Auxílio na Alocação de Pessoal na ATI-PE Utilizando Algoritmo Genético.

Contents

Da linguagem e do ambiente:

A solução foi desenvolvida na linguegem Python, versão 3.6.3. Para a execução do código é necessário instalar a biblioteca Pandas (responsável pelo parsing dos dados das planilhas). Recomenda-se fortemente a utilização do sistema operacional Linux, tendo em vista a impossibilidade de um setup completo no sistema operacional Windows (mais especificamente versões 7 e 10).

Da solução:

A solução consiste nos seguintes componentes:

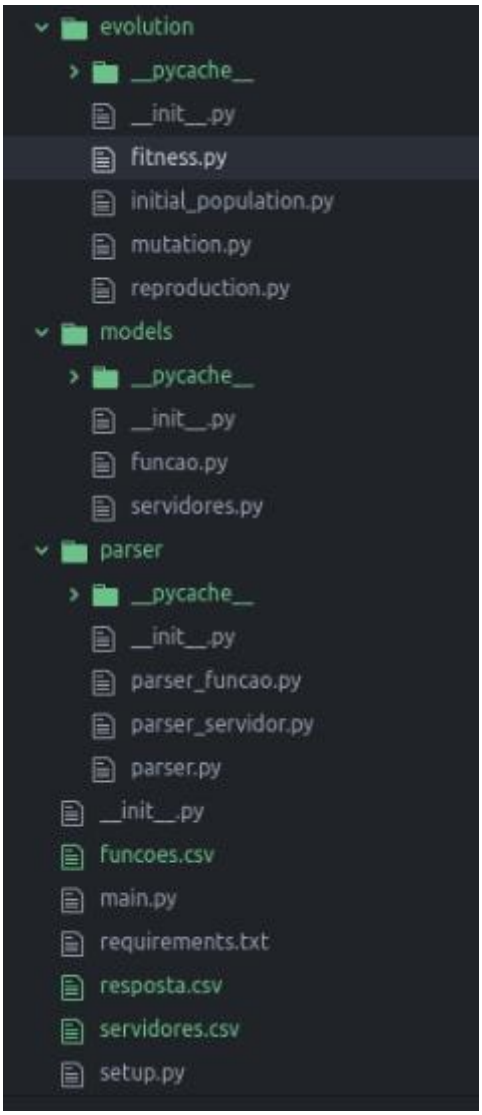

Figura 1 - Componentes do motor de alocação.

Na pasta “evolution” estão os arquivos onde foram implementados os algoritmos genéticos.

Está implementada no arquivo *fitness.py* a função de fitness do motor. No cabeçalho do texto é possível observar três parâmetros, conforme a imagem:

```
PESO_GERENCIAL = 1
PESO_CARGA_HORARIA = 1
PESO_CONHECIMENTO = 1
```

Figura 2- Pesos da função fitness.

Essas variáveis correspondem aos pesos da função fitness (lembrando que o fitness é uma média ponderada entre a carga horária total alocada da função, a quantidade de correspondências em áreas de conhecimento da função com o funcionário e a experiência em cargos gerenciais pelo funcionário).

Portanto, para alterar os pesos dos fitness basta modificar as variáveis apontadas acima para os valores desejados.

No arquivo *initial_population.py* está contida a função de geração de população inicial. A quantidade da população inicial é dada pelo número de chamadas a essa função, no arquivo *main.py*. cada indivíduo é um vetor (array) de funções com um conjunto de servidores (atributo *servidores_alocados*).

As funções de reprodução estão escritas no arquivo *reproduction.py*. Há dois tipos de reprodução: O primeiro, chamado de *regularReproduction* seleciona metade da quantidade de funções do primeiro genitor, e metade das funções do segundo genitor e concatena esses dados para gerar um terceiro indivíduo.

Já na *radomReproduction*, a mesma quantidade de indivíduos é selecionada de ambos os genitores, entretanto a seleção é feita de forma aleatória, em contraste com a *regularReproduction.*

No arquivo *mutation.py* está a função de mutação. No cabeçalho, a variável:

```
NUMERO_MAX_MUTACOES = 20
```

Figura 3- Variável de número de mutações

Que armazena a quantidade de tentativas de mutações. A mutação ocorrerá **se e somente se** a troca de indivíduos não gerar conflitos (seja de áreas de conhecimento, prática gerencial ou carga horária).

Os arquivos das pastas *models* e *parser* contém as implementações das entidades “servidor” e “função”, e o “parseamento” dos dados advindos das planilhas, respectivamente. É importante notar que, no arquivo *função.py*, a quantidade de slots de cada função está “marretada” em 5, e para que os novos valores sejam lidos da planilha, deve-se adaptar o código, conforme os demais atributos.

Finalmente, no arquivo *main.py* estão as chamadas das demais funções. Os parâmetros ajustáveis para a execução são:

```
TARGET = 0.7
RANDOM_WHEN = 0.5
NUMBER_OF_ITERACTIONS = 5000
```

Figura 4- Parâmetros do arquivo main.py

O “target” indica, qual o menor valor de fitness admissível para que a função pare de gerar novos indivíduos, ou seja, indica um valor de resposta bom o suficiente para o algoritmo. Essa variável deve estar no intervalo [0, 1].

RANDOM_WHEN é a variável que recebe o valor de fitness a partir do qual o método de reprodução será aleatório.

NUMBER_OF_ITERATIONS indica a quantidade de iterações que o algoritmo executará, caso o mínimo fitness aceitável não seja alcançado.

Da execução:

Para executar a solução, basta abrir um prompt de comando e executar a seguinte instrução:

```
python3 main.py funcoes.csv servidores.csv
```

Figura 5- Comando de execução do algoritmo

O comando *pyhton3* indica qual versão da linguagem python deve ser utilizada. O primeiro parâmetro indica qual arquivo será executado (*main.py)*. Os demais parâmetros (*funcoes.csv* e *servidores.csv*) correspondem aos arquivos de informações das funções e servidores, respectivamente. **Importante: os parâmetros dessa instrução devem ser enviados nessa ordem, os arquivos com informações de funções e servidores devem estar no formato .csv e na mesma formatação do documento fornecido pela ATI durante o andamento do projeto.**

Esse comando deverá gerar as seguintes informações no prompt:

```
0.3988034188034189 0.3962393162393163
4984
0.3988034188034189 0.3962393162393163
4985
0.3988034188034189 0.3962393162393163
4986
0.3988034188034189 0.3962393162393163
4987
0.3988034188034189 0.3962393162393163
4988
0.3988034188034189 0.3962393162393163
4989
0.3988034188034189 0.3962393162393163
4990
0.3988034188034189 0.3962393162393163
4991
0.3988034188034189 0.3962393162393163
4992
0.3988034188034189 0.3962393162393163
4993
0.3988034188034189 0.3962393162393163
4994
0.3988034188034189 0.3962393162393163
4995
0.3988034188034189 0.3962393162393163
4996
0.3988034188034189 0.3962393162393163
4997
0.3988034188034189 0.3962393162393163
```

Figura 6- Retorno do comando Python3.

A coluna com valores decimais à esquerda corresponde ao indivíduo com o melhor fitness gerado pelo algoritmo. A coluna com valores decimais à direita corresponde ao indivíduo com o pior fitness gerado pelo algoritmo. O número crescente entre cada par de valores de fitness corresponde ao número da iteração do algoritmo.

O algoritmo deverá gerar como saída o documento *resposta.csv*, que contém os dados da função (Código, nome, carga horária), servidores alocados e fitness de cada alocação.

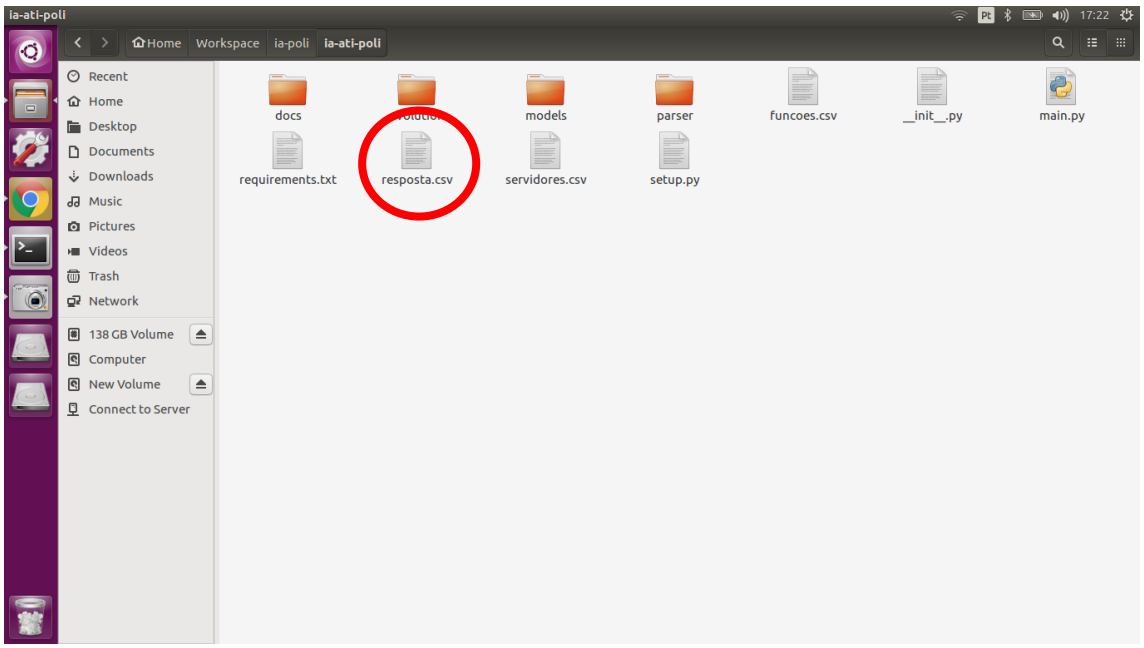

Figura 7- Arquivo gerado após a execução do algoritmo.