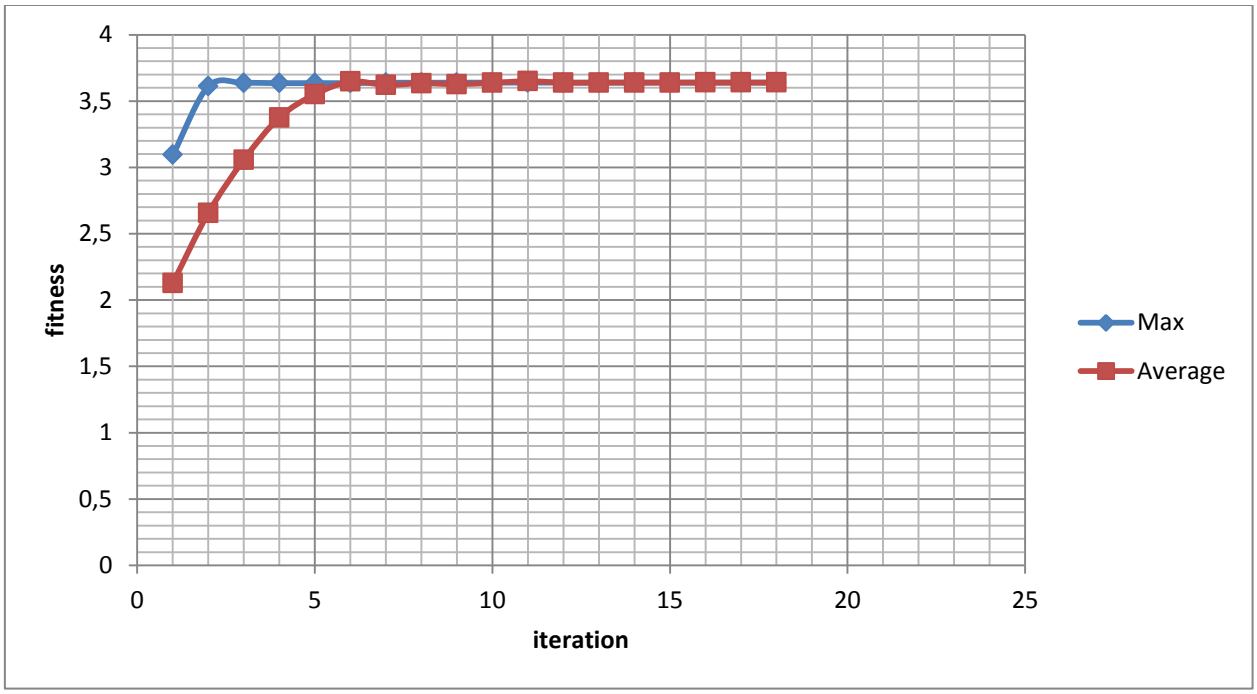
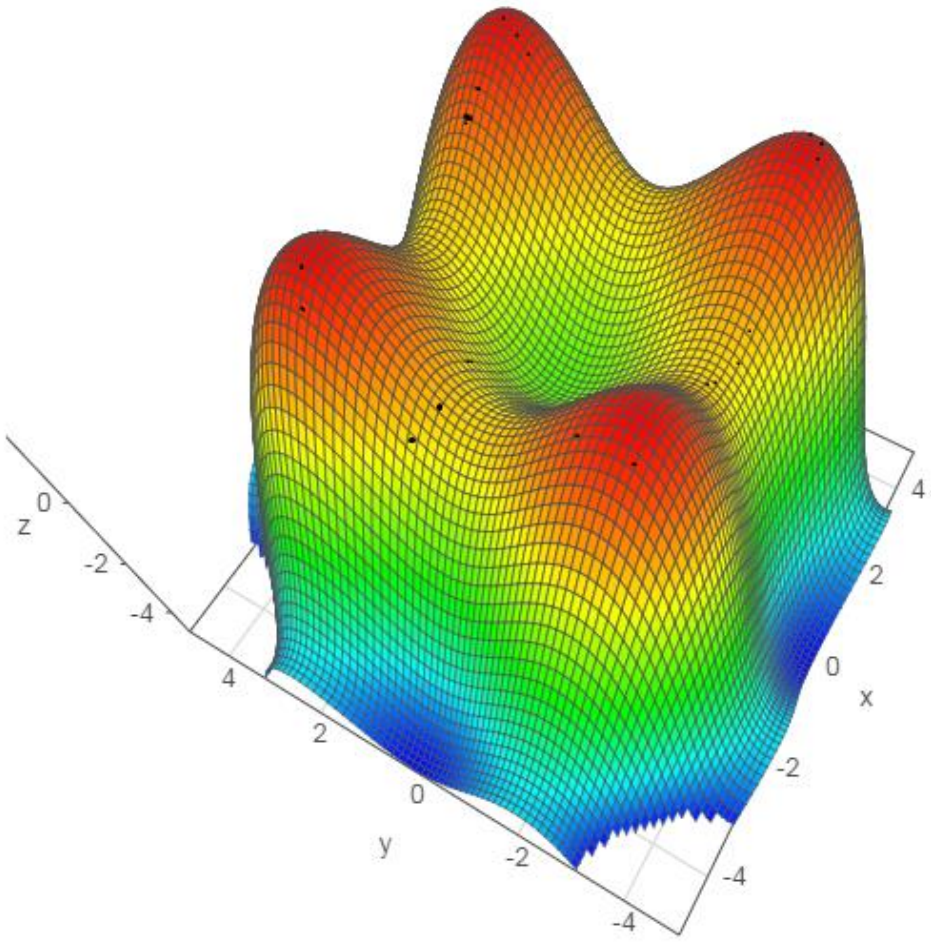
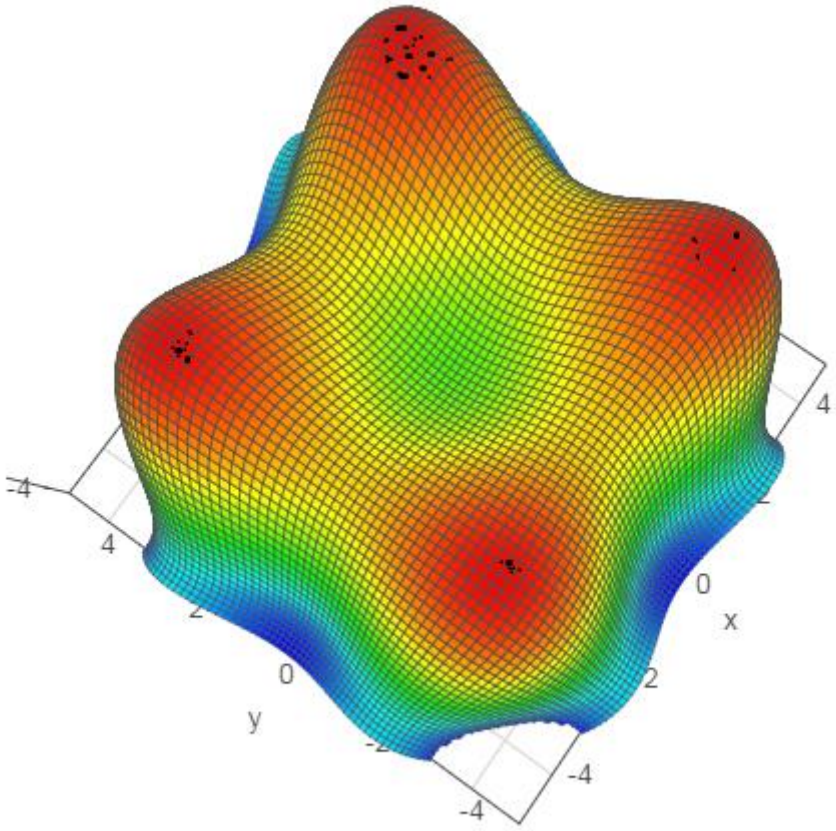


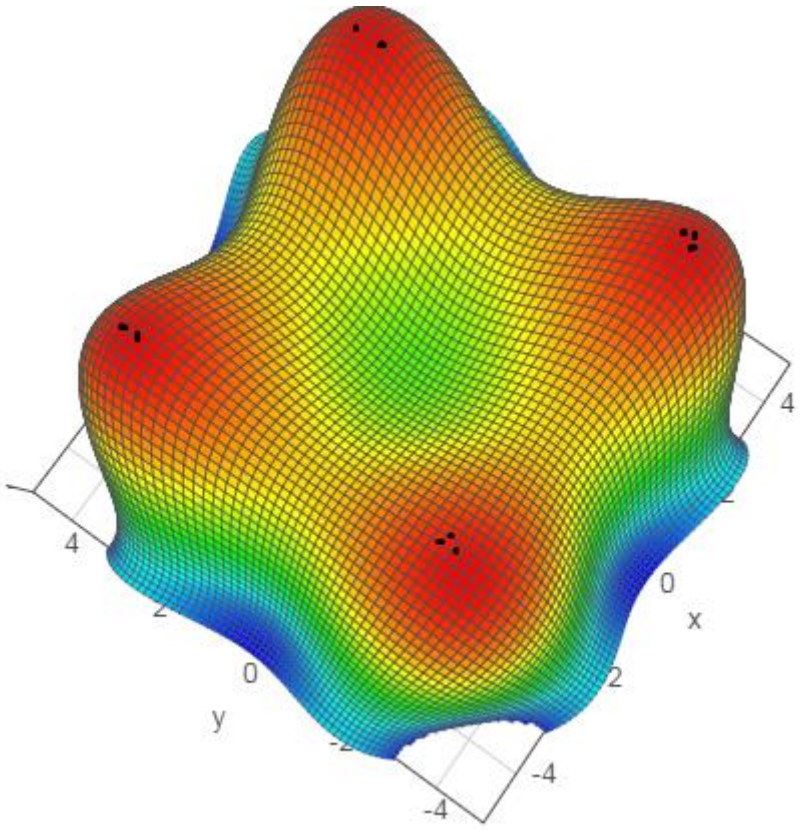
0:



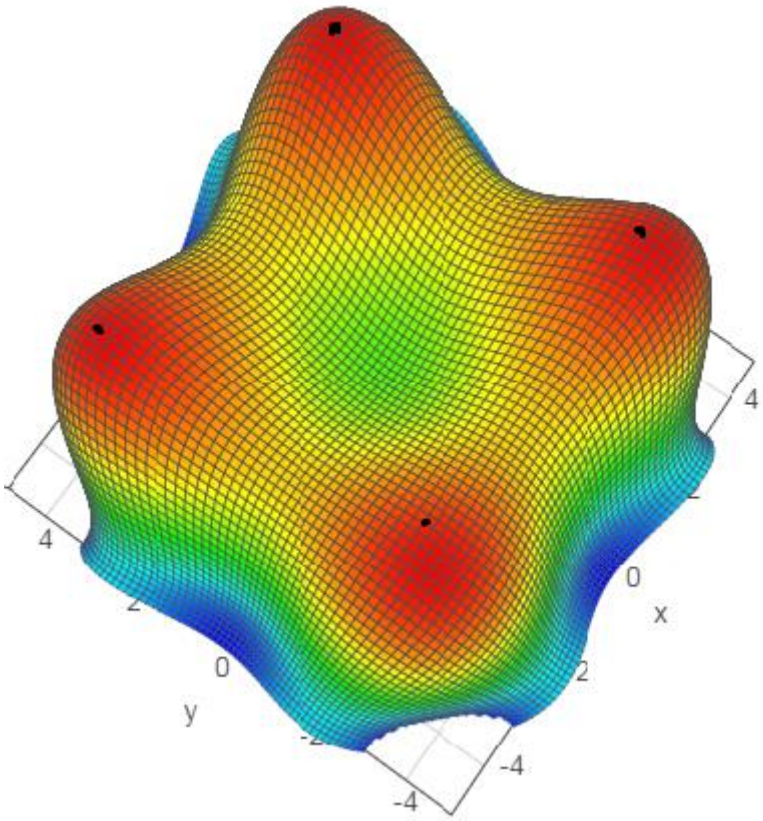
4:



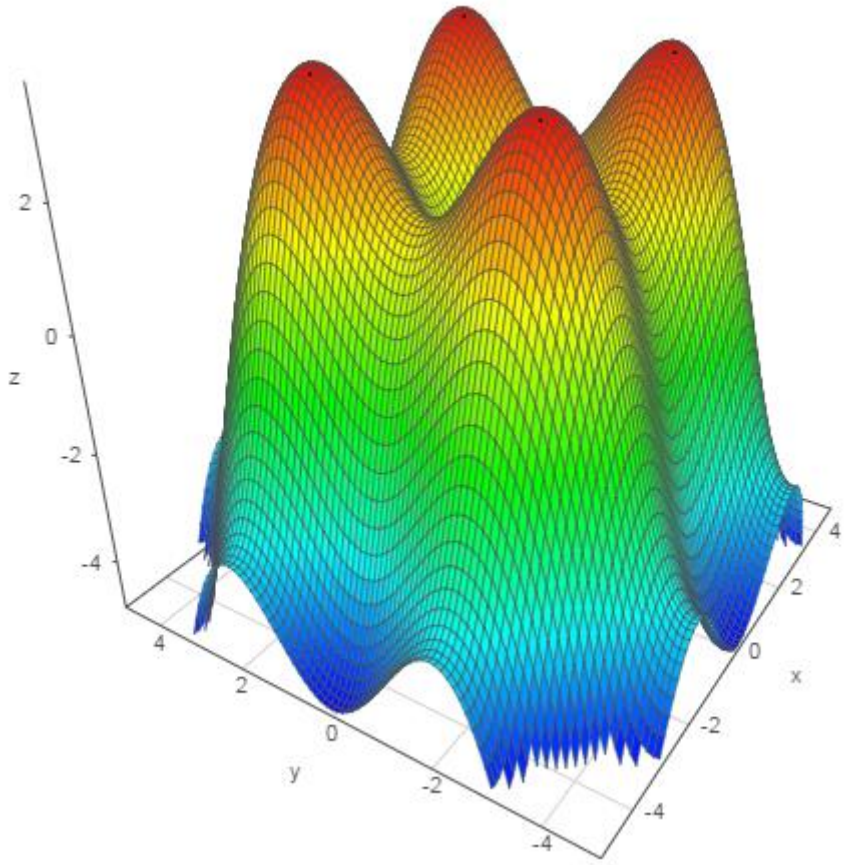
8:



12:



16:



0

x	y	z
2,158	-2,625	3,09386
-1,311	2,563	2,667856
-1,121	-2,375	2,655958
1,387	2,646	2,620707
1,46	2,705	2,594665
-0,969	-2,375	2,446098
-1,292	2,7	2,394991
1,985	0,747	2,324755
-1,97	0,742	2,31678
-1,89	0,635	2,170811
2,249	-0,5	1,991753
2,273	-0,5	1,975452
2,078	-0,315	1,913998
2,151	0,122	1,813897
1,414	3,154	1,684596
-2,3	-3,152	1,682929
-0,508	2,598	1,591241
0,867	-2,82	1,55235
-0,891	-2,834	1,549986
-2,117	-3,235	1,507775

4

x	y	z
1,985	1,997	3,635485
-2,053	1,934	3,626624
-1,975	1,938	3,624776
2,063	1,919	3,62193
1,99	1,919	3,621486
-1,907	1,953	3,612258
-1,907	1,953	3,612258
-1,907	2,109	3,610921
1,821	2,114	3,574091
-2,224	-1,899	3,563877
-2,263	-1,958	3,555075
-2,253	-1,895	3,544593
-2,327	-1,938	3,501978
2,253	-1,797	3,499625
-2,327	-1,895	3,48918
-2,371	2,061	3,470271
-2,38	2,041	3,461497
-2,38	2,061	3,460325
-2,38	2,061	3,460325
2,38	-2,09	3,45659

8

x	y	z
-2,024	2,017	3,639181
2,004	2,017	3,638414
2,004	-2,017	3,638414
-2,004	-1,997	3,637266
2,004	-1,997	3,637266
2,004	-1,997	3,637266
1,99	-2,017	3,637173
1,99	-2,017	3,637173
-1,985	-2,026	3,636825
1,985	2,017	3,636633
1,985	2,017	3,636633
1,985	2,017	3,636633
1,985	2,017	3,636633
1,985	2,017	3,636633
1,985	2,017	3,636633
1,985	2,017	3,636633
1,985	2,017	3,636633
-1,985	2,017	3,636633
1,985	2,017	3,636633
-1,985	2,017	3,636633

12

x	y	z
2,029	2,017	3,639212
-2,024	2,017	3,639181
-2,024	2,017	3,639181
-2,024	2,017	3,639181
2,024	-2,017	3,639181
2,024	-2,017	3,639181
-2,024	-2,017	3,639181
2,024	-2,017	3,639181
-2,024	-2,017	3,639181
2,024	-2,017	3,639181
-2,024	2,017	3,639181
-2,024	2,017	3,639181
2,024	2,017	3,639181
-2,024	2,017	3,639181
-2,024	2,017	3,639181
-2,024	2,017	3,639181
-2,024	2,017	3,639181
-2,024	2,017	3,639181
-2,024	2,017	3,639181
-2,024	2,017	3,639181

16

x	y	z
-2,024	2,026	3,639372
2,024	2,026	3,639372
2,024	2,026	3,639372
2,029	-2,019	3,639284
2,029	-2,019	3,639284
2,029	2,019	3,639284
2,029	2,019	3,639284
-2,024	2,019	3,639253
-2,024	2,019	3,639253
-2,024	-2,019	3,639253
-2,024	-2,019	3,639253
-2,024	-2,019	3,639253
2,024	-2,019	3,639253
2,024	-2,019	3,639253
-2,024	-2,019	3,639253
2,024	-2,019	3,639253
2,024	-2,019	3,639253
2,024	-2,019	3,639253
-2,024	-2,019	3,639253
2,029	2,017	3,639212

CODE:

```
#include <iostream>
#include <vector>
#include <time.h>
#include <math.h>
#include <algorithm>
using namespace std;

typedef vector<int> Chromosome;
typedef vector<Chromosome> Population;

Population population;
Population population2;
Population makePopulation();
double fitness(Chromosome chromosome);
double s_function(Chromosome chromosome);
bool sortPopulation(const Chromosome &chromosome1, const Chromosome &chromosome2);
Population newPopulation();
double distance(const Chromosome &chromosome1, int k);
double sharing_fitness(Chromosome chromosome);

int main()
{
	srand(time(NULL));
	ofstream file("OutData.txt", ios::out);
	population = makePopulation();
	population2 = population;
	sort(population.begin(), population.end(), sortPopulation);
```

```
        population2 = population;
        double aver = 0.0;
for (int noChanged = 0, k = 0; noChanged < 30 && k < 500; k++)
{
        population = newPopulation();
        population2 = population;
        sort(population.begin(), population.end(), sortPopulation);
        population2 = population;
        double max = sharing_fitness(population[0]);
        double average = 0.0;
for (int i = 0; i < 20; i++)
{
        average += sharing_fitness(population[i]);
}
        average = average / 20;
if (average == aver)
{
        noChanged++;
}
else
{
        noChanged = 0;
}
        aver = average;
        file << max << "\t" << average << endl;
}
file.close();
system("pause");
return 0;
```

```
}
	Population makePopulation()
{
Population result;
for (int i = 0; i < 20; i++)
{
	Chromosome chromosome;
		for (int j = 0; j < 22; j++)
{
			chromosome.push_back(rand() % 2);
}
	result.push_back(chromosome);
}
	return result;
}
double fitness(Chromosome chromosome)
{
	int sum = 0;
	for (int i = 1; i < 11; i++)
	{
		sum += chromosome[i] * pow(2, (i - 1));
	}
double x = sum / 1023.0 * 5.0;
if (chromosome[0] == 0)
	{
	x = x * (-1.0);
	}
sum = 0;
for (int i = 12; i < 22; i++)
```

```
        {
        sum += chromosome[i] * pow(2, (i - 12));
        }
double y = sum / 1023.0 * 5.0;
        if (chromosome[11] == 0)
{
        y = y * (-1.0);
}
double z = x * sin(abs(x)) + y * sin(abs(y));
return z;
}
double distance(const Chromosome &chromosome1, int k)
{
int sum = 0;
for (int i = 1; i < 11; i++)
        {
        sum += chromosome1[i] * pow(2, (i - 1));
        }
double x1 = sum / 1023.0 * 5.0;
if (chromosome1[0] == 0)
        {
        x1 = x1 * (-1.0);
        }
sum = 0;
for (int i = 12; i < 22; i++)
        {
        sum += chromosome1[i] * pow(2, (i - 12));
        }
double y1 = sum / 1023.0 * 5.0;
```

```
if (chromosome1[11] == 0)
        {
        y1 = y1 * (-1.0);
        }
sum = 0;
for (int i = 1; i < 11; i++)
        {
        sum += population2[k][i] * pow(2, (i - 1));
        }
double x2 = sum / 1023.0 * 5.0;
if (population2[k][0] == 0)
        {
        x2 = x2 * (-1.0);
        }
sum = 0;
for (int i = 12; i < 22; i++)
        {
        sum += population2[k][i] * pow(2, (i - 12));
        }
double y2 = sum / 1023.0 * 5.0;
if (population2[k][11] == 0)
        {
        y2 = y2 * (-1.0);
        }
double dist = sqrt((x1 - x2)*(x1 - x2) + (y1 - y2)*(y1 - y2));
return dist;
}
double s_function(Chromosome chromosome)
{
```

```
double sigma = 0.6;
double s = 0.0;
for (int i = 0; i < population.size(); i++)
{
        double d = distance(chromosome, i);
        if (d < sigma)
        {
        s += (1 - (d / sigma));
        }
}
return s;
}
double sharing_fitness(Chromosome chromosome)
        {
        double f = fitness(chromosome) / s_function(chromosome);
        return f;
        }
bool sortPopulation(const Chromosome &chromosome1, const Chromosome &chromosome2)
        {
        return sharing_fitness(chromosome1) > sharing_fitness(chromosome2);
        }
Population newPopulation()
        {
        int firstParent, secondParent;
        Population result;
        for (int i = 0; i < 10; i++)
                {
                firstParent = rand() % 10;
                secondParent = rand() % 10;
```

```
                while (firstParent != secondParent)
                {
                        secondParent = rand() % 10;
                }
        Chromosome child1, child2;
for (int j = 0; j < 22; j++)
        {
        int mask = rand() % 2;
        if (mask == 0)
        {
child1.push_back(population[firstParent][j]);
child2.push_back(population[secondParent][j]);
}
else
{
        child1.push_back(population[secondParent][j]);
        child2.push_back(population[firstParent][j]);
}
}
result.push_back(child1);
result.push_back(child2);
}
return result;
}
```