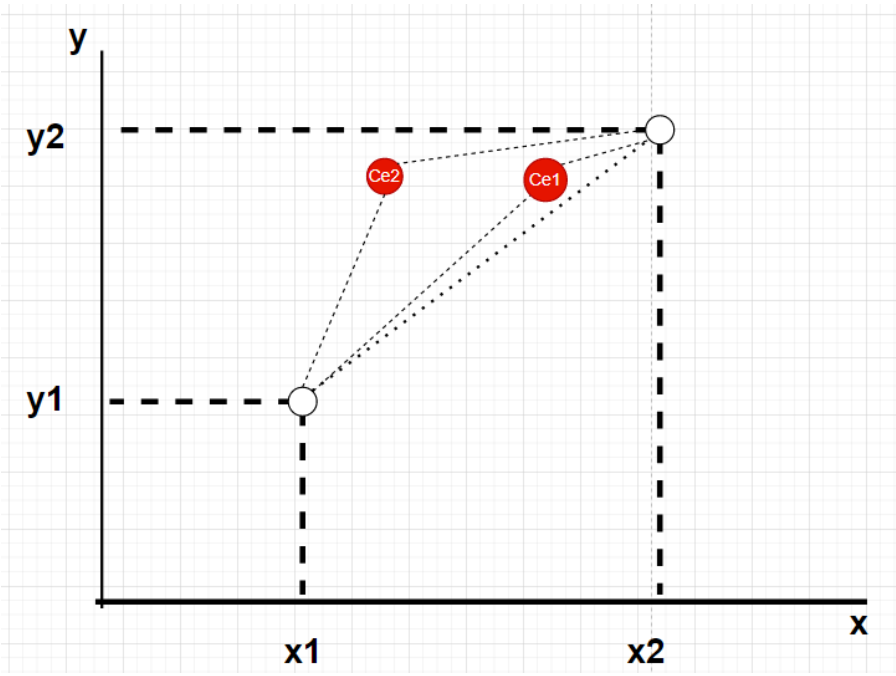
Introdução:

O modelo *Separatium* ajustas pontos “Centroides” para agrupar conjuntos de pontos, semelhante ao que faz o método k-means.

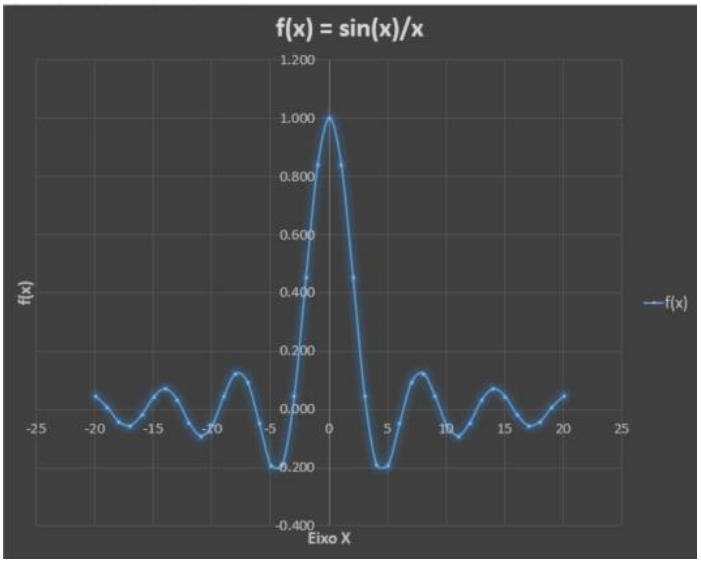
Por exemplo, dado dois pontos, relacionados probabilisticamente a dois conjuntos de dados, temos:

	x1 ; y1	x2 ; y2
Probabilidade 1	0.30	0.20
Probabilidade 2	0.70	0.80

O método cria centroides que devem se aproximar de forma iterativa de cada conjunto distinto de pontos:

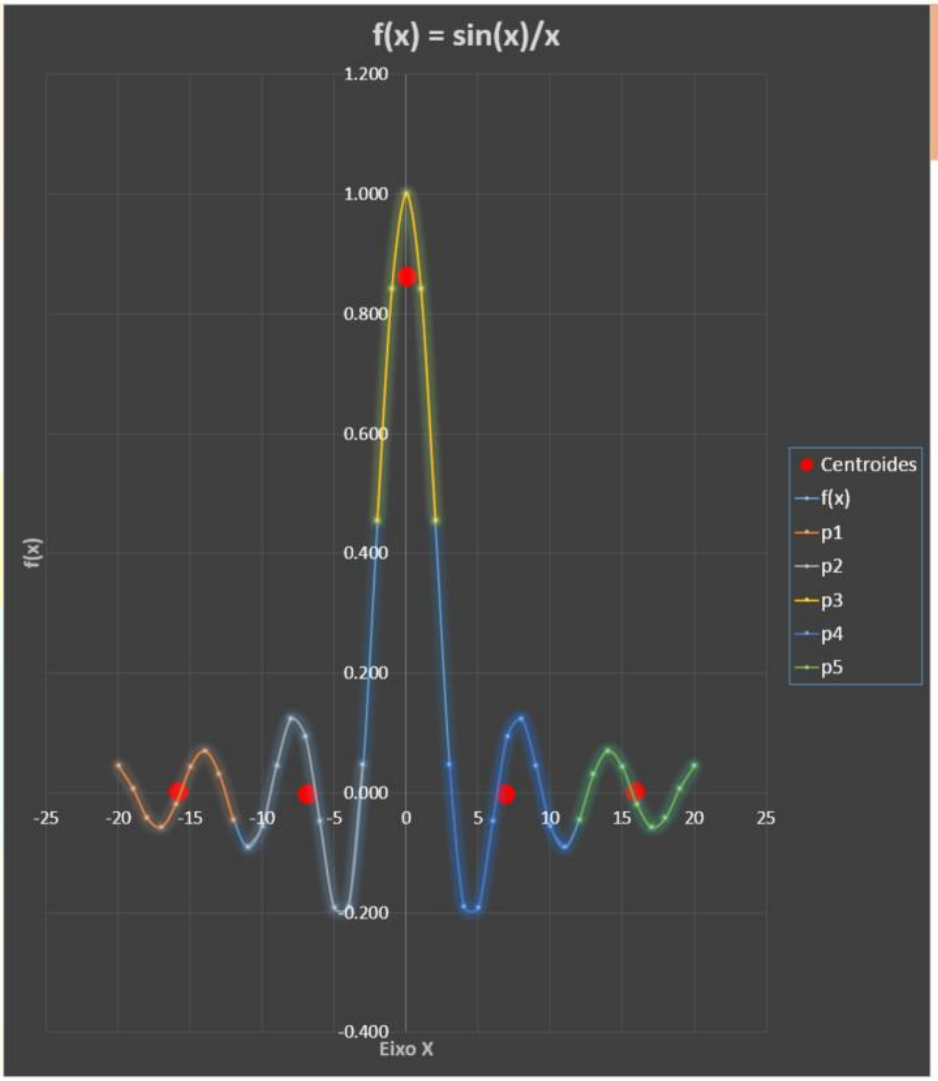


Primum test: Foi utilizado a Função Sin(X)/X:



Foram utilizados os pontos 41 pontos apresentados abaixo (com x normalizado de 0 a 1). Os valores de probabilidade foram chutados aleatoriamente. O modelo Separatium classificou cada conjunto de pontos nas respectivas “probabilidades” coloridas com a sua cor correspondente:

x	x normalizado	f(x)
-20	0.000	0.046
-19	0.025	0.008
-18	0.050	-0.042
-17	0.075	-0.057
-16	0.100	-0.018
-15	0.125	0.043
-14	0.150	0.071
-13	0.175	0.032
-12	0.200	-0.045
-11	0.225	-0.091
-10	0.250	-0.054
-9	0.275	0.046
-8	0.300	0.124
-7	0.325	0.094
-6	0.350	-0.047
-5	0.375	-0.192
-4	0.400	-0.189
-3	0.425	0.047
-2	0.450	0.455
-1	0.475	0.841
1E-07	0.500	1.000
1	0.525	0.841
2	0.550	0.455
3	0.575	0.047
4	0.600	-0.189
5	0.625	-0.192
6	0.650	-0.047
7	0.675	0.094
8	0.700	0.124
9	0.725	0.046
10	0.750	-0.054
11	0.775	-0.091
12	0.800	-0.045
13	0.825	0.032
14	0.850	0.071
15	0.875	0.043
16	0.900	-0.018
17	0.925	-0.057
18	0.950	-0.042
19	0.975	0.008
20	1.000	0.046



x	X desnorm	y
0.1	-15.8495	0.003
0.33	-6.85432	-0.002
0.5	-4E-05	0.863
0.67	6.84384	0.000
0.9	15.84156	0.003

^
|
|

CENTROIDES

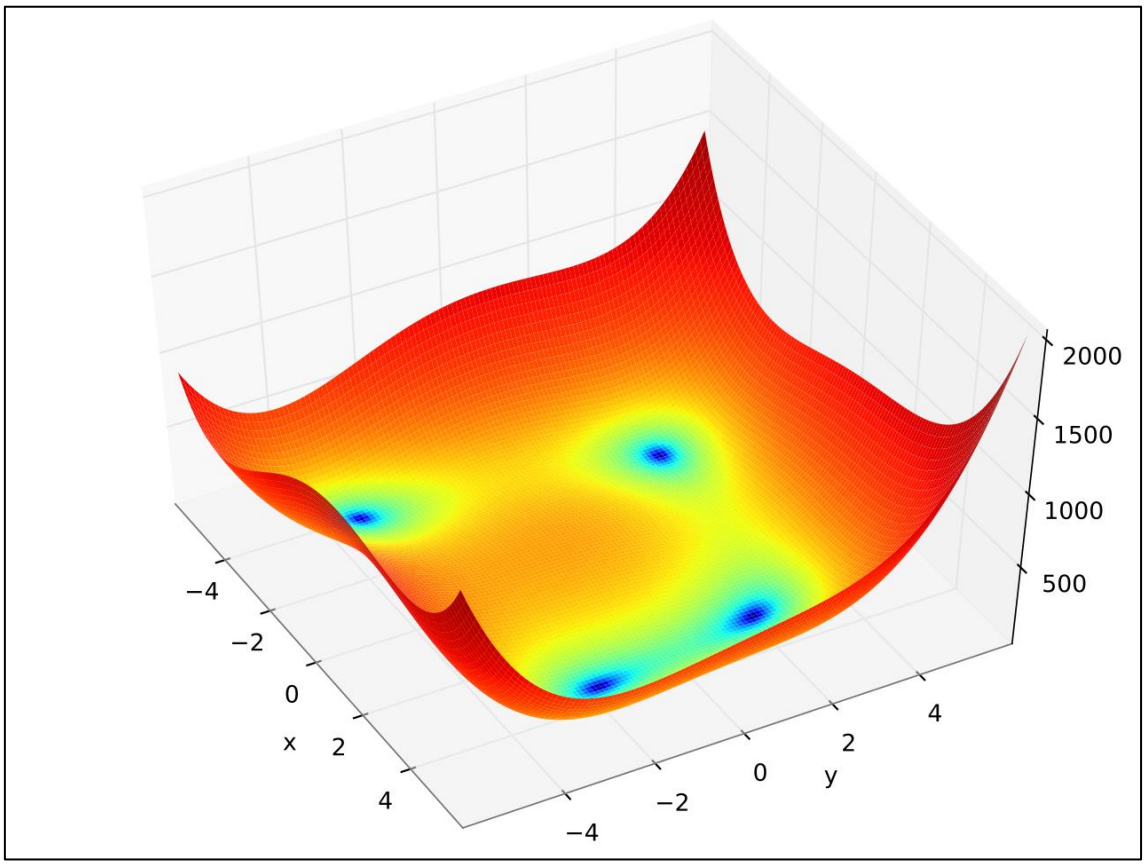
ITERAÇÃO 0 – Valores de probabilidade e dos Centróides

```
------------------------------------------------
        ITERACAO [0]
------------------------------------------------
 p [0]  0.289240|0.882790|0.046502|0.431303|0.792114|0.650750|0.645595|0.333929|0.417756|0.014
368|0.761740|0.875579|0.254888|0.624402|0.105321|0.850588|0.751141|0.624221|0.585787|0.398580|
0.054187|0.272297|0.172318|0.306553|0.054447|0.184788|0.026258|0.757963|0.640961|0.891673|0.65
5110|0.930201|0.774462|0.701612|0.361504|0.566576|0.352362|0.007099|0.900505|0.770118|0.021467
|
 p [1]  0.662245|0.645697|0.276355|0.286647|0.751018|0.126944|0.037787|0.375239|0.712730|0.436
367|0.429426|0.985027|0.608686|0.735979|0.039475|0.793474|0.762237|0.797438|0.434435|0.653910|
0.452548|0.364637|0.428372|0.154160|0.726140|0.994948|0.506522|0.733239|0.895453|0.276640|0.75
4706|0.557698|0.922337|0.031061|0.844344|0.673355|0.158005|0.882132|0.048594|0.870735|0.318499
|
 p [2]  0.478020|0.855763|0.927185|0.214000|0.895237|0.720659|0.976237|0.692675|0.155094|0.630
147|0.145223|0.519731|0.058519|0.299383|0.245871|0.053467|0.805905|0.979111|0.948920|0.082545|
0.733817|0.506618|0.004882|0.764878|0.350962|0.678237|0.922883|0.233094|0.726832|0.793618|0.55
1593|0.204852|0.649381|0.478778|0.418852|0.544618|0.199437|0.395089|0.237293|0.354531|0.025236
|
 p [3]  0.382515|0.874262|0.083755|0.681898|0.120133|0.137221|0.487803|0.099243|0.086141|0.570
347|0.833060|0.592759|0.575229|0.597938|0.943721|0.253467|0.520821|0.176815|0.980298|0.314439|
0.728408|0.185150|0.963820|0.207185|0.604002|0.508438|0.406622|0.999090|0.745730|0.761153|0.02
4326|0.128246|0.635414|0.108080|0.810144|0.755547|0.245302|0.297947|0.854791|0.331443|0.868294
|
 p [4]  0.687851|0.924202|0.443524|0.285789|0.867923|0.696990|0.806610|0.044737|0.677288|0.121
049|0.773145|0.862438|0.084869|0.980330|0.466440|0.593306|0.386952|0.465530|0.339037|0.148105|
0.489856|0.467283|0.783520|0.597937|0.277426|0.539067|0.843239|0.575373|0.393857|0.174682|0.44
3667|0.081708|0.098883|0.887191|0.367497|0.966806|0.584181|0.174107|0.011543|0.261469|0.295156
|
------------------------------------------------
------------------------------------------------
 CALCULANDO Ce
Ce[0][0] = 0.514448
Ce[1][0] = 0.018069
Ce[0][1] = 0.535971
Ce[1][1] = 0.035264
Ce[0][2] = 0.418587
Ce[1][2] = 0.079112
Ce[0][3] = 0.537571
Ce[1][3] = 0.110811
Ce[0][4] = 0.424701
Ce[1][4] = 0.064965
------------------------------------------------
```

ITERAÇÃO 19 – Valores de probabilidade e dos Centróides

```
------------------------------------------------
        ITERACAO [19]
------------------------------------------------
 p [0]  0.023767|0.013585|0.011693|0.011371|0.001394|0.006425|0.019892|0.018674|0.030115|0.046
385|0.034225|0.026689|0.072822|0.056231|0.021991|0.170149|0.196864|0.136287|0.196647|0.001490|
0.017002|0.001525|0.226882|0.735932|0.531605|0.548691|0.927277|0.790146|0.673726|0.823566|0.68
7763|0.508998|0.362157|0.185873|0.148383|0.044538|0.008457|0.060018|0.057019|0.061760|0.097984
|
 p [1]  0.013363|0.007475|0.006312|0.006016|0.000717|0.003234|0.009803|0.008862|0.013852|0.020
935|0.014661|0.010950|0.030143|0.021633|0.007699|0.068408|0.075926|0.038225|0.124871|0.001264|
0.015210|0.001323|0.155329|0.080309|0.173743|0.191483|0.038653|0.123091|0.206620|0.133810|0.25
5591|0.413309|0.586501|0.780946|0.814607|0.943394|0.988999|0.918704|0.920526|0.911088|0.852939
|
 p [2]  0.853635|0.911550|0.920366|0.918103|0.988739|0.943624|0.815150|0.780169|0.581928|0.407
759|0.250028|0.136107|0.209834|0.126398|0.036641|0.189548|0.172212|0.081358|0.155404|0.001324|
0.015214|0.001264|0.124827|0.037636|0.076332|0.068859|0.008092|0.021022|0.029635|0.010731|0.01
4905|0.021093|0.013853|0.008828|0.009806|0.003246|0.000703|0.005993|0.006316|0.007522|0.013430
|
 p [3]  0.011716|0.005926|0.004433|0.003968|0.000486|0.002370|0.007334|0.005793|0.007369|0.009
745|0.006822|0.005906|0.019013|0.011623|0.002720|0.018746|0.019222|0.012733|0.297796|0.994404|
0.935630|0.994403|0.297697|0.012550|0.019351|0.018894|0.002862|0.011304|0.018706|0.005793|0.00
6942|0.009828|0.007376|0.005775|0.007341|0.002381|0.000476|0.003956|0.004439|0.005967|0.011782
|
 p [4]  0.097519|0.061465|0.057195|0.060543|0.008664|0.044347|0.147821|0.186502|0.366736|0.515
176|0.694264|0.820349|0.668187|0.784116|0.930949|0.553150|0.535776|0.731397|0.225282|0.001519|
0.016944|0.001484|0.195264|0.133573|0.198969|0.172073|0.023117|0.054437|0.071313|0.026100|0.03
4798|0.046771|0.030113|0.018578|0.019863|0.006441|0.001365|0.011329|0.011699|0.013663|0.023865
|
------------------------------------------------
------------------------------------------------
 CALCULANDO Ce
Ce[0][0] = 0.671096
Ce[1][0] = -0.000435
Ce[0][1] = 0.896039
Ce[1][1] = 0.002882
Ce[0][2] = 0.103763
Ce[1][2] = 0.003095
Ce[0][3] = 0.499999
Ce[1][3] = 0.863205
Ce[0][4] = 0.328642
Ce[1][4] = -0.001797
------------------------------------------------
```

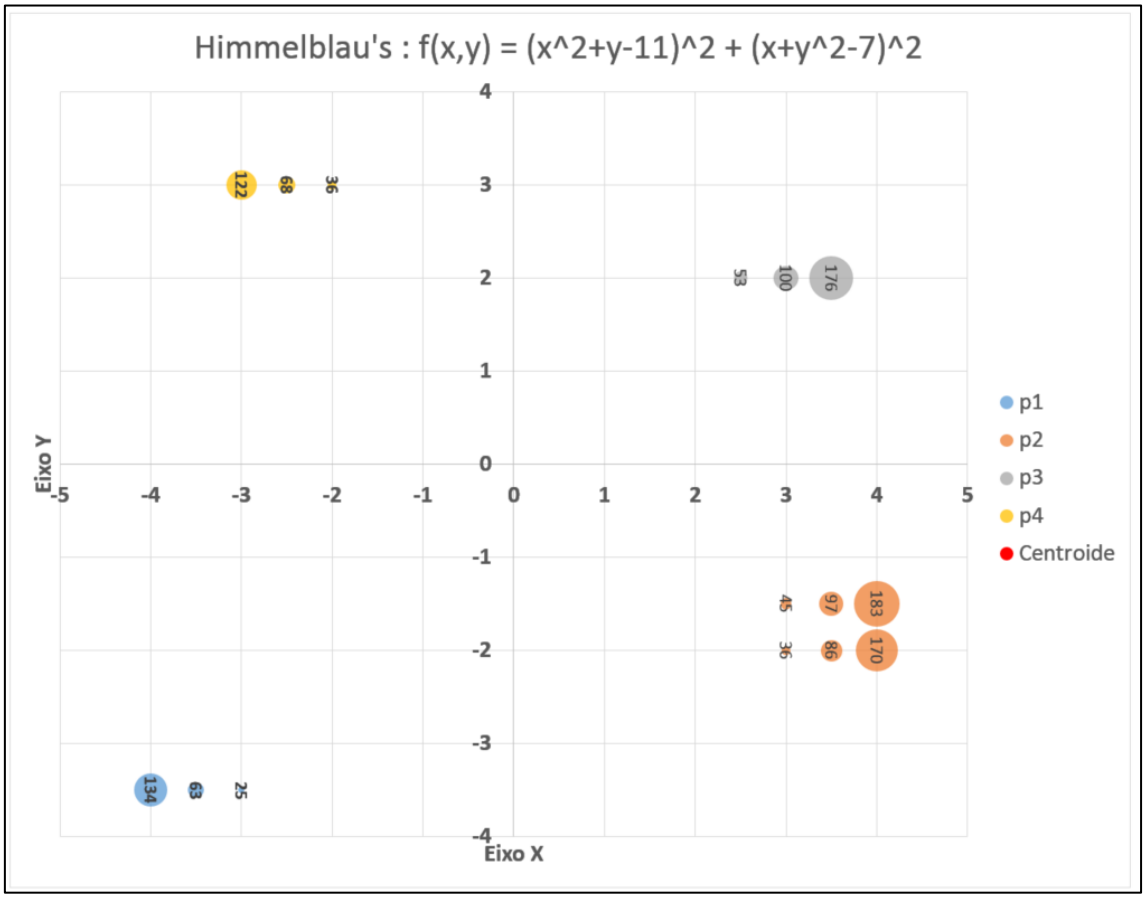
Secundo test: Foi utilizado a Função de Himmelblau, que tem duas variáveis, portanto 3D, e apresenta 4 pontos de mínimos conhecidos:



Foram submetidos ao modelo Separatium os seguintes 15 pontos, agrupados em 4 conjuntos de pontos que geometricamente estão na região dos pontos de mínimos, sendo que seus valores de x, y e f(x) foram normalizados de 0 a 1:

Nº	Eixo X	Eixo Y	f(x) = Himmelblau's	norm_fo
0	-4	-3.5	133.81	0.689
1	-3.5	-3.5	63.13	0.240
2	-3	-3.5	25.31	0.000
3	3	-2	36.00	0.068
4	3.5	-2	85.81	0.384
5	4	-2	170.00	0.919
6	3	-1.5	45.31	0.127
7	3.5	-1.5	96.63	0.453
8	4	-1.5	182.81	1.000
9	2.5	2	52.81	0.175
10	3	2	100.00	0.474
11	3.5	2	175.81	0.956
12	-3	3	122.00	0.614
13	-2.5	3	68.31	0.273
14	-2	3	36.00	0.068

O modelo Separatium fez a classificação dos 15 pontos em 4 probabilidades como segue:



Eixo X	Eixo Y	f(x) = Himmelblau's	norm_fo
-2.49971	2.99915	75.15	0.316413
-3.49913	-3.49913	74.00	0.309144
2.999702	1.994228	108.94	0.530958
3.499336	-1.75062	102.31	0.488862

^

|

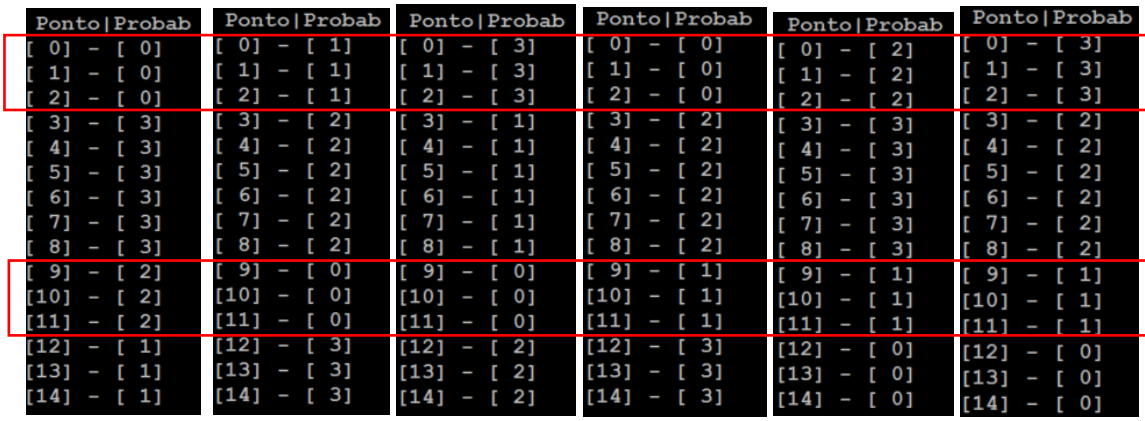
|

CENTROIDE

Finalis Considerations:

- Todos os testes foram feitos utilizando **valores aleatórios da matriz de probabilidades**.

- Foram realizadas algumas execuções e, apesar de ora associar o conjunto de ponto a uma probabilidade ou a outra, **o resultado da classificação se manteve sempre igual!!** Como exemplo, segue o resultado das probabilidades associada aos pontos em 6 rodadas do Segundo Teste:



Ponto | Probab	Ponto | Probab	Ponto | Probab	Ponto | Probab	Ponto | Probab	Ponto | Probab
[0] - [0]	[0] - [1]	[0] - [3]	[0] - [0]	[0] - [2]	[0] - [3]
[1] - [0]	[1] - [1]	[1] - [3]	[1] - [0]	[1] - [2]	[1] - [3]
[2] - [0]	[2] - [1]	[2] - [3]	[2] - [0]	[2] - [2]	[2] - [3]
[3] - [3]	[3] - [2]	[3] - [1]	[3] - [2]	[3] - [3]	[3] - [2]
[4] - [3]	[4] - [2]	[4] - [1]	[4] - [2]	[4] - [3]	[4] - [2]
[5] - [3]	[5] - [2]	[5] - [1]	[5] - [2]	[5] - [3]	[5] - [2]
[6] - [3]	[6] - [2]	[6] - [1]	[6] - [2]	[6] - [3]	[6] - [2]
[7] - [3]	[7] - [2]	[7] - [1]	[7] - [2]	[7] - [3]	[7] - [2]
[8] - [3]	[8] - [2]	[8] - [1]	[8] - [2]	[8] - [3]	[8] - [2]
[9] - [2]	[9] - [0]	[9] - [0]	[9] - [1]	[9] - [1]	[9] - [1]
[10] - [2]	[10] - [0]	[10] - [0]	[10] - [1]	[10] - [1]	[10] - [1]
[11] - [2]	[11] - [0]	[11] - [0]	[11] - [1]	[11] - [1]	[11] - [1]
[12] - [1]	[12] - [3]	[12] - [2]	[12] - [3]	[12] - [0]	[12] - [0]
[13] - [1]	[13] - [3]	[13] - [2]	[13] - [3]	[13] - [0]	[13] - [0]
[14] - [1]	[14] - [3]	[14] - [2]	[14] - [3]	[14] - [0]	[14] - [0]

- Os **testes foram realizados com 20 iterações**, mas a convergência acontece já logo nas primeiras dezenas de iterações.

- Um teste com **10.000 iterações** foi realizado e consumiu apenas **0.8 segundos** em uma máquina (Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz)

Computatrum linguarum (Linguagem de Computador):

O código utilizado nos testes foi escrito de forma generalizada utilizando a linguagem C/C++. Da forma que está escrito pode ser facilmente reescrito para outra linguagem de programação:

```cpp
#include <iostream>
#include <math.h>
#include <vector>
#include <iomanip>
#include <stdio.h>

using namespace std;

void separatium(   vector<vector<double>> &p,
                   vector<vector<double>> &H,
                   vector<vector<double>> &Ce,
                   vector<vector<double>> &D,
                   vector<vector<double>> &X,
                   int iteracoes)
{
  int i = 0;
  while(i < iteracoes)
  {

    std::cout << " ------------------------------------------------\n";
    std::cout << "          ITERACAO ["<< i << "] \n";
    std::cout << " ------------------------------------------------\n";
    for (int i = 0; i < X.size(); i++) {
      std::cout << " X[" << i << "]\t";
      for (int j = 0; j < X[i].size(); j++)
        printf("%f|", X[i][j]);
      std::cout << endl;
    }

    std::cout << " ------------------------------------------------\n";
    for (int i = 0; i < p.size(); i++) {
      std::cout << " p [" << i << "]\t";
      for (int j = 0; j < p[i].size(); j++)
        printf("%f|", p[i][j]);
      std::cout << endl;
    }
  std::cout << " ------------------------------------------------\n";
  for (int i = 0; i < H.size(); i++) {
    std::cout << " H [" << i << "]\t";
    for (int j = 0; j < H[i].size(); j++)
      printf("%f|", H[i][j]);
    std::cout << endl;
  }

    std::cout << " ------------------------------------------------\n";
    std::cout << " CALCULANDO Ce \n";
    int Probabilidades = p.size();
    int Dimensoes = X.size();
    int Pontos = 0;
    if(Dimensoes > 0)
      Pontos = X[0].size();

```

```
52
53      for (int i = 0; i < Probabilidades; i++)
54      {
55        for (int j = 0; j < Dimensoes; j++)
56        {
57          double upper = 0.0;
58          double lower = 0.0;
59          for (int k = 0; k < Pontos; k++)
60          {
61            upper += pow(p[i][k], 2.0) * X[j][k];
62            lower += pow(p[i][k], 2.0);
63          }
64          Ce[j][i] = upper / lower;
65          printf("Ce[%d][%d] = %f\n",j, i, Ce[j][i]);
66        }
67      }
68
69      std::cout << " ------------------------------------------------\n";
70      std::cout << " CALCULANDO D11 D12 D21 D22 \n";
71
72    for (int i = 0; i < Probabilidades; i++)
73    {
74      for (int k = 0; k < Pontos; k++)
75      {
76        double soma = 0.0;
77        for (int j = 0; j < Dimensoes; j++)
78        {
79          soma += pow(X[j][k] - Ce[j][i], 2.0);
80        }
81        D[i][k] = sqrt(soma);
82        printf("D[%d][%d] = %f\n",i, k, D[i][k]);
83      }
84    }
85
86    std::cout << " ------------------------------------------------\n";
87    std::cout << " CALCULANDO H11 H12 H21 H22 \n";
88
89    for (int i = 0; i < Probabilidades; i++)
90    {
91      for (int k = 0; k < Pontos; k++)
92      {
93        double soma = 0.0;
94        for (int i2 = 0; i2 < Probabilidades; i2++)
95        {
96          soma += pow(D[i][k] / D[i2][k], 2.0);
97        }
98        H[i][k] = 1.0 / soma;
99        printf("H[%d][%d] = %f\n",i, k, H[i][k]);
100     }
101   }
102
103   std::cout << " ------------------------------------------------\n";
104   std::cout << " ATUALIZANDO p11 p12 p21 p22 \n";
105
106   for (int i = 0; i < Probabilidades; i++)
107   {
108     for (int k = 0; k < Pontos; k++)
109     {
110       p[i][k] = H[i][k];
111     }
112   }
113
114     std::cout << " ------------------------------------------------\n";
115     i++;
116   }
117 }
```

```
119  int main()
120  {
121      int qtd_dimensoes = -1;
122      int qtd_probab = 4;
123      int qtd_pontos = -1;
124      int qtd_iteracoes = 20;
125
126      vector<vector<double>> p;
127      vector<vector<double>> Ce;
128      vector<vector<double>> D;
129      vector<vector<double>> X;
130
131      // ------------------------
132      // DEFINIÇÃO DOS PONTOS ...
133      // ------------------------
134
135      // ------------------------
136      //  f(x) = Himmelblau's ...
137      // ------------------------
138      X.resize(3);
139
140      X[0].push_back(-4);
141      X[0].push_back(-3.5);
142      X[0].push_back(-3);
143      X[0].push_back(3);
144      X[0].push_back(3.5);
145      X[0].push_back(4);
146      X[0].push_back(3);
147      X[0].push_back(3.5);
148      X[0].push_back(4);
149      X[0].push_back(2.5);
150      X[0].push_back(3);
151      X[0].push_back(3.5);
152      X[0].push_back(-3);
153      X[0].push_back(-2.5);
154      X[0].push_back(-2);
155
156
157      X[1].push_back(-3.5);
158      X[1].push_back(-3.5);
159      X[1].push_back(-3.5);
160      X[1].push_back(-2);
161      X[1].push_back(-2);
162      X[1].push_back(-2);
163      X[1].push_back(-1.5);
164      X[1].push_back(-1.5);
165      X[1].push_back(-1.5);
166      X[1].push_back(2);
167      X[1].push_back(2);
168      X[1].push_back(2);
169      X[1].push_back(3);
170      X[1].push_back(3);
171      X[1].push_back(3);
172
173      X[2].push_back(0.69);
174      X[2].push_back(0.24);
175      X[2].push_back(0.00);
176      X[2].push_back(0.07);
177      X[2].push_back(0.38);
178      X[2].push_back(0.92);
179      X[2].push_back(0.13);
180      X[2].push_back(0.45);
181      X[2].push_back(1.00);
182      X[2].push_back(0.17);
183      X[2].push_back(0.47);
184      X[2].push_back(0.96);
185      X[2].push_back(0.61);
186      X[2].push_back(0.27);
187      X[2].push_back(0.07);
188
```

```
282     qtd_dimensoes = X.size();
283
284     if ( X.size() > 0)
285         qtd_pontos = X[0].size();
286
287     // ------------------------------------------------
288     // DEFINIÇÃO DA PROBABILIDADE INICIAL, ALEATORIA ...
289     // ------------------------------------------------
290     p.resize(qtd_probab);
291     srand (time(NULL));
292     for (int i=0; i < qtd_probab; i++)
293     {
294         p[i].reserve(qtd_pontos);
295         for (int j=0; j < qtd_pontos; j++)
296         {
297             p[i].push_back(((double) rand() / (RAND_MAX)));
298         }
299     }
300
301     vector<vector<double>> H (p);
302
303     for (int i=0; i < qtd_probab; i++)
304         D.push_back(p[i]);
305
306
307     X.resize(qtd_dimensoes);
308
309     for(int i=0; i< qtd_dimensoes; i++)
310     {
311       vector<double> Temp (qtd_probab, 0.0);
312       Ce.push_back(Temp);
313     }
314
315     separatium(p, H, Ce, D, X, qtd_iteracoes);
316
317
318     std::cout << " ------------------------------------------------\n";
319
320     int Probabilidades = p.size();
321     int Pontos = 0;
322     if (Probabilidades > 0)
323         Pontos = p[0].size();
324     vector<int> PontoEscolhido(Pontos,0);
325     for (int j = 0; j < Pontos; j++)
326     {
327         double maior = 0.0;
328         for (int ip = 0; ip < Probabilidades; ip++)
329         {
330             if (p[ip][j] >= maior)
331             {
332                 maior = p[ip][j];
333                 PontoEscolhido[j] = ip;
334             }
335         }
336     }
337
338     std::cout << " Ponto|Probab\n";
339     for (int k = 0; k < Pontos; k++)
340         printf("[%2d] - [%2d]\n", k, PontoEscolhido[k]);
341     std::cout << " ------------------------------------------------\n";
342
343 }
```