

Kiryl Kiryluk

1. Maximize the following y in 3-D cases.

$$y = x_1 * \sin(|x_1|) + x_2 * \sin(|x_2|)$$

2. Create a population of 20 chromosomes at random, with fitness being y.

3. Evolve this population till fitness doesn't change

CODE:

```
#include <stdafx.h>

#include <fstream>

#include <iostream>

#include <vector>

#include <algorithm>

#include <time.h>

#include <math.h>

typedef vector<Chromosome> Population;

int Chromosome::getGen(int index)

{

    return genes[index];

}

bool sortPopulation(Chromosome &chromosome1, Chromosome &chromosome2);

Population distance(Population population);

void setGen(int index, int newGen);

void mutation();

void currentPosition();

int getDistance();

int getGen(int index);

void Chromosome::currentPosition()

for (int i = 0; i < genes.size(); i++)

    switch (genes[i])

return distance;

}

Population uniformDistance(Population population);

int XY(int tmp);
```

```

void Chromosome::setGen(int index, int newGen)

genes[index] = newGen;

void Chromosome::mutation()

{for (int i = 0; i < genes.size(); i++)

{int tmp = rand() % 100;

if (tmp == 0)

{

genes[i] = rand() % 4 + 1;

bool sortPopulation(Chromosome &chromosome1, Chromosome &chromosome2)

class Gen { private List hromosom;

private double fitness; public Gen(Random rnd)

{ hromosom = new List(); for (int i = 0; i < 22; i++) hromosom.Add(rnd

.Next() % 2); List temp = new List(parse(hromosom));

fitness = temp[0] * Math.Sin(Math.Abs(temp[0])) + temp[1] * Math.Sin(Math.Abs(temp[1])); }

Population distance(Population population)

{

int distancePoint;

int firstParent, secondParent;

Population result;

for (int i = 0; i < 10; i++)

{

distancePoint = rand() % 1000;

firstParent = rand() % 10;

secondParent = rand() % 10;

while (firstParent != secondParent)

{

secondParent = rand() % 10;

}

Population uniformDistance(Population population)

{

int firstParent, secondParent;

Population result;

for (int i = 0; i < 10; i++)

```

```

{firstParent = rand() % 10;

secondParent = rand() % 10;

while (firstParent != secondParent)

{

secondParent = rand() % 10;

}

intmain()

{

srand(time(NULL));

ofstream file("OutData.txt", ios::out);

ofstreamfileTable("Table.txt", ios::out);

Population population;

Population table;

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

{

Chromosome tmp;

population.push_back(tmp);

}

for (int i = 0; i < population.size(); i++)

{

population[i].currentPosition();

}

sort(population.begin(), population.end(), sortPopulation);

for (int k = 0; k <= 400; k++)

{

if ((k % 100) == 0)

{

for (int i = 0; i < 3; i++)

{

table.push_back(population[i])}

population = uniformDistance(population);

for (int i = 0; i < population.size(); i++)

{

```

```

population[i].currentPosition();
}

sort(population.begin(), population.end(), sortPopulation);

int min = population[0].getDistance();

int average = 0;

for (int i = 0; i < population.size(); i++)
{
    average += population[i].getDistance();
}

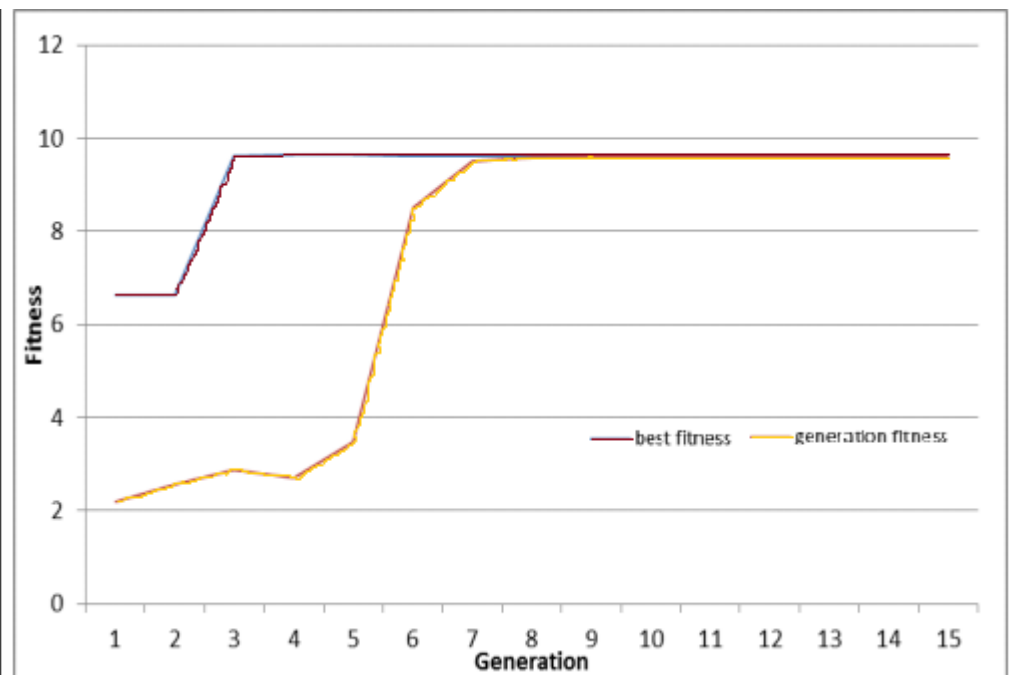
average = average / 20;

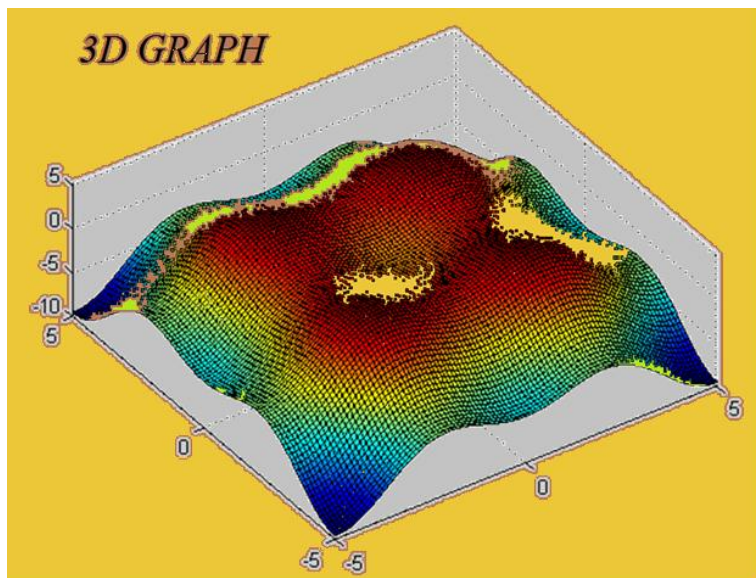
file<< max << "\t" << endl;

for (int k = 0; k <= 400; k++)
{
    if ((k % 100) == 0)
    {
        for (int i = 0; i < 3; i++)
            table.push_back(population[i]);
    }

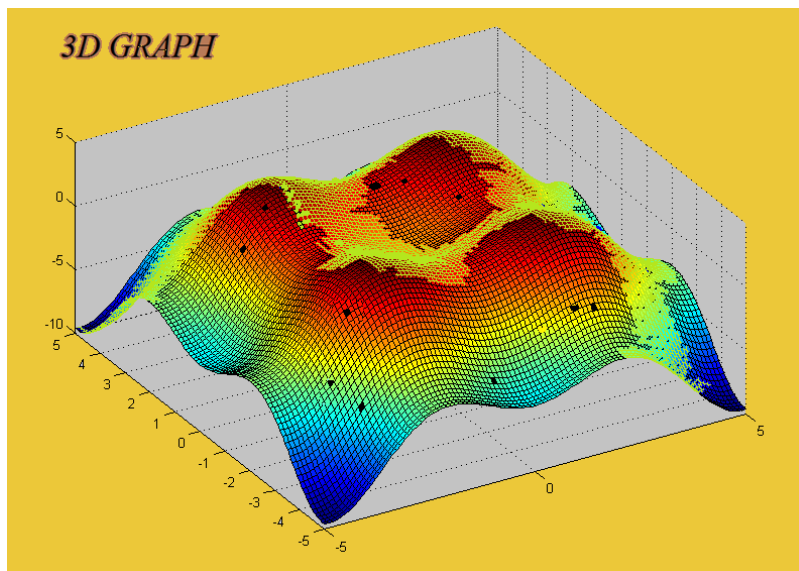
    population = Distance(population);
}

```

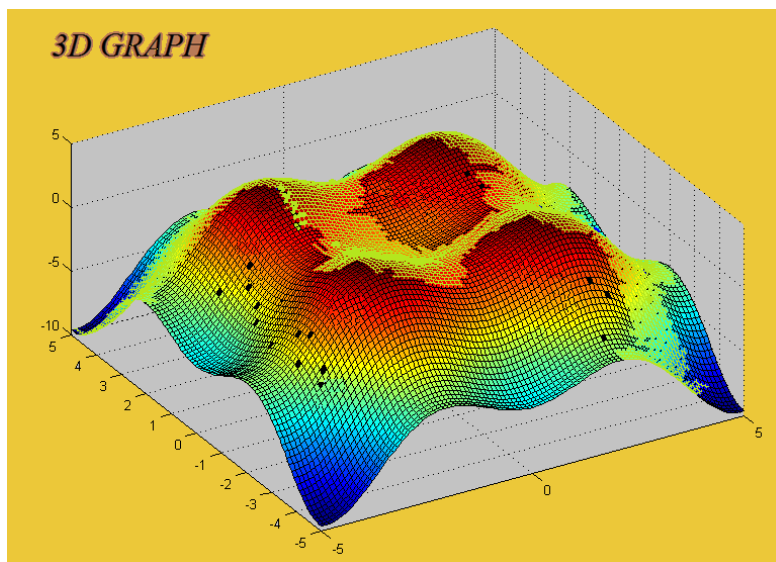




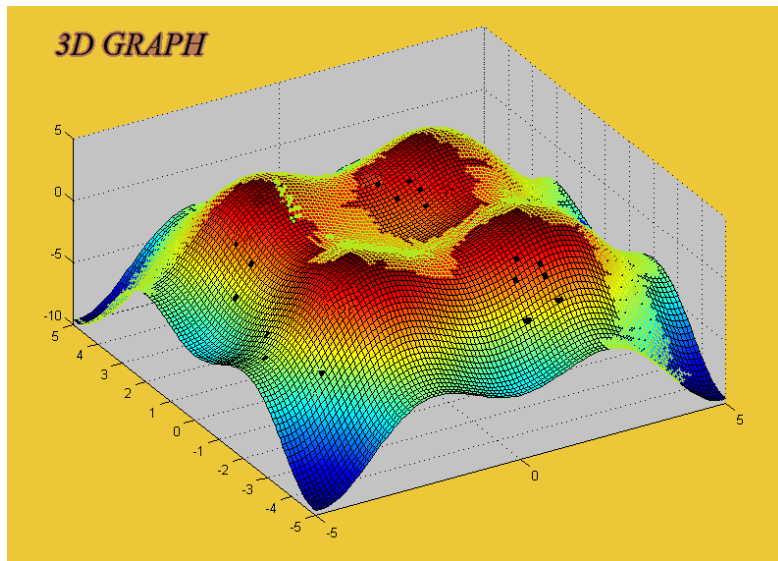
#1 Iteration



#25 Iteration



#50 Iteration



#100 Iteration

