

```

#include <time.h>
#include <math.h>
#include <algorithm>
#include <iostream>
#include <math.h>
#include <fstream>
#include <vector>

using namespace std;

typedef vector<int> Chrm;
typedef vector<Chrm> Pplt;
Pplt population;

double fitness(Chrm Chrm)
{
    int sum = 0;
    for (int i = 1; i < 11; i++)
    {
        sum += Chrm[i] * pow(2, (i - 1));
    }
    int x = sum / 1023 * 5;
    if (Chrm[0] == 0)
    {
        x = x * (-1);
    }

    sum = 0;
    for (int i = 12; i < 22; i++)
    {
        sum += Chrm[i] * pow(2, (i - 12));
    }
    int y = sum / 1023 * 5;

```

```

    if (Chrm[11] == 0)
    {
        y = y * (-1);
    }

    double z = x * sin(abs(x)) + y * sin(abs(y));
    return z;
}

```

```

double distance(Chrm Chrm1, Chrm Chrm2)
{
    int sum = 0;
    for (int i = 1; i < 11; i++)
    {
        sum += Chrm1[i] * pow(2, (i - 1));
    }
    int x1 = sum / 1023 * 5;
    if (Chrm1[0] == 0)
    {
        x1 = x1 * (-1);
    }

    sum = 0;
    for (int i = 12; i < 22; i++)
    {
        sum += Chrm1[i] * pow(2, (i - 12));
    }
    int y1 = sum / 1023 * 5;
    if (Chrm1[11] == 0)
    {
        y1 = y1 * (-1);
    }
}

```

```

sum = 0;
for (int i = 1; i < 11; i++)
{
    sum += Chrm2[i] * pow(2, (i - 1));
}
int x2 = sum / 1023 * 5;
if (Chrm2[0] == 0)
{
    x2 = x2 * (-1);
}

sum = 0;
for (int i = 12; i < 22; i++)
{
    sum += Chrm2[i] * pow(2, (i - 12));
}
int y2 = sum / 1023 * 5;
if (Chrm2[11] == 0)
{
    y2 = y2 * (-1);
}

double dist = sqrt((x1 - x2)*(x1 - x2) + (y1 - y2)*(y1 - y2));
return dist;
}

```

```

Pplt makePplt()
{
    Pplt result;
    for (int i = 0; i < 20; i++)
    {

```

```

        Chrm Chrm;
        for (int j = 0; j < 22; j++)
        {
            Chrm.push_back(rand() % 2);
        }
        result.push_back(Chrm);
    }
    return result;
}

```

```

Pplt makePplt();
double fitness(Chrm Chrm);
double s_function(Chrm Chrm);
bool sortPpl(const Chrm &Chrm1, const Chrm &Chrm2);
Pplt newPplt();
double distance(Chrm Chrm1, Chrm Chrm2);
double sharing_fitness(Chrm Chrm);

```

```

int main()
{
    srand(time(NULL));
    ofstream file("file.txt", ios::out);

    population = makePplt();
    sort(population.begin(), population.end(), sortPplt);

    double aver = 0.0;
    for (int noChanged = 0, k = 0; noChanged < 30 && k < 500; k++)
    {
        population = newPplt();
        sort(population.begin(), population.end(), sortPplt);
    }
}

```

```

        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;
}

double s_function(Chrm Chrm)
{
    double sigma = 1.0;

    double s = 0.0;

    for (int i = 0; i < 20; i++)

```

```

    {
        double d = distance(Chrm, population[i]);
        if (d < sigma)
        {
            s += (1 - (d / sigma));
        }
    }

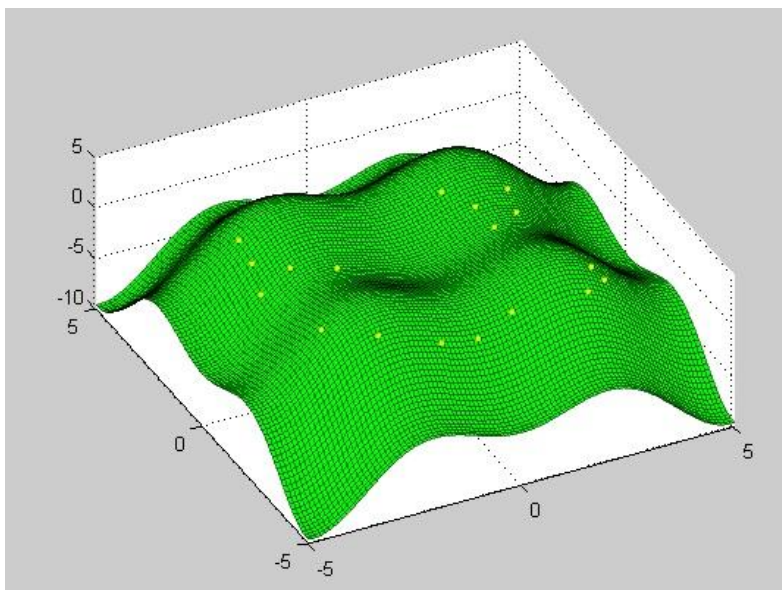
    return s;
}

double sharing_fitness(Chrm Chrm)
{
    double f = fitness(Chrm) / s_function(Chrm);
    return f;
}

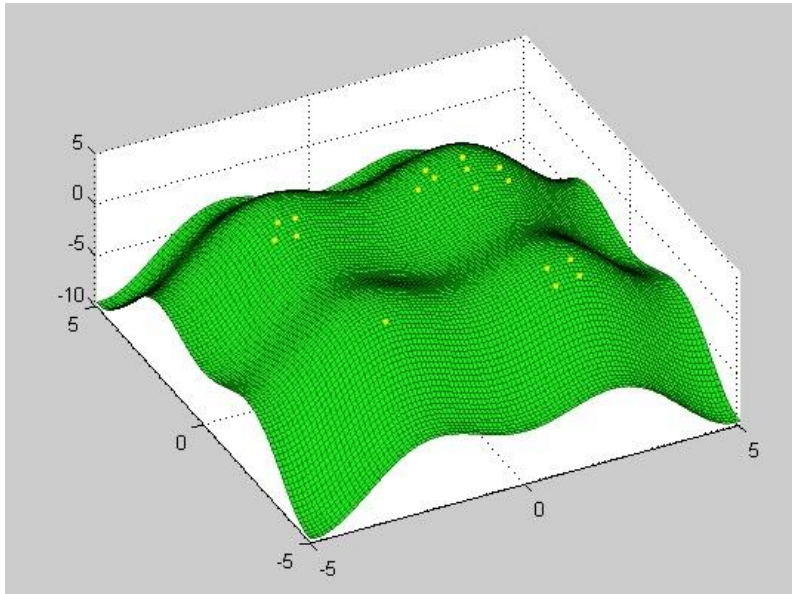
bool sortPplt(const Chrm &Chrm1, const Chrm &Chrm2)
{
    return sharing_fitness(Chrm1) > sharing_fitness(Chrm2);
}

```

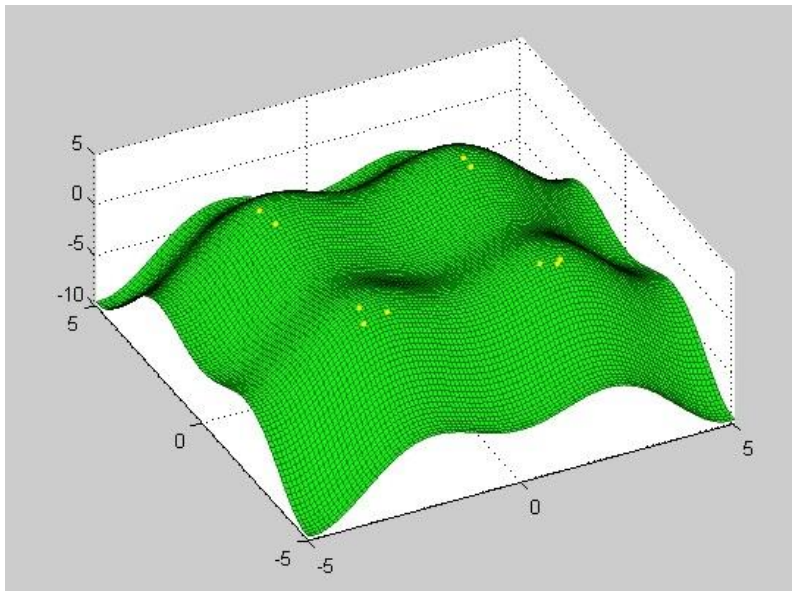
1st iteration



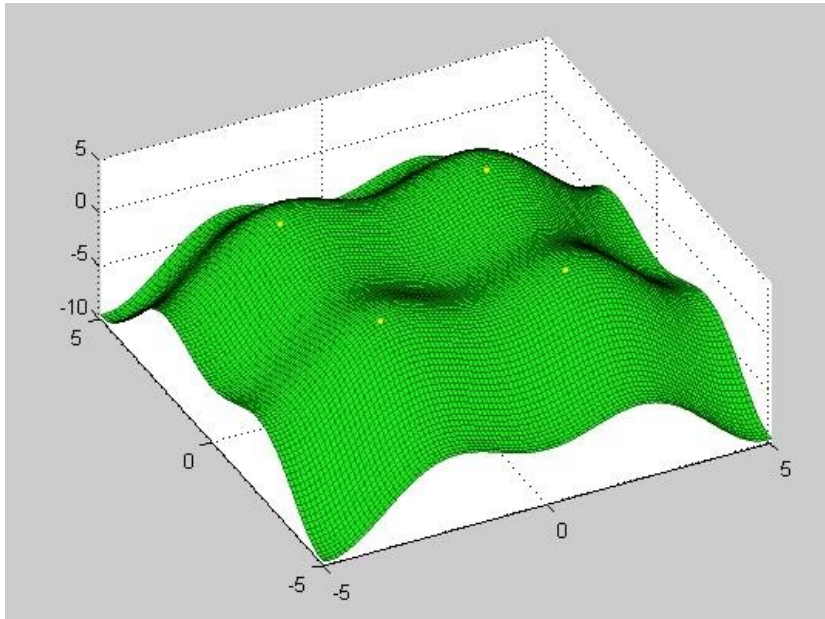
5th iteration



10th iteration



15th iteration



20th iteration

