mod_evol_v1

August 26, 2019

1 Eficiência em Processos Cooperativos

Neste projeto, estudamos a eficiência de um grupo de agentes (pessoas) em resolver um dado problema. A ideia é avaliar o que é melhor: cada pessoa resolver o problema separadamente ou as pessoas conversarem entre si. Para isso calculamos o tempo necessário para o primeiro agente resolver o problema.

[3]:
```
import numpy as np
```

M = número de pessoas do grupo. N = número de bits de cada pessoa. Cada pessoa tem um vetor binário de N componentes formados por 0 ou 1.

[4]:
```
def cria_estado_inicial(M,N):
    lista = []
    for i1 in range(M):
        phi = np.random.randint(low = 0, high = 2, size = N)
        lista.append(phi)
    return lista
```

Ao longo da evolução temporal cada pessoa pode sofrer mutação com uma probabilidade p, na qual um bit aleatório é alterado. Adicionalmente com probabilidade $1 - p$ cada agente pode copiar um bit diferente de um outro agente desde q esse tenha uma fitness melhor.

[5]:
```
def mutacao(lista,ag,N):
    rn = np.random.randint(low = 0, high = N, size = 1)
    if lista[ag][rn] == 1:
        lista[ag][rn] = 0
    else:
        lista[ag][rn] = 1
    return lista
```

[6]:
```
def interacao(lista,ag1,ag2,M,N):
    lista2 = []
    for i1 in range(N):
        if lista[ag1][i1] != lista[ag2][i1]:
            lista2.append(i1)
    nid = len(lista2)
    if nid > 0:
        rn = int(nid*np.random.random())
        if lista[ag1][rn] == 1:
```

```
            lista[ag1][rn] = 0
        else:
            lista[ag1][rn] = 1
    return lista
```

A fitnnes, ou métrica, é definida como sendo o número de bits iguais a 1 no vetor do agente.

[7]:
```
def calc_metrica(lista,ag,N):
    soma = 0
    for i1 in range(N):
        soma += lista[ag][i1]
    return soma
```

Função que avalia se algum agente encontrou a solução, que é chegar na fitness máxima.

[8]:
```
def avalia(lista):
    flag = False
    M = len(lista)
    N = len(lista[0])
    for i1 in range(M):
        fit = calc_metrica(lista,i1,N)
        if fit == N:
            flag = True
            break
    return flag
```

Parâmetros iniciais.

[15]:
```
M = 20
N = 10
p = 0.5
lista_phi = cria_estado_inicial(M,N)
```

[16]:
```
for i1 in range(M):
    print('agente ',i1,':',lista_phi[i1])
```

```
agente  0 : [1 1 0 1 0 1 0 1 1 1]
agente  1 : [1 1 0 1 0 1 1 1 1 1]
agente  2 : [0 0 1 0 0 1 0 0 1 1]
agente  3 : [1 1 1 0 0 0 0 1 0 1]
agente  4 : [0 1 0 1 0 0 0 0 1 1]
agente  5 : [1 0 0 1 1 0 1 1 1 1]
agente  6 : [1 0 1 0 0 1 1 1 0 0]
agente  7 : [0 1 0 0 0 1 0 1 0 1]
agente  8 : [1 0 1 1 1 0 1 0 1 0]
agente  9 : [1 1 1 1 0 0 0 1 1 0]
agente  10 : [1 0 0 0 1 0 1 1 1 0]
agente  11 : [0 0 0 1 1 1 1 0 1 1]
agente  12 : [1 0 1 1 1 1 0 0 0 0]
```

```
agente  13 : [1 0 1 0 1 1 0 1 1 1]
agente  14 : [0 1 1 1 1 0 0 1 0 0]
agente  15 : [1 1 1 0 1 1 1 1 1 0]
agente  16 : [1 1 0 1 0 1 0 0 1 1]
agente  17 : [1 0 1 0 1 1 1 1 0 1]
agente  18 : [1 1 0 1 1 1 0 1 1 0]
agente  19 : [0 0 1 0 1 1 0 1 0 1]
```

Cálculo do tempo para alguém encontrar a solução.

[17]:
```
lista_phi = cria_estado_inicial(M,N)
flag = False
tempo = 0
while not flag:
    flag = avalia(lista_phi)
    tempo += 1
    for i1 in range(M):
        ag0 = int(M*np.random.random()) #agente aleatorio
        rn = np.random.random() #numero aleatorio entre 0 e 1
        if rn < p:
            #ele vai sofrer mutacao
            lista_phi = mutacao(lista_phi,ag0,N)
        else:
            ag2 = int(M*np.random.random()) #agente a ser copiado
            fit0 = calc_metrica(lista_phi,ag0,N)
            fit2 = calc_metrica(lista_phi,ag2,N)
            if fit2 > fit0: lista_phi = interacao(lista_phi,ag0,ag2,M,N)

print(tempo)
```

```
30
```

Como este modelo é probabilístico, é necessário calcular o tempo de solução várias vezes então tirar a média.

[20]:
```
media_tempo = 0.0
S = 100
for i2 in range(S):
    lista_phi = cria_estado_inicial(M,N)
    flag = False
    tempo = 0
    while not flag:
        flag = avalia(lista_phi)
        tempo += 1
        for i1 in range(M):
            ag0 = int(M*np.random.random()) #agente aleatorio
            rn = np.random.random() #numero aleatorio entre 0 e 1
            if rn < p:
                #ele vai sofrer mutacao
                lista_phi = mutacao(lista_phi,ag0,N)
```

```
            else:
                ag2 = int(M*np.random.random()) #agente a ser copiado
                fit0 = calc_metrica(lista_phi,ag0,N)
                fit2 = calc_metrica(lista_phi,ag2,N)
                if fit2 > fit0: lista_phi = interacao(lista_phi,ag0,ag2,M,N)
    media_tempo += tempo
media_tempo = media_tempo / S
print(media_tempo)
```

```
92.99
```

[18]:
```
for i1 in range(M):
    print('agente ',i1,':',lista_phi[i1])
```

```
agente  0 : [1 0 0 0 0 0 1 0 1 0]
agente  1 : [1 1 1 1 1 1 1 1 1 1]
agente  2 : [1 1 1 0 1 1 0 1 0 0]
agente  3 : [0 0 1 0 1 0 1 0 0 0]
agente  4 : [1 1 1 1 1 1 0 0 0 0]
agente  5 : [1 1 1 1 1 0 0 0 0 0]
agente  6 : [0 0 1 0 1 0 0 1 0 1]
agente  7 : [0 1 0 0 0 0 1 0 1 1]
agente  8 : [0 1 1 1 0 1 1 1 0 0]
agente  9 : [0 1 1 1 0 0 0 0 0 0]
agente  10 : [1 0 0 0 0 0 1 1 0 0]
agente  11 : [1 1 0 1 0 1 0 0 1 1]
agente  12 : [1 1 0 1 1 0 0 0 0 0]
agente  13 : [0 0 1 0 0 0 1 1 0 1]
agente  14 : [1 0 1 0 1 0 1 1 1 0]
agente  15 : [1 1 1 0 0 1 1 0 0 1]
agente  16 : [1 0 1 0 1 1 0 0 0 0]
agente  17 : [0 0 0 0 1 0 1 1 0 0]
agente  18 : [1 0 1 0 1 0 0 1 0 0]
agente  19 : [1 1 0 1 1 0 0 0 1 1]
```

Repare que o agente 1 encontrou a solução.