UNIVERSIDADE FEDERAL DO CEARÁ - CAMPUS SOBRAL

CURSO ENGENHARIA DA COMPUTAÇÃO

DISCIPLINA: INTELIGÊNCIA COMPUTACIONAL

PROFESSOR: JARBAS JOACI DE MESQUITA SÁ JUNIOR

RELATÓRIO

Andressa Gomes Moreira – 402305

Sobral – CE
2020

1. Questão 01

As soluções referentes à questão 01 encontram-se no arquivo questao01.sce, na qual foi-se utilizado o software Scilab versão 6.1.0. Dessa forma, implementou-se a rede neural ELM para plotar o gráfico de dispersão contendo os dados do arquivo two_classes. Assim sendo, os dados foram divididos, na qual as duas primeiras colunas representam os atributos e a terceira a classe. Ademais, foi-se definido os valores para algumas variáveis, como a quantidade de atributos de um vetor (p), a quantidade de neurônios da camada oculta (q), além do tamanho da amostra.

```
// Carregando a base de dados
data = fscanfMat("two_classes.dat");

entrada = data(:, 1:2)'      // Dados de entrada: 1° e 2° coluna
rotulos = data(:, 3)'        // Rótulos: 3° coluna

// Variáveis necessárias
p = 2;                         // p = Quantidade de atributos em um vetor
q = 40;                        // q = Número de neurônios da camada oculta.
amostra = length(entrada)/2;   // Amostra = 1000

x = [(-1)*ones(1,amostra); entrada];        // Adicionar o bias = -1
```

Figura 1 – Base de dados e variáveis necessárias

Outrossim, inicializou-se os pesos dos neurônios ocultos de forma aleatória, na qual trata-se de uma matriz de pesos (W), com q linhas e p+1 colunas. Ademais, iniciou-se o treinamento da rede, na qual o primeiro passo foi calcular as ativações dos neurônios da camada oculta (u). Em seguida, as saídas correspondentes (z) são calculadas, assumindo a função de ativação Logística.

```
// Fase 1: Iniciar aleatóriamente os pesos da camada oculta
W = rand(q, p+1,'normal');      // W = Matriz de pesos da camada oculta

// Fase 2: Acúmulo da saída dos neurônios
u = W * x;                      // u(t) = Vetor de ativação

//Cálculo da função de ativação (Logística)
z = 1./(1+(exp(-u)))            // Vetores produzidos pela camada oculta
z = [(-1)*ones(1,amostra); z];  // Adicionar o bias = -1
```

Figura 2 – Fase 1: Iniciação dos pesos da camada oculta e Fase 2: Acúmulo da saída dos neurônios.

Ademais, utilizou-se o método dos mínimos quadrados calcular os pesos dos neurônios de saída, finalizando assim, a etapa de treinamento da rede.

```
//Fase 3: Cálculo dos Pesos dos Neurônios de saída
M = rotulos*z'*((z*z'))^(-1) ;          // Método dos mínimos quadrados
```

Figura 3 – Cálculo dos pesos dos neurônios de saída.

Por fim, após o treinamento da rede, plotou-se a superfície de decisão obtida com o uso de todas as amostras como treinamento. Desse modo, utilizou-se cores diferentes para diferenciar os dados das classes. Na qual, a cor azul representa os atributos cuja classe é igual a 1 (referente às linhas 1 até 500 da base de dados) enquanto que a cor verde representa os atributos cuja classe é igual a -1 (referente às linhas 500 até 1000 da base de dados).

```
// Separando as classes
classe1 = data(1:500,1:2)       // Classe 1: Rótulo = 1
classe2 = data(501:1000,1:2)    // Classe 2: Rótulo = -1

plot(classe1(:,1),classe1(:,2), 'b*')   // Plotagem dos dados da classe 1: Cor Azul
plot(classe2(:,1),classe2(:,2), 'g*')   // Plotagem dos dados da classe 2: Cor Verde
```

Figura 4 – Plotagem dos dados de acordo com a classe.

Em seguida, foi determinado mil pontos para compor o plano. Ademais, existe uma região cujo resultado é bastante inconclusivo. Assim, nessa região os pontos não possuem classe definida e os rótulos estão próximos a zero, logo, essa será a superfície de decisão que separa as classes.

```
  intevalo_x1 = linspace(0, 5, 1000);
  intevalo_x2 = linspace(0, 5, 1000);

  for aux1=1:1000
      for aux2=1:1000
          x_new = [-1 intevalo_x1(aux1) intevalo_x2(aux2)]' // Nova matriz de pontos
          u_new = W * x_new                                  // u(t) = Vetor de ativação
          z_new  = 1./(1+exp(-u_new))                        // Função de ativação (Logística)
          z_new  = [(-1)*ones(1,1); z_new];                  // Adicionar o bias = -1
          a_new = M*z_new                                    // Vetor de saídas

         if a_new < 0.001 & a_new > -0.001 then
             plot(intevalo_x1(aux1),intevalo_x2(aux2),'redd') // Plotagem da superfície de decisão
         end
      end
  end
```

Figura 5– Plotagem da superfície de decisão.

Por fim, pode-se observar a superfície de decisão (em vermelho) obtida com o uso de todas as amostras como treinamento usando a rede neural ELM e 40 neurônios ocultos.

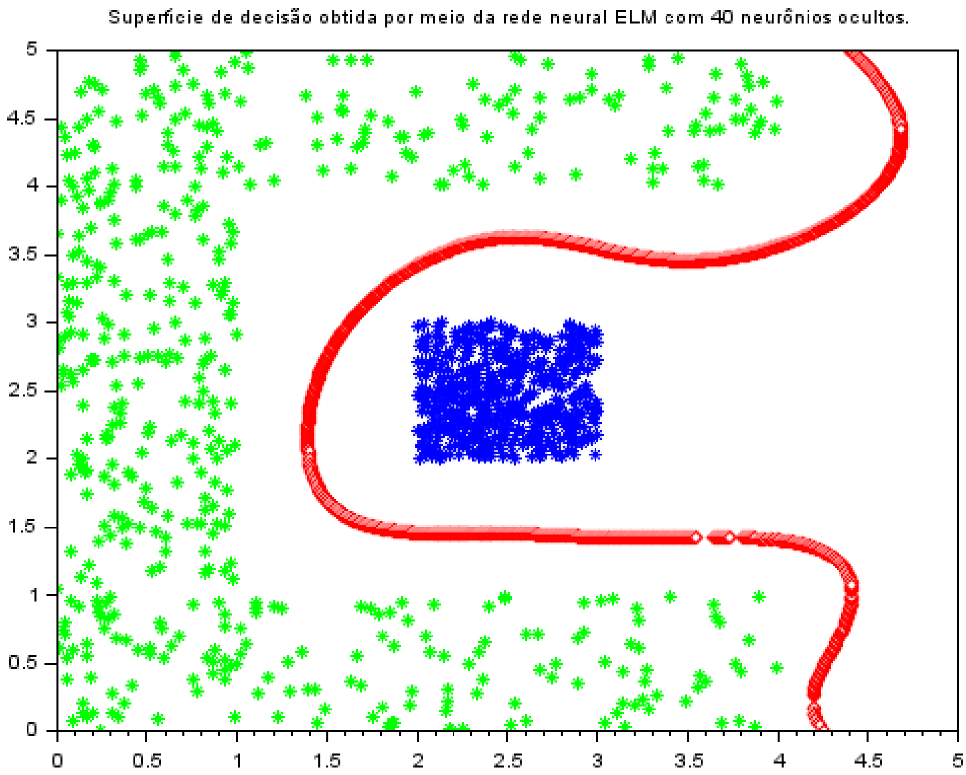

Figura 6 – Superfície de decisão.

2. Questão 02

As soluções referentes à questão 02 encontram-se no arquivo questao02.sce. Dessa forma, desenvolveu-se um algoritmo genético para achar o mínimo da função de Rosenbrock, na qual fez-se uso de uma população de 100 indivíduos, sendo cada um deles composto por um vetor binário de 20 bits. Assim sendo, a população inicial foi selecionada aleatoriamente, representada em forma binária e depois convertida para números. Ademais, utilizou-se uma função de avaliação para determinar a nota de um indivíduo, ou seja, a qualidade de um indivíduo para solução do problema.

```
individuos_x = matriz_individuos(i, 1:10);  // Composto pelos 10 primeiros bits da matriz de indivíduos
x_str = strcat(string(individuos_x))        // Concatenando e convertendo os bits para string

individuos_y = matriz_individuos(i, 11:20)  // Composto pelos 10 últimos bits da matriz de indivíduos
y_str = strcat(string( individuos_y))       // Concatenando e convertendo os bits para string

x(i) =bin2dec(x_str)                        // Converte de binário para decimal
y(i)= bin2dec(y_str)                        // Converte de binário para decimal

/* Convertendo bits para números, na qual x e y pertencem ao intervalo [-5 5]*/
x_real(i) = inf+(sup - inf)/((2^10)-1)*x(i) // Representação de x em real entre o intervalo [-5 5]
y_real(i) = inf+(sup - inf)/((2^10)-1)*y(i  // Representação de y em real entre o intervalo [-5 5]

nota(i) = (1 - x_real(i))^2 + 100*(y_real(i) - x_real(i)^2)^2     // Notas dos indivíduos
```

Figura 7 – Nota de um indivíduo

Ademais, para selecionar novo pais utilizou-se o método do torneio, na qual 4 indivíduos foram selecionados aleatoriamente para participar do torneio e foi escolhido aquele cuja nota era mais baixa, para assim determinar a matriz de pais.

```
  // Vamos selecionar "n" indivíduos que irão participar do torneio:
  participantes = matriz_individuos(1:qnt_participantes, 1:20);   // Dimensionando a matriz "participantes"
  nota_part = nota(1:qnt_participantes, :);                       // Dimensionando a matriz para notas dos participantes

  for i = 1:qnt_individuos

      for p = 1:qnt_participantes
          pos_indiv_aleat = ceil(rand()*qnt_individuos)                      // Gera uma posição aleatórias.
          participantes(p, :) = matriz_individuos(pos_indiv_aleat, 1:20);    // Seleciona aleatoriamente "n" indivíduos
      end                                                                    // para participar do torneio

      [valor indice] = min(nota_part)                          // Recebe os valores e os índices das menores notas
      matriz_individuos_pais(i, :) = participantes(indice, :)  // Recebe os pais que foram selecionados
  end
```

Figura 8 – Método do torneio

Após a seleção os pais, utilizou-se o operador de crossover, na qual determinou-se um ponto de corte, que é a posição entre dois genes, responsável por separar os pais em duas partes formando dois filhos, sendo o primeiro filho composto da parte esquerda do primeiro pai com a parte direita do segundo pai e o segundo filho é composto da concatenação das partes que sobraram.

```
   pontoCorte = int(rand()*linspace(1,19, 1))  // Seleciona um ponto de corte

   for aux = 1:2:qnt_individuos
       pontoCorte1 = pontoCorte + 1  // Ponto de corte: Vai separ os pais em duas partes

       // Para o filho 1:
       pos1_filho1 =  matriz_individuos_pais(aux, 1:pontoCorte);       // Parte esquerda do primeiro pai
       pos2_filho1 =  matriz_individuos_pais(aux+1, pontoCorte1:20);  // Parte direita do segundo pai.

       // Para o filho 2:
       pos1_filho2 =  matriz_individuos_pais(aux+1, 1:pontoCorte);    // Parte esquerda do segundo pai
       pos2_filho2 =  matriz_individuos_pais(aux, pontoCorte1:20);    // Parte direita do primeiro pai.

       filhosn(aux, :) = [pos1_filho1, pos2_filho1];      // Filho 1
       filhosn(aux+1, :) = [pos1_filho2, pos2_filho2];    // Filho 2
   end
```

Figura 9 – Operador de Crossover

Assim sendo, depois de compostos os filhos, utiliza-se o operador de mutação, na qual é necessário determinar um valor aleatório entre 0 e 1, se o valor for menor do que a probabilidade determinada (0,5%) o operador atua alterando o valor do gene em questão e determinando assim a nova população.

```
for i = 1:qnt_individuos
        num_sort = rand(1, 'uniform')
        colun_sort =(ceil(rand()*num_bit));

        if num_sort <= 0.005 then
             filhos_mut(i, colun_sort) = ~(filhos_mut(i, colun_sort));
        end
end
matriz_individuos = filhos_mut;
```

Figura 10 – Mutação

Logo, após completar 50 gerações determina-se as notas da nova população como também o valor e o índice responsáveis por minimizar a função Rosenbrock.

```
//Cálculo da função de avaliação para a nova população
nota(i) = (1 - x_real(i))^2 + 100*(y_real(i) - x_real(i)^2)^2  // Notas dos indivíduos da nova população

[valor_min indice_min]= min(nota)   // Valor e índice que minimizam a função Rosenbrock
```

Figura 11 – Mínimo da função de Rosenbrock

Por fim, plotou-se o gráfico 3D para exibir o resultado do algoritmo genético para achar o mínimo da função de Rosenbrock.

```
//Determinando a quantidade de pontos para a plotagem da superficie:
intervalo_x1 = linspace(-6,6,num_bit)
intervalo_x2 = linspace(-6,6,num_bit)

for i=1:num_bit
    for j=1:num_bit
        func_avali(i,j) = (1 - intervalo_x1(i))^2 + 100*(intervalo_x2(j) - intervalo_x1(i)^2)^2
    end
end

plot3d(intervalo_x1,intervalo_x2,func_avali)    // Plotagem da superficie
scatter3d(x_real,y_real,nota,'fill')            // Plotagem para última geração
```

Figura 12 – Plotagem do gráfico

Logo, determinou-se resultado do algoritmo genético que minimiza função para diferentes gerações. Assim, a tabela exibe o valor mínimo encontrado e os valores de “x” e “y” que minimizam a função de Rosenbrock:

Geração	Valor para x	Valor para y	Valor Mínimo
1	0.1221896	0.1221896	0.5107447
5	1.1681329	1.3343109	0.119616
10	1.2658847	1.6080156	0.0737768
20	1.1681329	1.3734115	0.0361487
35	1.3636364	1.8621701	0.1329421
50	1.0508309	1.0899316	0.0230728

Tabela 01 – Resultados

Por fim, os gráficos referentes aos resultados achados na tabela são exibidos a seguir:

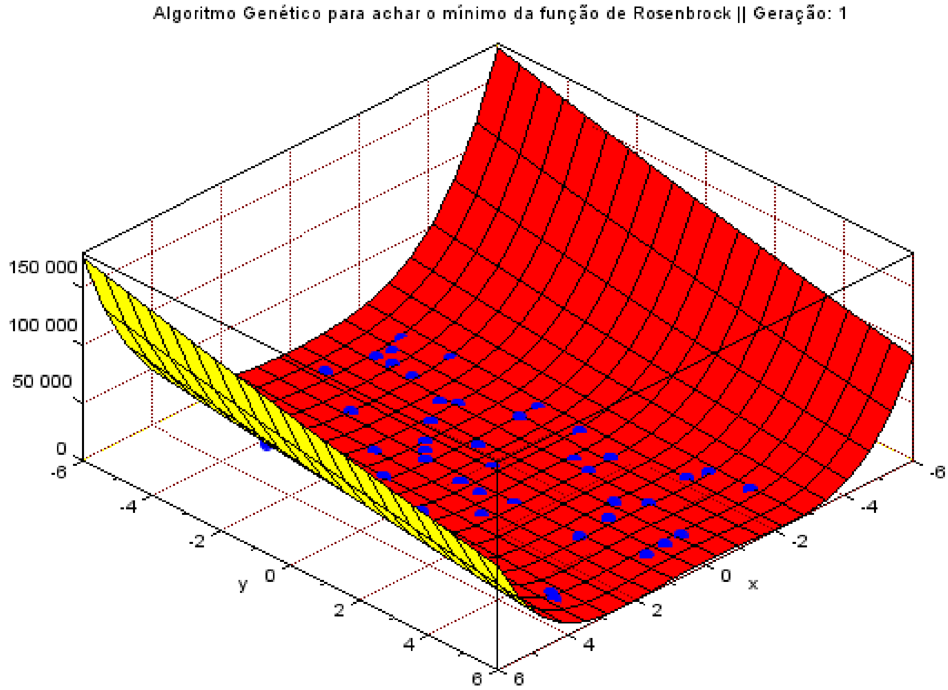

Figura 13 – Geração 1

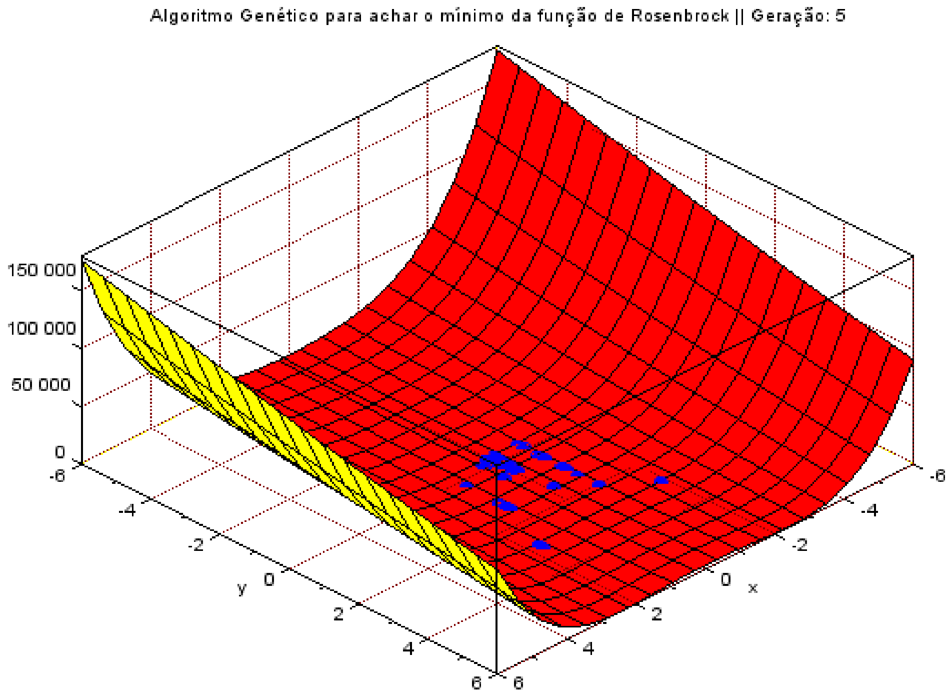

Figura 14 – Geração 5

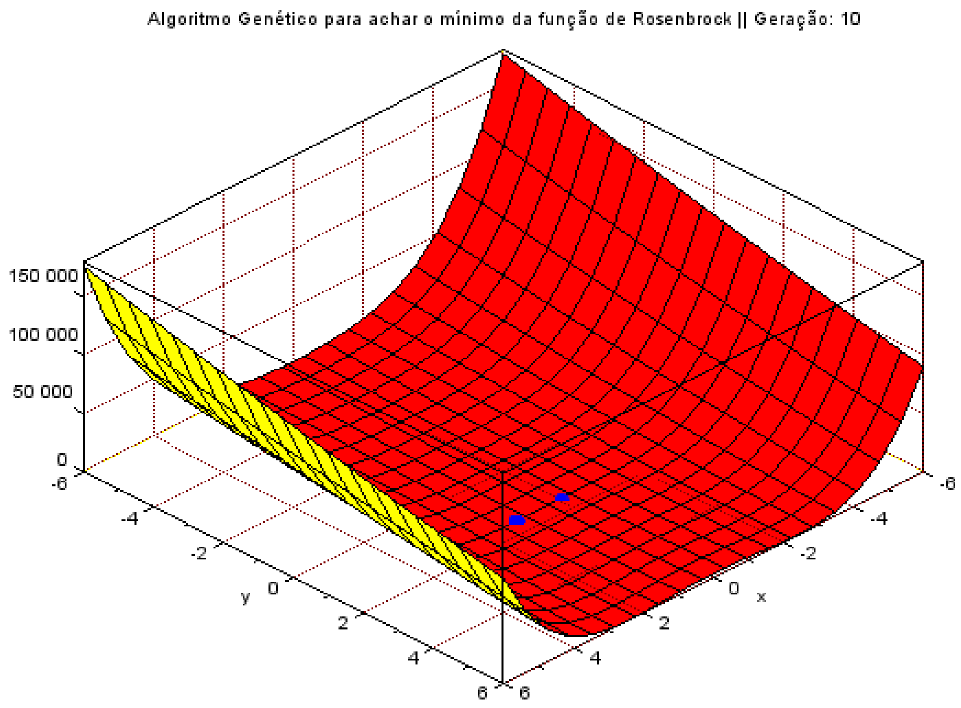

Figura 15 – Geração 10

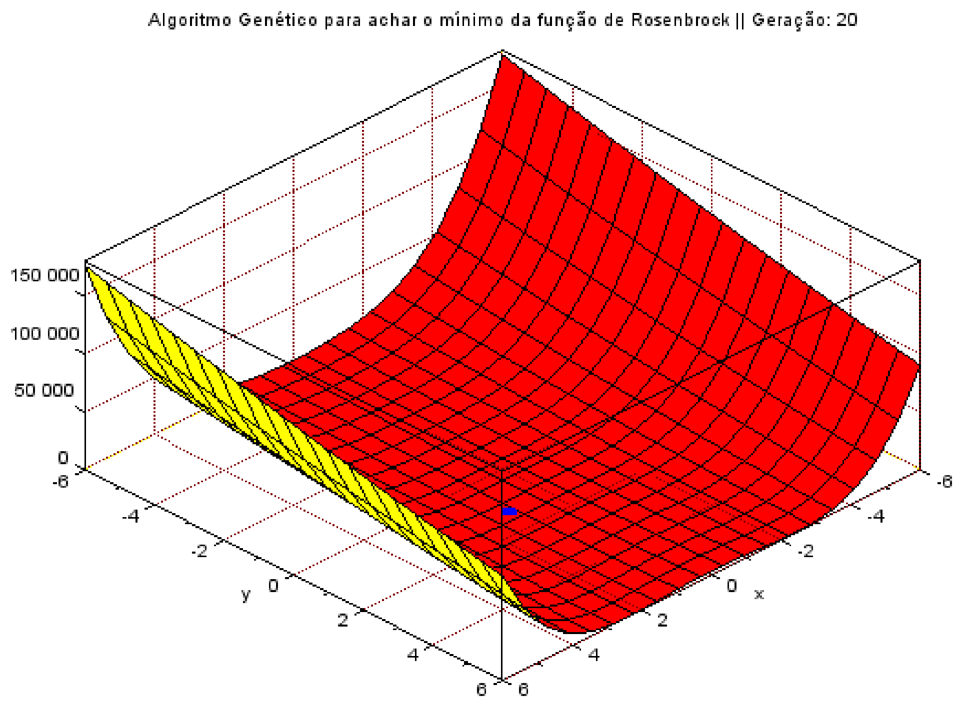

Figura 16 – Geração 20

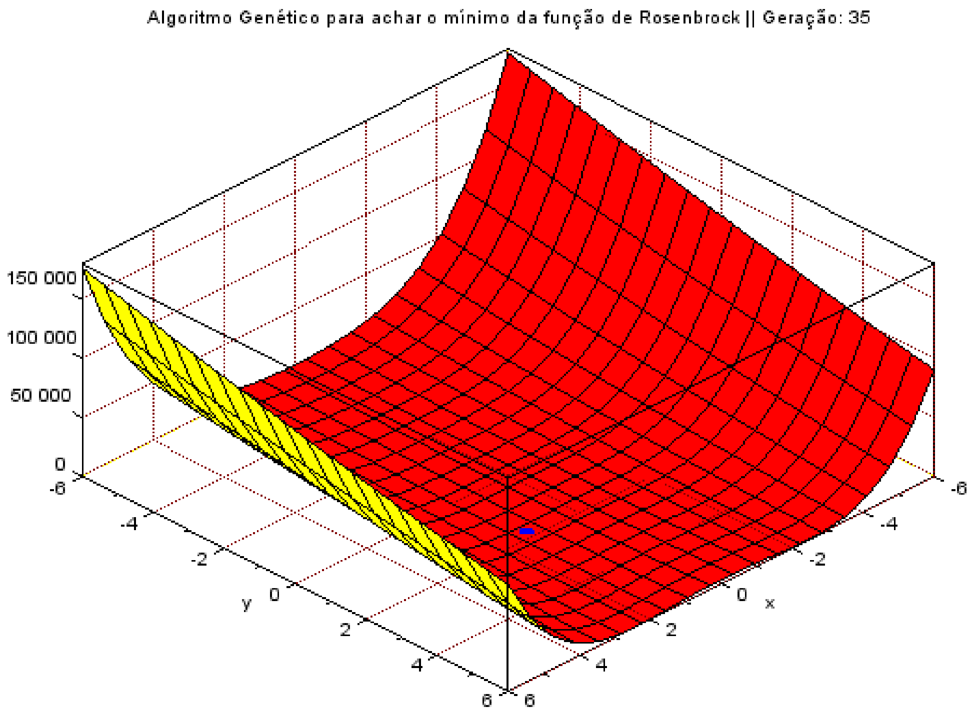

Figura 17 – Geração 35

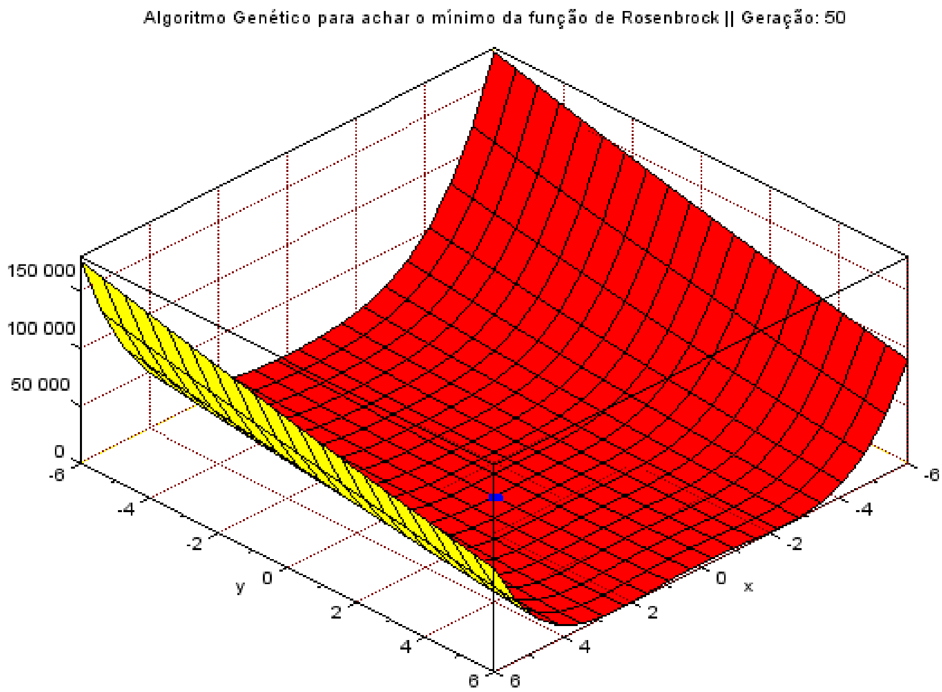

Figura 18 – Geração 50