voltar \n\n')
```

## A.5 main.py

```
1  import interface
2  import baseDeConhecimento as base
3  from constraint import *
4  import os
5
6
7  def main():
8      while True :
9          interface.showMenuPrincipal()
10         opcao = int(input("Escolha a sua opcao: "))
11         if opcao == 1:
12             while True:
13                 interface.showMenu_Inicializacao()
14                 opcao2 = int(input("Escolha a sua opcao: "))
15                 if opcao2 == 1:
16                     while(True):
17                         interface.showMenu_Constraints()
18                         opcao3 = input("Escolha as suas opcao: ")
19                         if ',' in opcao3:
20                             flags = opcao3.split(',')
21                             flags = [int(x) for x in flags]
22                             print(base.get_solver_default(flags))
23                         else:
24                             if int(opcao3) == 10:
25                                     break
26                             else:
27                                 num = int(opcao3)
28                                 flags = [int(x) for x in str(num)]
29                                 print(base.get_solver_default(flags))
30                 else:
31                     if opcao2 == 2:
32                         while True:
33                             interface.showMenu_Constraints()
34                             opcao3 = input("Escolha as suas opcao: ")
```

```
35                          if ',' in opcao3:
36                              flags = opcao3.split(',')
37                              flags = [int(x) for x in flags]
38                              print(base.get_backtracking_solver(flags))
39                          else:
40                              if int(opcao3) == 10:
41                                  break
42                              else:
43                                  num = int(opcao3)
44                                  flags = [int(x) for x in str(num)]
45                                  print(base.get_backtracking_solver(flags
                                        ))
46                  else:
47                      if opcao2 == 3:
48                          while True:
49                              interface.showMenu_Constraints()
50                              opcao3 = input("Escolha as suas opcao: ")
51                              if ',' in opcao3:
52                                  flags = opcao3.split(',')
53                                  flags = [int(x) for x in flags]
54                                  print(base.get_min_conflit_solver(flags)
                                        )
55                              else:
56                                  if int(opcao3) == 10:
57                                      break
58                                  else:
59                                      num = int(opcao3)
60                                      flags = [int(x) for x in str(num)]
61                                      print(base.get_min_conflit_solver(
                                            flags))
62                      else:
63                          if opcao2 == 4:
64                              while True:
65                                  interface.showMenu_Constraints()
66                                  opcao3 = input("Escolha as suas opcao: "
                                        )
67                                  if ',' in opcao3:
68                                      flags = opcao3.split(',')
69                                      flags = [int(x) for x in flags]
70                                      print(base.
                                            get_recursive_backtraking_solver(
                                            flags))
71                                  else:
```

```
72                                          if int(opcao3) == 10:
73                                              break
74                                          else:
75                                              num = int(opcao3)
76                                              flags = [int(x) for x in str(num
                                                    )]
77                                              print(base.
                                                    get_recursive_backtraking_solver
                                                    (flags))
78                                  else:
79                                      if opcao2 == 5:
80                                          break
81                                      else:
82                                          print('\nOpcao Invalida!!!\n')
83          else:
84              if opcao == 2:
85                  while True:
86                      interface.showMenu_Adicionar()
87                      opcao2 = int(input("Escolha a sua opcao: "))
88                      if opcao2 == 1:
89                          nome = input("Nome do Filho: ")
90                          dataNasc = input("Data de Nascimento(DD-MM-AAAA): ")
91                          dataMorte = input("Data de Morte(DD-MM-AAAA): ")
92                          if dataMorte == '':
93                              dataMorte = '10-10-2100'
94                          filho = {
95                              "tipo":"filho",
96                              "nome": nome,
97                              "dataNasc": dataNasc,
98                              "dataMorte": dataMorte
99                          }
100                         base.addFilho(filho)
101                     else:
102                         if opcao2 == 2:
103                             nome = input("Nome do Pai: ")
104                             dataNasc = input("Data de Nascimento(DD-MM-AAAA)
                                    : ")
105                             dataMorte = input("Data de Morte(DD-MM-AAAA): ")
106                             if dataMorte == '':
107                                 dataMorte = '10-10-2100'
108                             nFilhos = int(input("N mero de Filhos :"))
109                             ePais = []
110                             for i in range(0, nFilhos):
```

```
111                             ePai = input("Nome do Filho: ")
112                             ePais.append(ePai)
113                         pai = {
114                             "tipo":"pai",
115                             "nome": nome,
116                             "dataNasc": dataNasc,
117                             "dataMorte": dataMorte,
118                             "ePai": ePais
119                         }
120                         base.addPai(pai)
121                     else:
122                         if opcao2 == 3:
123                             nome = input("Nome do Mae: ")
124                             dataNasc = input("Data de Nascimento(DD-MM-
                                    AAAA): ")
125                             dataMorte = input("Data de Morte(DD-MM-AAAA)
                                    : ")
126                             if dataMorte == '':
127                                 dataMorte = '10-10-2100'
128                             nFilhos = int(input("N mero de Filhos :"))
129                             eMaes = []
130                             for i in range(0, nFilhos) :
131                                 eMae = input("Nome do Filho:")
132                                 eMaes.append(eMae)
133                             mae = {
134                                 "tipo":"mae",
135                                 "nome": nome,
136                                 "dataNasc": dataNasc,
137                                 "dataMorte": dataMorte,
138                                 "eMae": eMaes
139                             }
140                             base.addMae(mae)
141                         else:
142                             if opcao2 == 4:
143                                 nome = input("Nome do Av : ")
144                                 dataNasc = input("Data de Nascimento(DD-
                                        MM-AAAA): ")
145                                 dataMorte = input("Data de Morte(DD-MM-
                                        AAAA): ")
146                                 if dataMorte == '':
147                                     dataMorte = '10-10-2100'
148                                 nFilhos = int(input("N mero de Filhos :
                                        "))
```

```
149                     ePais = []
150                     for i in range(0, nFilhos) :
151                         ePai = input("Nome do Filho: ")
152                         ePais.append(ePai)
153                     eAvos = []
154                     nNetos = int(input("N mero de Netos :")
                            )
155                     for i in range(0, nNetos):
156                         eAvo = input("Nome do Neto: ")
157                         eAvos.append(eAvo)
158                     avo_m = {
159                         "tipo":"avo_m",
160                         "nome": nome,
161                         "dataNasc": dataNasc,
162                         "dataMorte": dataMorte,
163                         "ePai": ePais,
164                         "eAvo": eAvos
165                     }
166                     base.addAvo_M(avo_m)
167                 else:
168                     if opcao2 == 5:
169                         nome = input("Nome da Av : ")
170                         dataNasc = input("Data de Nascimento
                                (DD-MM-AAAA): ")
171                         dataMorte = input("Data de Morte(DD-
                                MM-AAAA): ")
172                         if dataMorte == '':
173                             dataMorte = '10-10-2100'
174                         nFilhos = int(input("N mero de
                                Filhos :"))
175                         eMaes = []
176                         for i in range(0, nFilhos):
177                             eMae = input("Nome do Filho:")
178                             eMaes.append(eMae)
179                         eAvos = []
180                         nNetos = int(input("N mero de Netos
                                :"))
181                         for i in range(0, nNetos):
182                             eAvo = input("Nome do Neto: ")
183                             eAvos.append(eAvo)
184
185                         avo_f = {
186                             "tipo":"avo_f",
```

```
                                        "nome": nome,
                                        "dataNasc": dataNasc,
                                        "dataMorte": dataMorte,
                                        "eMae": eMaes,
                                        "eAvo": eAvos
                                    }
                                    base.addAvo_F(avo_f)
                                else:
                                    if opcao2 == 6:
                                        break
                                    else:
                                        print('\nOpcao Invalida!!!\n')
                    else:
                        if opcao == 3:
                            break
                        else:
                            print('\nOpcao Invalida!!!\n')


main()
```

## A.6 teste.json

```
[
    {
        "nome": "joao",
        "tipo": "filho",
        "dataNasc": "08-08-1996",
        "dataMorte": "10-10-2100"
    },
    {
        "nome": "damasio",
        "tipo": "filho",
        "dataNasc": "08-08-1996",
        "dataMorte": "10-10-2100"
    },
    {
        "nome": "jose",
        "tipo": "pai",
        "dataNasc": "08-08-1963",
        "dataMorte": "10-10-2100",
        "ePai": [
            "joao"
```

```
            ]
        },
        {
            "nome": "maria",
            "tipo": "mae",
            "dataNasc": "08-08-1963",
            "dataMorte": "10-10-2100",
            "eMae": [
                "joao"
            ]
        },
        {
            "nome": "jose",
            "tipo": "avo_m",
            "dataNasc": "08-08-1930",
            "dataMorte": "10-10-2100",
            "ePai": [
                "jose"
            ],
            "eAvo": [
                "joao"
            ]
        },
        {
            "nome": "maria antonia",
            "tipo": "avo_f",
            "dataNasc": "08-08-1930",
            "dataMorte": "10-10-2100",
            "eMae": [
                "jose"
            ],
            "eAvo": [
                "joao"
            ]
        },
        {
            "nome": "alberto",
            "tipo": "avo_m",
            "dataNasc": "08-08-1930",
            "dataMorte": "10-10-2100",
            "ePai": "maria",
            "eAvo": "joao"
        },
```

```
    {
        "nome": "ana",
        "tipo": "avo_f",
        "dataNasc": "08-08-1930",
        "dataMorte": "10-10-2100",
        "eMae": "maria",
        "eAvo": "joao"
    },
    {
        "tipo": "filho",
        "nome": "tobias",
        "dataNasc": "10-10-2010",
        "dataMorte": "10-10-2100"
    },
    {
        "tipo": "pai",
        "nome": "serafim",
        "dataNasc": "10-10-1980",
        "dataMorte": "10-10-2009",
        "ePai": [
            "tobias"
        ]
    },
    {
        "tipo": "mae",
        "nome": "rosa",
        "dataNasc": "10-10-1980",
        "dataMorte": "10-10-2100",
        "eMae": [
            "tobias"
        ]
    },
    {
        "tipo": "avo_m",
        "nome": "tiago",
        "dataNasc": "10-10-1970",
        "dataMorte": "10-10-2100",
        "ePai": [
            "serafim"
        ],
        "eAvo": [
            "tobias"
        ]
```

```
        },
        {
            "tipo": "avo_f",
            "nome": "rosas",
            "dataNasc": "10-10-1940",
            "dataMorte": "10-10-2100",
            "eMae": [
                "rosa"
            ],
            "eAvo": [
                "tobias"
            ]
        }
]
```