Student – Vlad Golovchik (ii-10)

Task 2 (Lucky dog)

We must get dog that get the sausage and eat it). we place dog at (500;500) coordinates, sausage at (800;800). The best dog which reach sausage we get throuth the previous algoritm. But we have not binary chromosome, we have 1(go up), 0(go down), 2(go left), 3(go right). Fitness is the number of steps to sausage. The less fitness the bettet. It is mean that dog is closer to sausage.

Create graph on which display routes 3 random dog.
Pick 3 best dogs from the last population. Create graph with routes.

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

structcoord
{
	int X;
	int Y;
};

int stepsToSousage(intdog[1000])
{
	coord c;
	c.X = 500;
	c.Y = 500;
	for (int i = 0; i<1000; i++)
	{
		switch (dog[i]) {
		case 0:
			c.Y--;
			break;
		case 1:
			c.Y++;
			break;
		case 2:
			c.X--;
			break;
		case 3:
			c.X++;
			break;
		default:
			cout << "Error in \'stepsToSousage\'" << endl;
			exit(1);
			break;
		}
	}

	int steps = 0;
	steps = abs(c.X - 800) + abs(c.Y - 800);
	return steps;
}
```

```
coord getCoord(intdog[1000])
{
	coord c;
	c.X = 500;
	c.Y = 500;
	for (int i = 0; i<1000; i++)
	{
		switch (dog[i]) {
		case 0:
			c.Y--;
			break;
		case 1:
			c.Y++;
			break;
		case 2:
			c.X--;
			break;
		case 3:
			c.X++;
			break;
		default:
			cout << "Error in \'getCoord\'" << endl;
			exit(1);
			break;
		}
	}
	return c;
}

int main()
{
	srand(time(NULL));
	ofstream AvrF("e:\\AvrFitness_dogs.txt");
	ofstream BestFitnessF("e:\\BestFitness_dogs.txt");

	ofstream S[5];
	ofstream SX[5];
	S[0].open("d.txt1X.txt");
	S[1].open("d.txt2X.txt");
	S[2].open("d.txt3X.txt");


	SX[0].open("d.txt1X.txt");
	SX[1].open("d.txt2X.txt");
	SX[2].open("d.txt3X.txt");


	ofstream F[3];
	ofstream FX[3];
	F[0].open("d.txt1X.txt");
	F[1].open("d.txt2X.txt");
	F[2].open("d.txt3X.txt");


	FX[0].open("d.txt1X.txt");
	FX[1].open("d.txt2X.txt");
	FX[2].open("d.txt3X.txt");


	ofstream Opt("d.txt");


	int osob[100][1000];
```

```
int iter = 0;
double avr = 0;
double numb = 0;
for (int i = 0; i<100; i++)
for (int j = 0; j<1000; j++)
{

	osob[i][j] = rand() % 4;


	avr += osob[i][j];
}
cout << "//M=" << (avr / (100 * 1000)) << endl;
int temp[1000] = { 0 };
int best[50][1001] = { 0 };
do
{
	for (int i = 0; i < 100; i++)
	{
		int min_fitness = i;
		for (int j = i + 1; j < 100; j++)
		{
			int fitMIN = stepsToSousage(osob[min_fitness]);
			int fitJ = stepsToSousage(osob[j]);;
			if (fitJ<fitMIN)min_fitness = j;
		}

		if (min_fitness != i)
		{
			for (int k = 0; k<1000; k++)
				temp[k] = osob[i][k];

			for (int k = 0; k<1000; k++)
				osob[i][k] = osob[min_fitness][k];

			for (int k = 0; k<1000; k++)
				osob[min_fitness][k] = temp[k];
		}
	}



	double AvrFit = 0;
	int BestFit = 0;
	int numOfBest = 0;
	coord c;
	for (int i = 0; i<50; i++)
	{
		c = getCoord(osob[i]);
		if (c.X<0 || c.Y<0)continue;
		numOfBest++;


		for (int j = 0; j<1000; j++)
		{
			best[i][j] = osob[i][j];

		}
		AvrFit += stepsToSousage(best[i]);


	}
```

```
        AvrFit = AvrFit / 50;
        cout << "//AvrFit = " << AvrFit << endl;

        BestFit = stepsToSousage(best[0]);

        AvrF << AvrFit;
        BestFitnessF << BestFit;
        cout << "//BestFit = " << BestFit << endl;
        AvrF << '\n';
        BestFitnessF << '\n';

        if (BestFit == 0)break;

        int mom = 0;
        int dad = 0;
        int dog = 0;
        int frstChild[1000] = { 0 };
        int ScndChild[1000] = { 0 };
        for (int i = 0; i<100; i++)
        {
                do
                {
                        mom = rand() % numOfBest;
                        dad = rand() % numOfBest;
                } while (mom == dad);
                cat = rand() % 1000;
                for (int k = 0; k<cat; k++)
                {
                        frstChild[k] = best[mom][k];
                        ScndChild[k] = best[dad][k];
                }

                for (int k = cat; k<1000; k++)
                {
                        frstChild[k] = best[dad][k];
                        ScndChild[k] = best[mom][k];
                }
                int fitness1 = 0;
                int fitness2 = 0;

                fitness1 = stepsToSousage(frstChild);
                fitness2 = stepsToSousage(ScndChild);

                if (fitness1<fitness2)
                {
                        for (int k = 0; k<1000; k++)
                                osob[i][k] = frstChild[k];
                }
                else
                {
                        for (int k = 0; k<1000; k++)
                                osob[i][k] = ScndChild[k];
                }

        }

        iter++;

} while (iter<100);

for (int s = 0; s<3; s++)
{
```

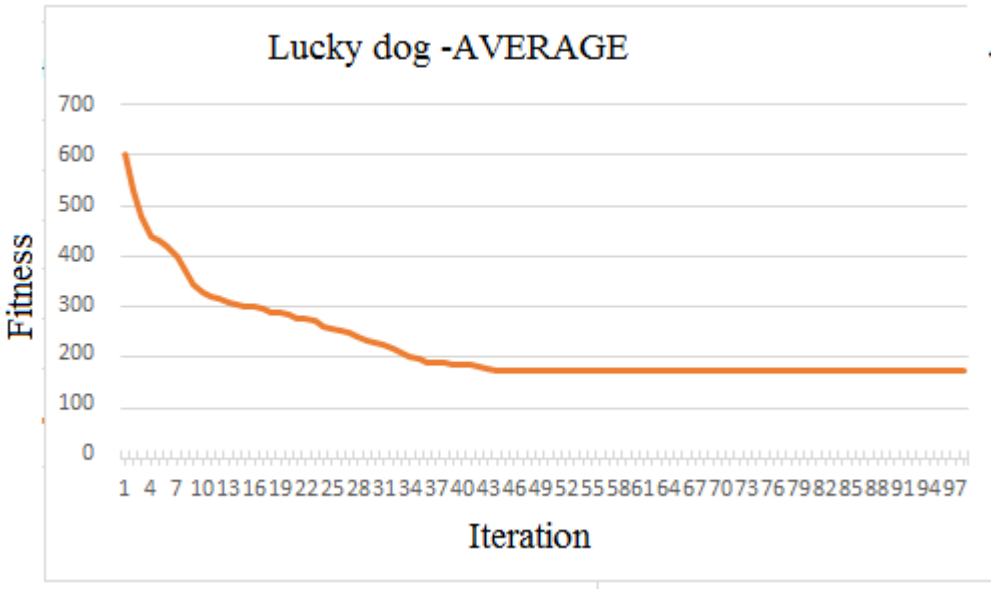
```
                coord c;
                c.X = 500;
                c.Y = 500;
                for (int i = 0; i<1000; i++)
                {
                        switch (best[s][i]) {
                        case 0:
                                c.Y--;
                                break;
                        case 1:
                                c.Y++;
                                break;
                        case 2:
                                c.X--;
                                break;
                        case 3:
                                c.X++;
                                break;
                        default:
                                cout << "Error in \'getCoord\'" << endl;
                                exit(1);
                                break;
                        }
                        F[s] << c.Y << '\n';
                        FX[s] << c.X << '\n';

                }
        }
        int total = 800;
        for (int i = 0; i<150; i++)
        {
                Opt << (total -= 4) << "\n";
        }



        return 0;
}


                if (iter == 0)
                {
                        for (int s = 0; s<3; s++)
                        {
                                coord c;
                                c.X = 500;
                                c.Y = 500;
                                for (int i = 0; i<1000; i++)
                                {
                                        switch (osob[s][i]) {
                                        case 0:
                                                c.Y--;
                                                break;
                                        case 1:
                                                c.Y++;
                                                break;
                                        case 2:
                                                c.X--;
                                                break;
                                        case 3:
                                                c.X++;
                                                break;
                                        default:
                                                cout << "Error in \'getCoord\'" << endl;
```

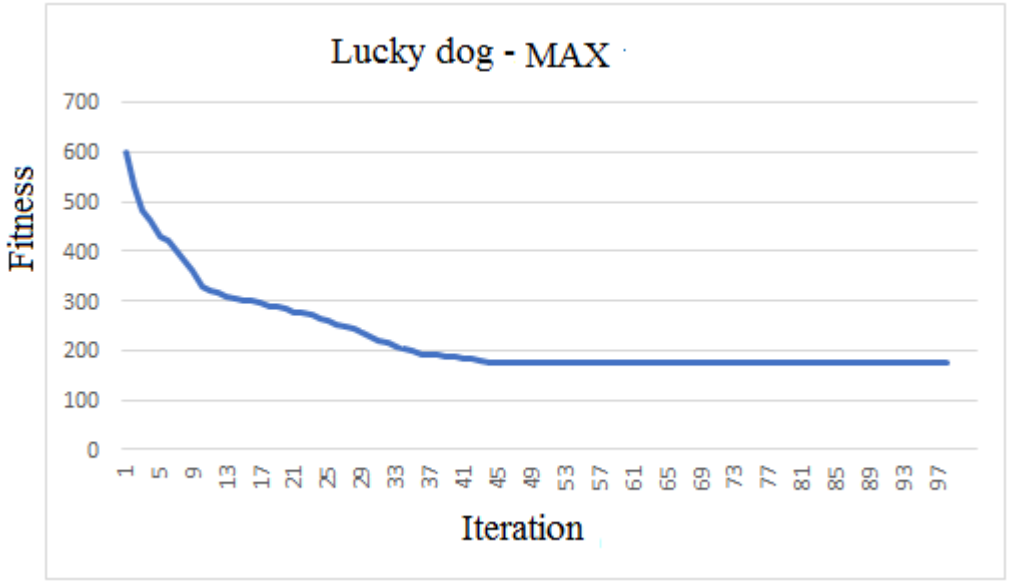
```
                                        exit(1);
                                        break;
                                }
                                S[s] << c.Y << "\n";
                                SX[s] << c.X << "\n";

                        }
                }
        }
```

```
int isFitnessGood(intosob[50][1000])
{
        int count = 0;
        for (int i = 0; i<50; i++)
        {
                count = 0;
                for (int j = 0; j<1000; j++)
                {
                        count += osob[i][j];
                }
                if (count == 1000)return i;
        }
        return -1;
}
```

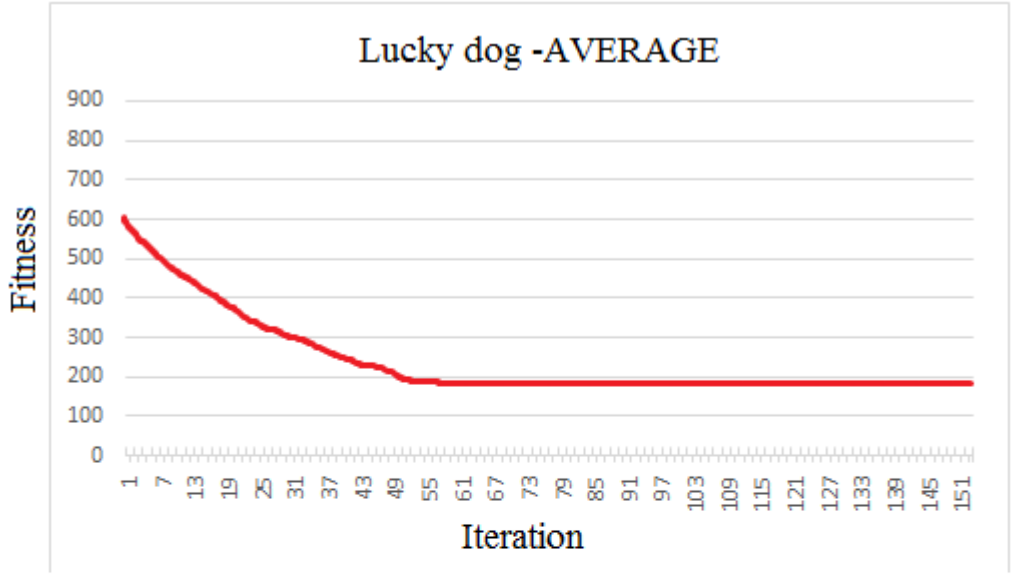
Results :

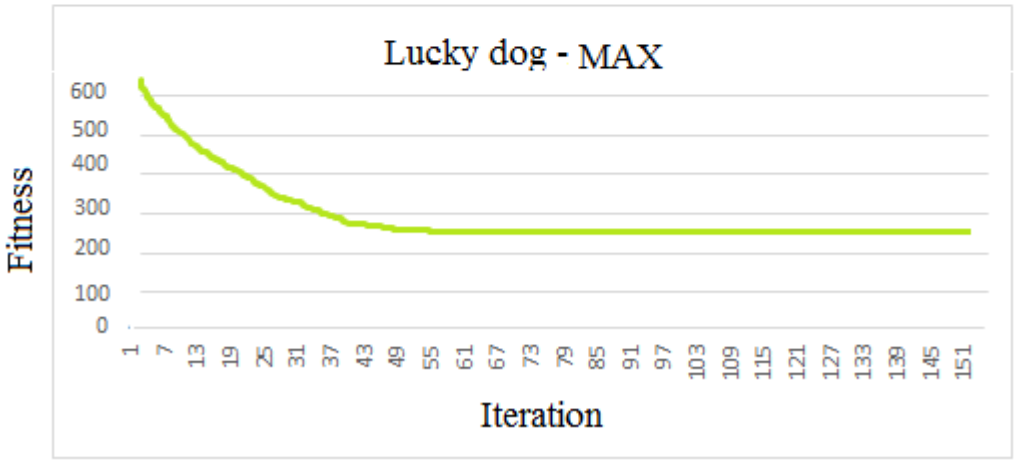
For one point crossover:





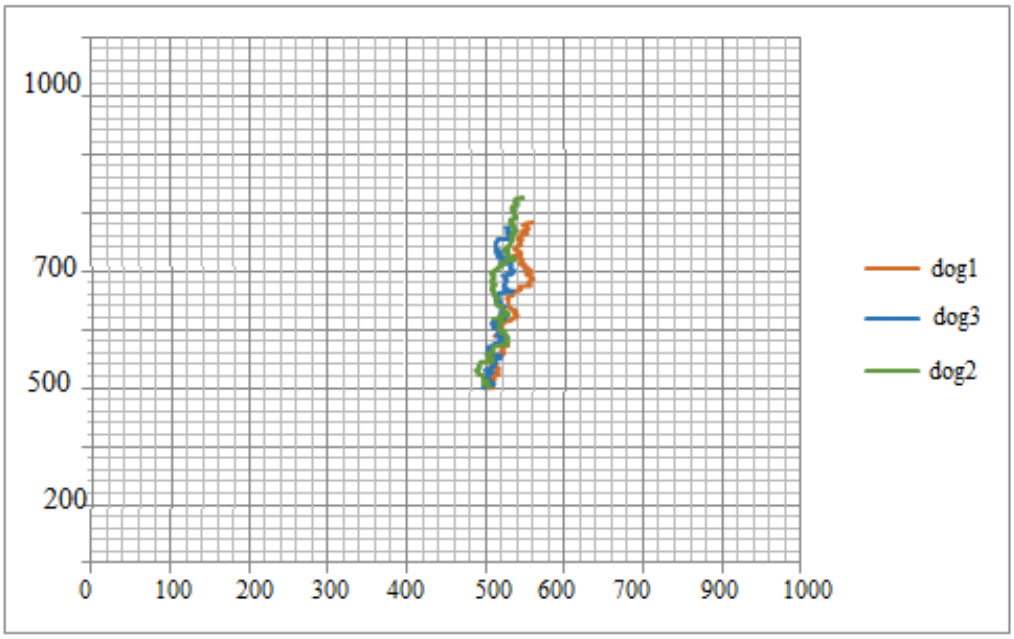
For uniform crossover:



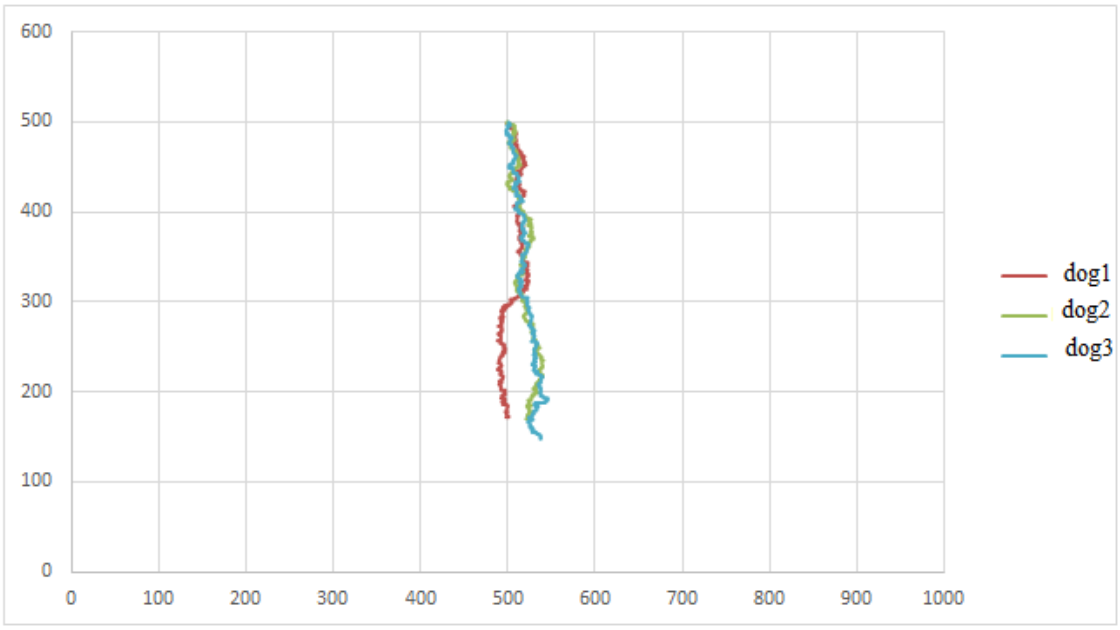


For one point crossover:

1)one iteration



2)50 iteration


600
500
400
300
200
100
0
0
100
200
300
400
500
600
700
800
900
1000
dog1
dog2
dog3