Modelação Numérica 2017

Aula 21, 10/Maio

- Estimativa de parâmetros e optimização: Algoritmo genético.

http://modnum.ucs.ciencias.ulisboa.pt

Algoritmo genético

- Neste algoritmo utilizam-se conceitos da genética para resolver um problema de otimização.
- Cada iteração do algoritmo simula uma nova geração numa população de organismos em que existe diversidade genética.
- A diversidade é produzida no início, de forma aleatória, e reforçada em cada geração com mutação.
- A transição entre gerações inclui:
  1. Seleção do indivíduo com melhor desempenho (menor função de custo).
  2. Eliminação do pior elemento.
  3. Cruzamento aleatório para produzir os novos (N-2) membros.
  4. Introdução de um (ou mais) mutantes.

Algoritmo genético

- Como realizar a mutação?
  - Escolher um novo elemento aleatoriamente (= ao início)
  - Perturbar um membro da geração anterior por uma "pequena" perturbação (perturbar o melhor?)
- Como selecionar os progenitores?
  - Aleatoriamente (impedir autocruzamentos)
  - Na proporção inversa da função de custo: os "melhores" têm mais descendentes
- Como acasalar?
  - Média aritmética dos parâmetros
  - Média pesada

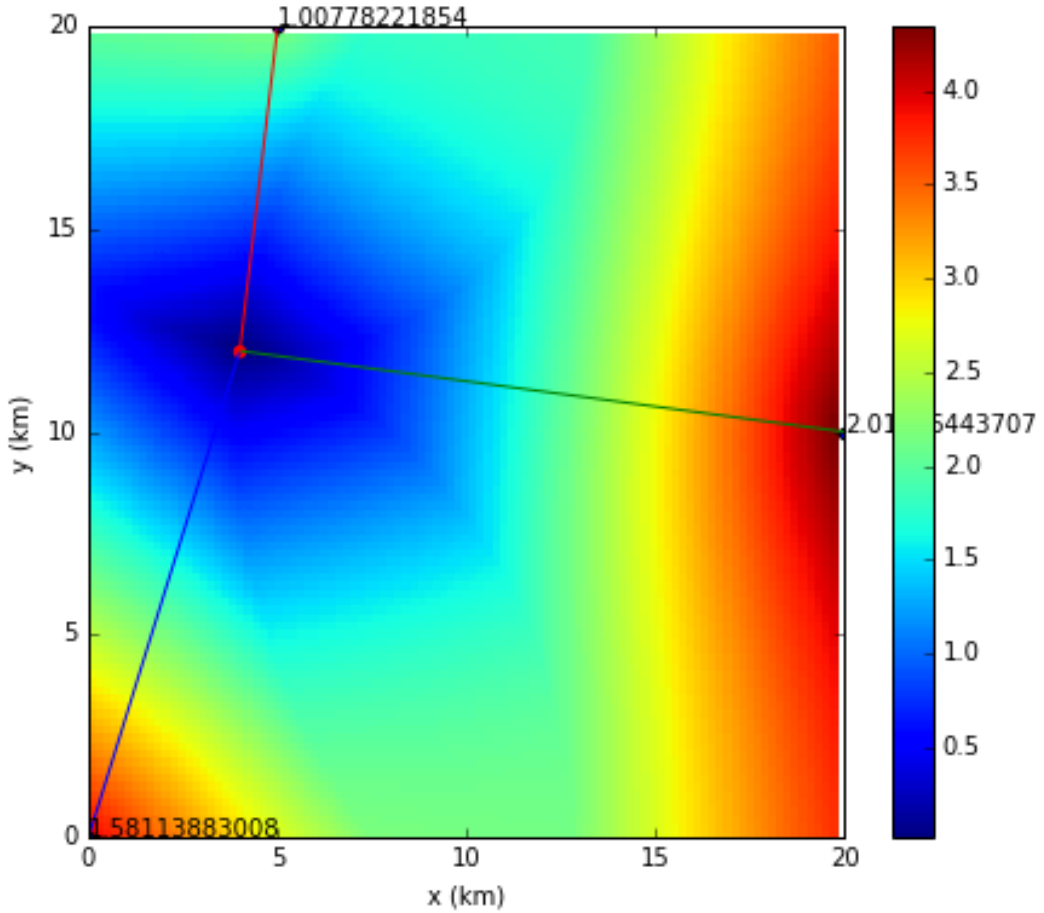
... Não existe uma teoria para nos ajudar.

Parâmetros ajustáveis

- Domínio da solução (mínimo e máximo de cada parâmetro a ajustar/ optimizar)
- Dimensão da população (~10× Número de variáveis a ajustar)
- Critérios de paragem:
  - Número de gerações (Número máximo de iterações)
  - Target para a função de custo J (se for conhecido)

Problema

- Localizar uma fonte sísmica em 2 dimensões (x,y) com velocidade constante.



```
import matplotlib.pyplot as plt
import numpy as np

#%%
plt.rcParams['figure.figsize'] = 10, 6

################## Função Optimize

def optimize_gen():
    global firstcall        # variáveis globais
    global mut1, mut2, mutant
    global xmin,xmax,ymin,ymax

    firstcall=0;

    outITER=10;          # write output
    maxPOP=50;         # maximum size of population
    maxGEN=100;          # maximum number of generations
    Jmin=1e-20;          # custo mínimo (critério de paragem)

    # inicializa a função de custo
    iniCOST();

    # domínio espacial
    xmin=0.;xmax=20000.;
    ymin=0.;ymax=20000.;
    vmin=np.array([xmin,ymin]);
    vmax=np.array([xmax,ymax]);

    # mutações
    mut1=1;          # mutação local
    mut2=0;          # mutação global
    mutRANGE=0.1;    # amplitude da mutação local (em fracção do domínio)
    mutant=(vmax-vmin)*mutRANGE;    # amplitude da mutação local
```

```
    [V,J,kGEN]=optim(vmin,vmax,Jmin,maxGEN,maxPOP,outITER);
    print(V,J)

    plotCOST(xmin,xmax,ymin,ymax,V[0],V[1]);
    plt.title(u'No Gerações:'+str(maxGEN)+' Pop:'+str(maxPOP)+' MutantRG:'+str(mutRANGE)+'@'+str
```

```
#%% Problema directo: calcular tempos de chegada às estações

def iniCOST():
    global xE, yE, tE       # observations
    global cs               # constants (velocidade da propagação das ondas)
    global xF, yF

    cs=8000;
    xF=4000; yF=12000;      # fonte
    xE=np.array([0,20000,5000])
    yE=np.array([0,10000,20000])
    distE=np.sqrt((xF-xE)**2+(yF-yE)**2)
    tE=distE/cs;
```

```
#%% Função de custo

def cost(V):
    global xE, yE, tE       # observations
    global cs               # constantes

    if len(V.shape)==2:
        n=V.shape;
        num=n[1]
    else:
        num=1

    custo=np.zeros(num);

    for k in range(num):
        if len(V.shape)==2:
            X=V[0,k];
            Y=V[1,k];       # parameters to optimize
        else:
            X=V[0];
            Y=V[1];       # parameters to optimize

        dist=np.sqrt((X-xE)**2+(Y-yE)**2);
        erro=dist/cs-tE;
        custo[k]=np.sum(np.abs(erro));
    return custo
```

```
#%% Optimização

def optim(vmin, vmax, Jmin, maxGEN, maxPOP, outITER):

    global mut1, mut2, mutant, P
    global xmin,xmax,ymin,ymax

    ndim = len(vmin)
    P = np.zeros([ndim,maxPOP]);
    for iP in range(maxPOP):
        P[:,iP]= vmin + (vmax-vmin)*np.random.rand(ndim);     # população inicial (aleatória)
    JP=cost(P)

    # ordena (melhor primeiro)
    JS=np.sort(JP);
    IS=np.argsort(JP);

    # actualiza o melhor membro
    J=JS[0];            # custo
    V=P[:,IS[0]];       # população
    bestJ=J;
    bestV=V;

    # inicialização dos índices de contagem
    iGEN=0;
    kGEN=1;

    # ciclo que corre as várias gerações
    while iGEN<maxGEN and J>Jmin:
        print(iGEN,J)

        P[:,0]=V;          # save best

        # escolha dos progenitors
        for iP in range(1,maxPOP):
            # escolher dois índices aleatórios para os progenitores a seleccionar
```

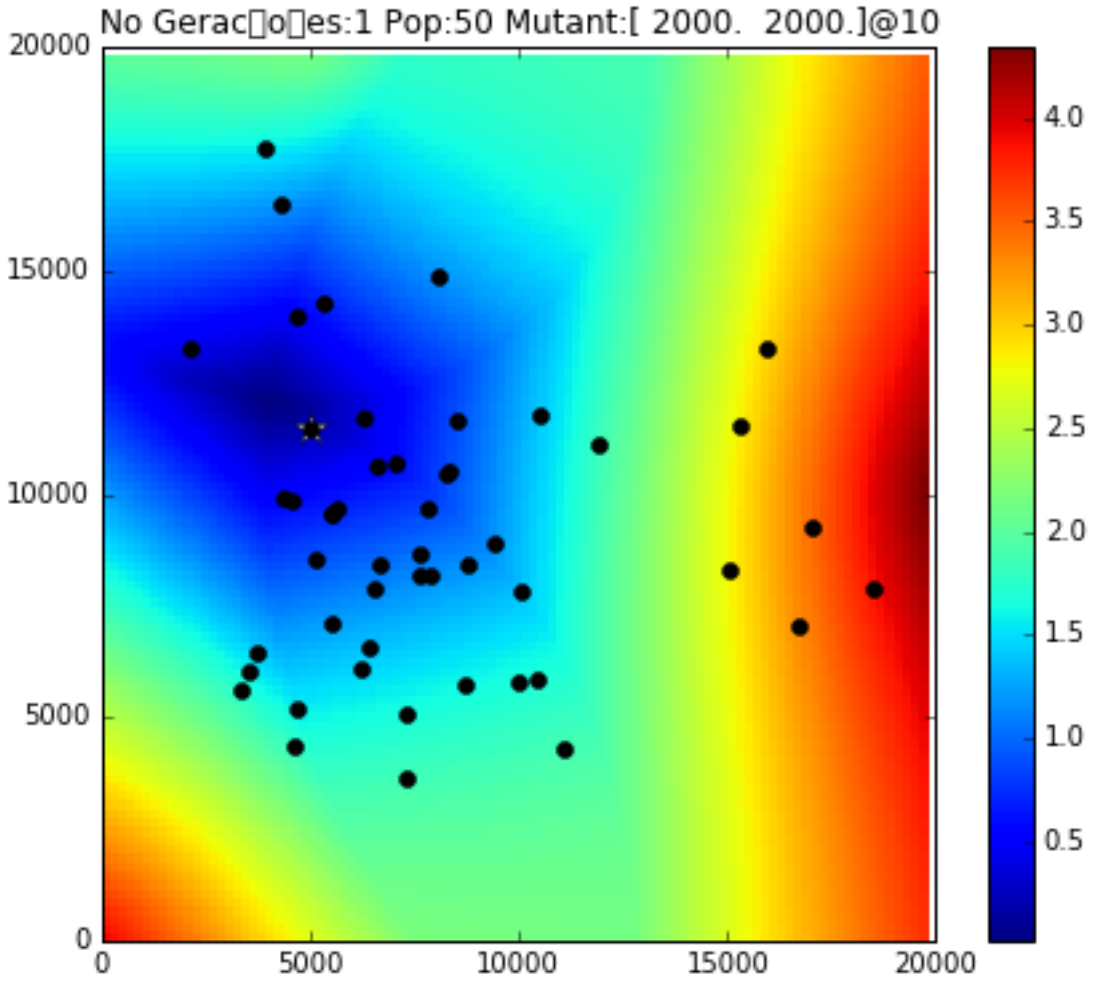
```
# escolha dos progenitors
for iP in range(1,maxPOP):
    # escolher dois índices aleatórios para os progenitores a seleccionar
    rr1=int(np.ceil(np.random.rand()*(maxPOP-1)));     # maxPOP-1 potenciais progenitores
    rr2=int(np.ceil(np.random.rand()*(maxPOP-1)));
    while rr1==rr2:
                rr2=np.ceil(np.random.rand()*(maxPOP-1));

    # acasalamento
    P[:,iP] = 0.5*P[:,int(IS[rr1])] + 0.5*P[:,int(IS[rr2])];

# mutante próximo do melhor
if mut1:
    Pmut = V + np.random.rand(ndim);
    while (Pmut<vmin).all() or (Pmut>vmax).all():
        Pmut = V + np.random.rand(ndim);
    P[:,maxPOP-2] = Pmut

# mutante global
if mut2:
    P[:,maxPOP-1] = vmin + (vmax-vmin)*np.random.rand(ndim);

# custo da nova população
JP=cost(P);

# ordena (melhor primeiro)
JS=np.sort(JP);
IS=np.argsort(JP);

# actualiza o melhor membro
J=JS[0];
V=P[:,IS[0]];

# actualiza a geração
iGEN=iGEN+1;
```

```
        if J<bestJ:
            bestJ=J;
            bestV=V;
            kGEN=iGEN;

#         if iGEN==1 or np.mod(iGEN,outITER)==0:
        if iGEN in [1,2,3,4,5,10]+range(10,maxGEN,10):
            print('iGEN='+ str(iGEN) +' kGEN='+str(kGEN)+' J='+str(np.log(J))+' V='+str(V)+' bestV

            plotCOST(xmin,xmax,ymin,ymax,V[0],V[1]);
            plt.title(u'No Gerações:'+str(iGEN)+' Pop:'+str(maxPOP)+' Mutant:'+str(mutant)+'@'+str
            plt.savefig('p2-gen-igen'+str(iGEN)+'.png')


    return  [V, J, kGEN]
```

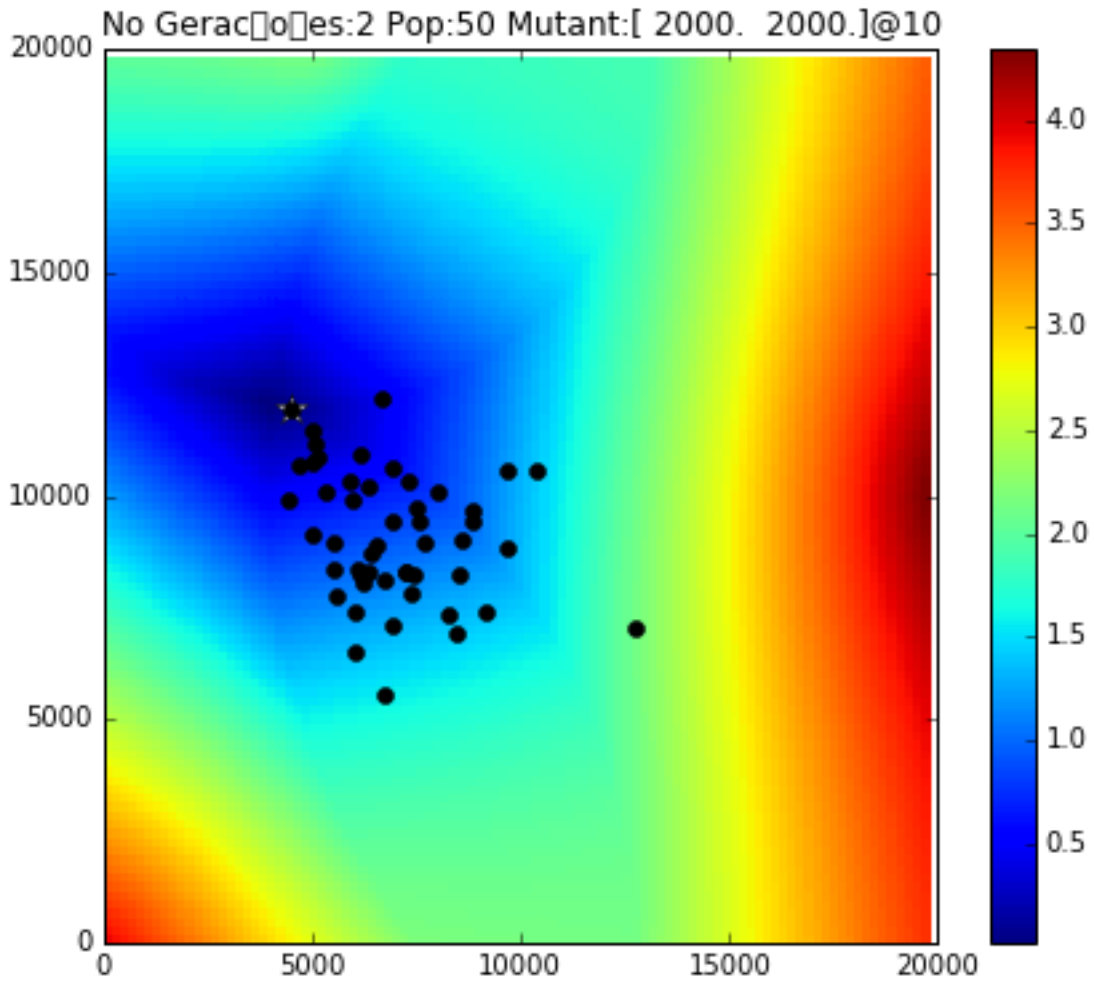
```
#%% plot

def plotCOST(xmin,xmax,ymin,ymax,X,Y):
    global firstcall, P
    global xx,yy,CC
    global xF, yF
    global xE, yE, tE


    if firstcall == 0:
        firstcall=1;

        nx=101;ny=101;
        xx=np.zeros([nx,ny]);
        yy=np.zeros([nx,ny]);
        CC=np.zeros([nx,ny]);

        x=np.arange(xmin,xmax,(xmax-xmin)/nx)
        y=np.arange(ymin,ymax,(ymax-ymin)/ny)

        for ix in range(nx):
            for iy in range(ny):
                xx[ix,iy]=x[ix];
                yy[ix,iy]=y[iy];
                CC[ix,iy]=cost(np.array([x[ix], y[iy]]));

            #%% Plot

        plt.rcParams['figure.figsize'] = 7,6
        plt.close()

        plt.pcolor(xx/1e3,yy/1e3,CC)        # matriz dos tempos de propagação
        plt.colorbar()
        plt.scatter(xF/1e3,yF/1e3, color='r');         # fonte
```

```
        # estações
        for iE in range(len(xE)):
            plt.plot([xE[iE]/1000, xF/1000], [yE[iE]/1000, yF/1000])
            plt.scatter(xE[iE]/1e3,yE[iE]/1e3)
            plt.text(xE[iE]/1e3,yE[iE]/1e3, str(tE[iE]))

        plt.xlabel('x (km)')
        plt.ylabel('y (km)')
        plt.xlim(np.array([xmin, xmax])/1e3)
        plt.ylim(np.array([ymin, ymax])/1e3)

        plt.savefig('p1-TempoPropagacao.png')


    plt.rcParams['figure.figsize'] = 7,6
    plt.close()

    plt.pcolor(xx,yy,CC)
    plt.colorbar()
    plt.plot(X,Y, '*', markersize=12, markerfacecolor='white');
    plt.plot(P[0,:],P[1,:], 'o', markersize=6, markerfacecolor='black');


#%% Run

optimize_gen()
```

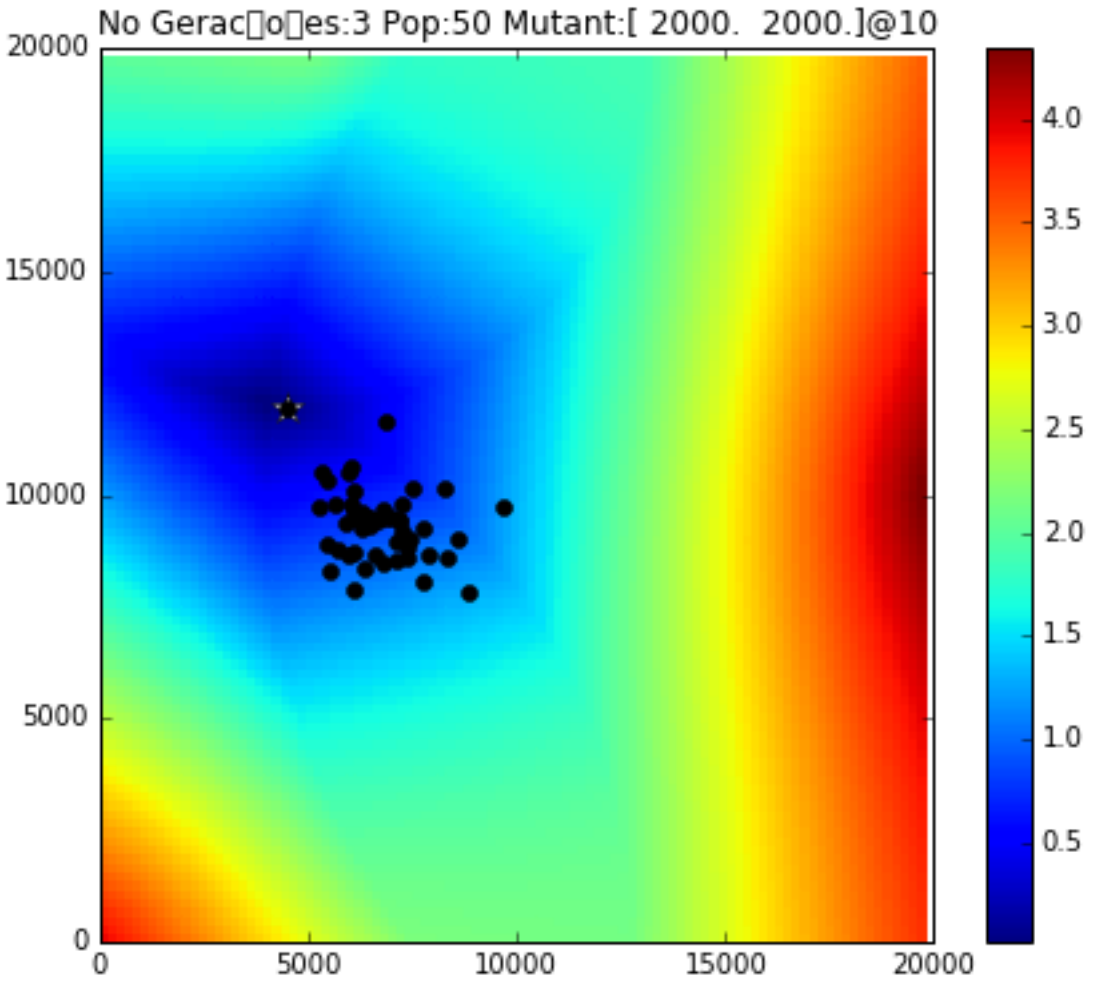
Exemplo:


No Gerac□o□es:1 Pop:50 Mutant:[2000. 2000.]@10
20000
15000
10000
5000
0
0
5000
10000
15000
20000
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5

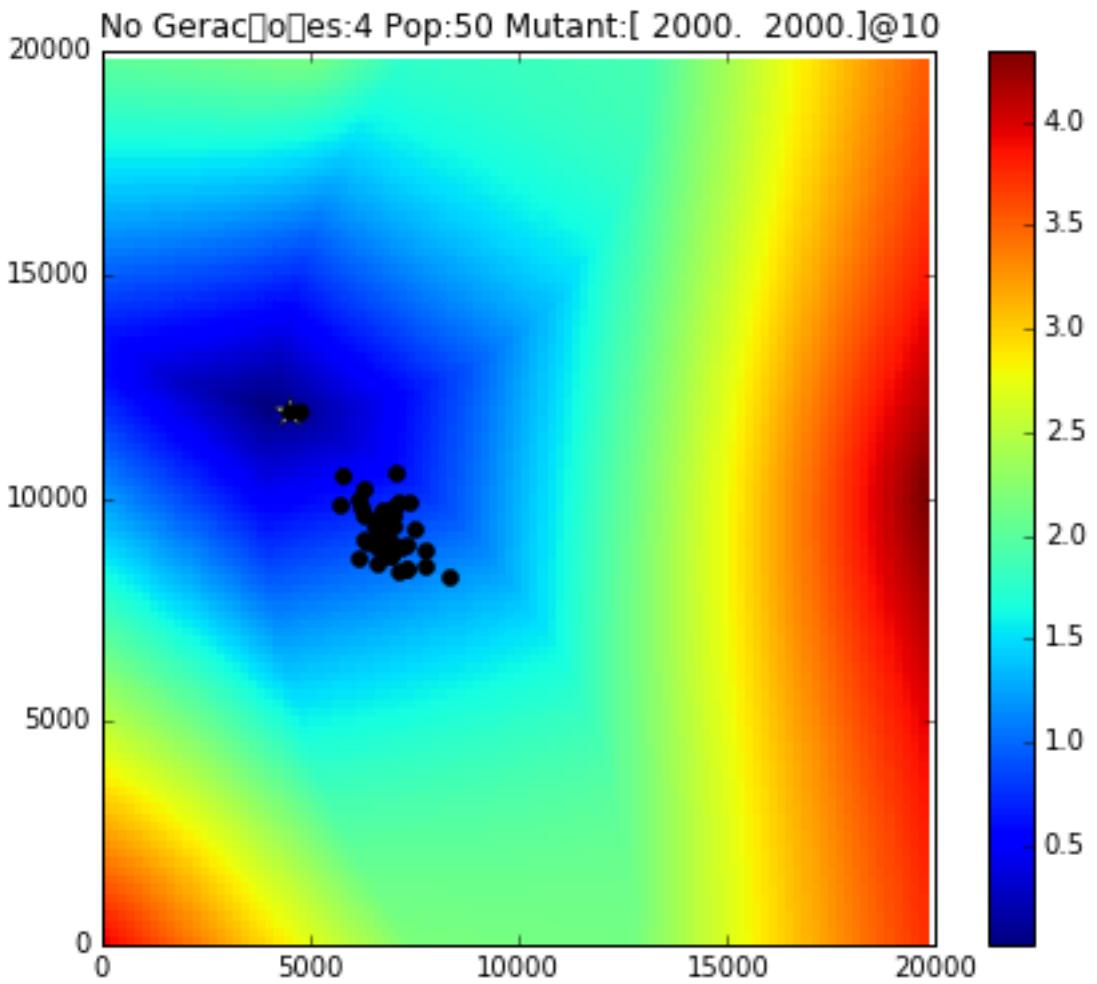
Exemplo:


No Gerac□o□es:2 Pop:50 Mutant:[2000. 2000.]@10
20000
15000
10000
5000
0
0
5000
10000
15000
20000
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5

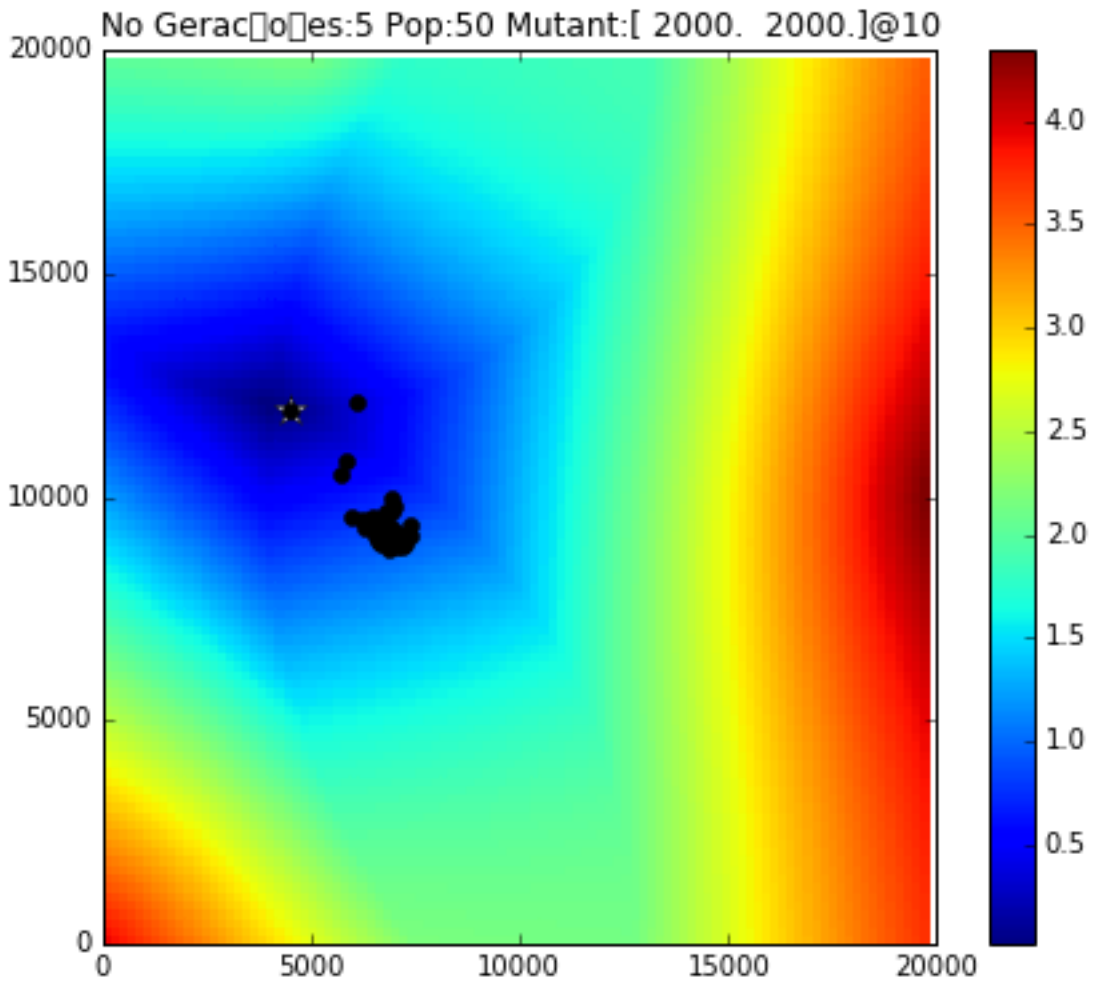
Exemplo:


No Gerac□o□es:3 Pop:50 Mutant:[2000. 2000.]@10
20000
15000
10000
5000
0
0
5000
10000
15000
20000
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5

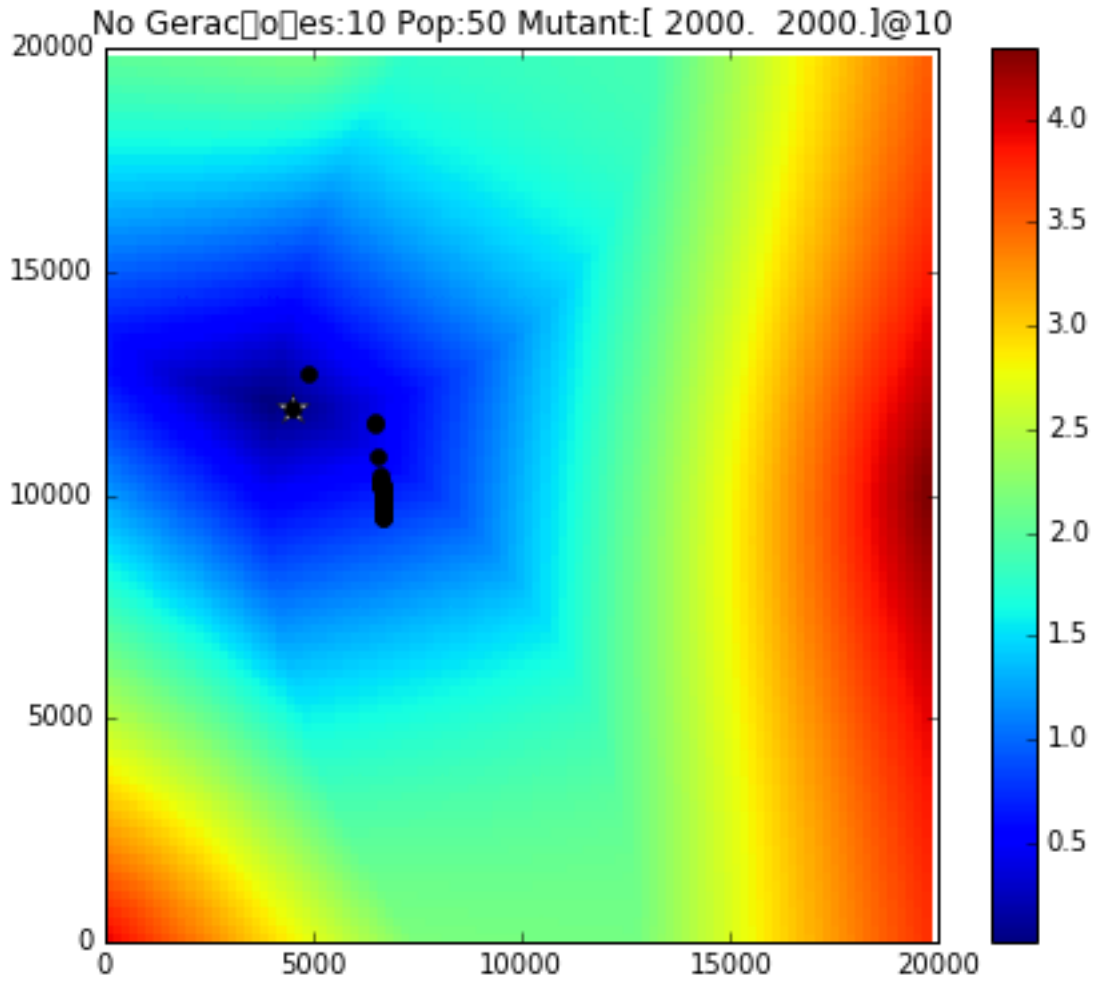
Exemplo:


No Gerac□o□es:4 Pop:50 Mutant:[2000. 2000.]@10
20000
15000
10000
5000
0
0
5000
10000
15000
20000
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5

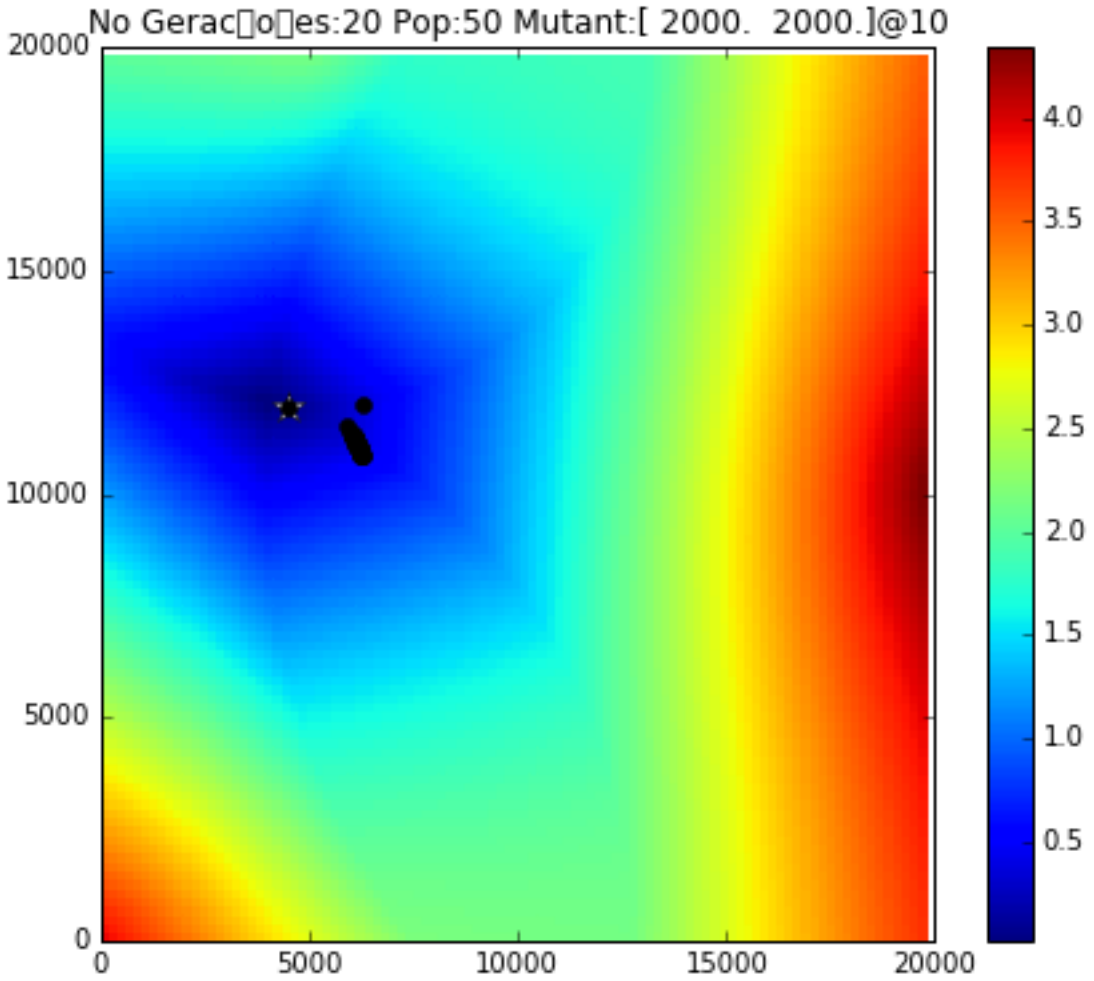
Exemplo:


No Geraç□o□es:5 Pop:50 Mutant:[2000. 2000.]@10
20000
15000
10000
5000
0
0
5000
10000
15000
20000
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5

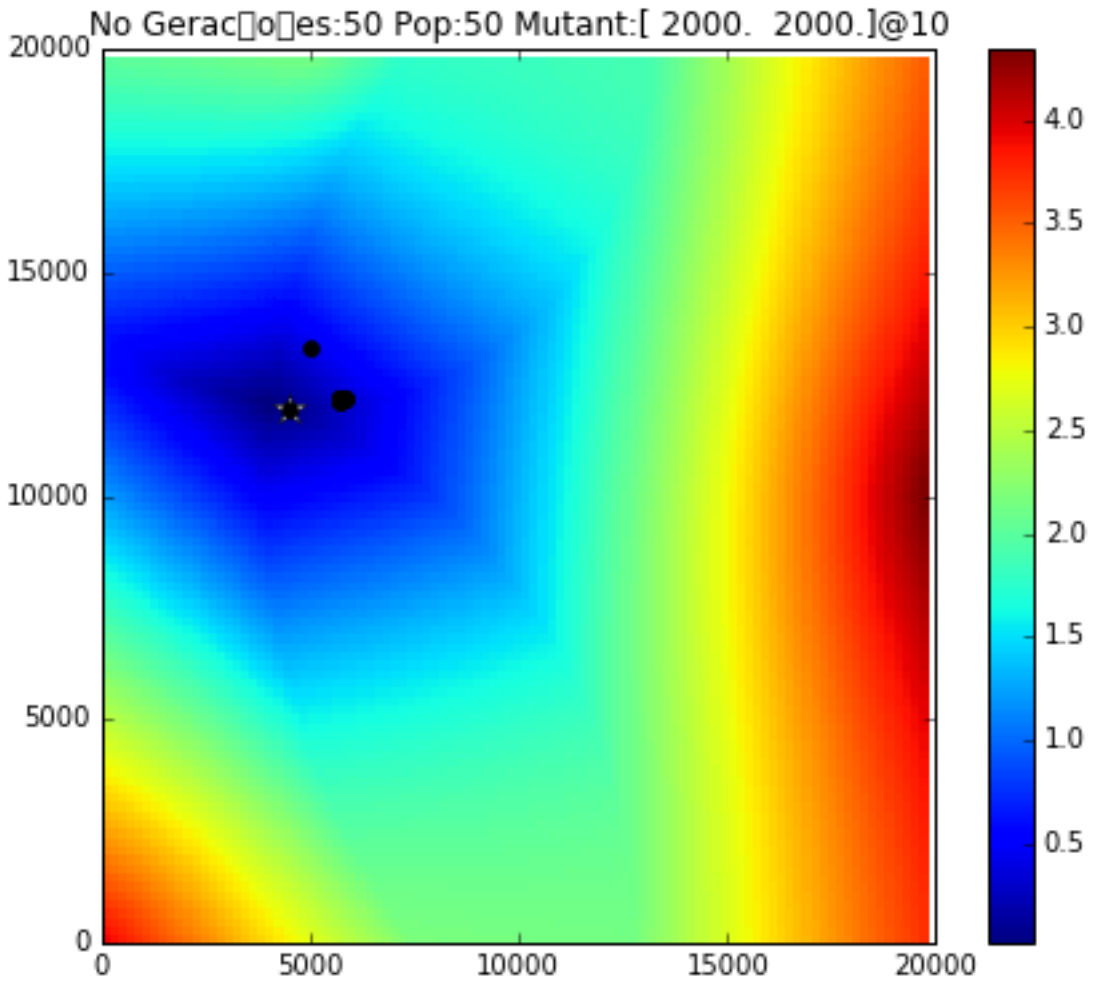
Exemplo:


No Gerac□o□es:10 Pop:50 Mutant:[2000. 2000.]@10
20000
15000
10000
5000
0
0
5000
10000
15000
20000
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5

Exemplo:


No Gerac□o□es:20 Pop:50 Mutant:[2000. 2000.]@10
20000
15000
10000
5000
0
0
5000
10000
15000
20000
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5

Exemplo:


No Geraç□o□es:50 Pop:50 Mutant:[2000. 2000.]@10
20000
15000
10000
5000
0
0
5000
10000
15000
20000
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5