DENIS RAMSKIY II 11

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LakyDog
{
    public class Gen
    {
        private List<int> hromosom;
        private double fitness;
        private double shfitness;
        private const int max=1000;

        public Gen(Random rnd,Points mypnt)
        {
            shfitness = 0;
            hromosom = new List<int>();
            for (int i = 0; i < max; i++)
                hromosom.Add(rnd.Next() % 4);
            fitness = calcFit(mypnt);
        }
        public Gen(Gen obj)
        {
            this.hromosom = new List<int>();
            for (int i = 0; i < max; i++)
                this.hromosom.Add(obj.get()[i]);
            this.fitness = obj.get_fitness();
            this.shfitness = obj.get_shfitness();
        }
        public Gen(List<int> obj, Points mypnt)
        {
            shfitness = 0;
            this.hromosom = new List<int>();
            for (int i = 0; i < max; i++)
                this.hromosom.Add(obj[i]);
            fitness = calcFit(mypnt);
        }
        public void mutation(Random rnd)
        {
            for (int i = 0; i < max; i++)
                if (rnd.Next() % 1000 < 2)
                    hromosom[i] = rnd.Next() % 4;
        }
        private int calcFit(Points mypnt)
        {
            int result = 1000;
            for (int i = 0; i < mypnt.getCount(); i++)
            {
                int temp1 = Math.Abs(mypnt.getPoint(i).getPoints()[0] - corOfEnd()[0]);
                int temp2 = Math.Abs(mypnt.getPoint(i).getPoints()[1] - corOfEnd()[1]);
                if (temp1 + temp2 < result)
                    result = temp1 + temp2;
            }

            return result;
        }
        static public Gen operator +(Gen obj1, Gen obj2)
        {
            obj1.get().Clear();
            for (int i = 0; i < max; i++)
                obj1.get().Add(obj2.get()[i]);
            obj1.set_fitness(obj2.get_fitness());
            obj1.set_shfitness(obj2.get_shfitness());
            return obj1;
        }
```

```
        public List<int> get()
        {
            return hromosom;
        }
        public double get_fitness()
        {
            return fitness;
        }
        public void set_fitness(double a)
        {
            this.fitness = a;
        }
        public double get_shfitness()
        {
            return shfitness;
        }
        public void set_shfitness(double a)
        {
            this.shfitness = a;
        }
        public List<int> corOfEnd()
        {
            List<int> temp = new List<int>();
            int x = 500, y = 500;
            for (int i = 0; i < max; i++)
            {
                switch (hromosom[i])
                {
                    case 0: x--; break;
                    case 1: x++; break;
                    case 2: y++; break;
                    case 3: y--; break;
                }
            }
            temp.Add(x);
            temp.Add(y);

            return temp;
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LakyDog
{
    public class Generation
    {
        private List<Gen> generation;
        private double fitness;
        private double shfitness;
        private const int max = 1000;

        public Generation(Random rnd, Points mypnt)
        {
            generation = new List<Gen>();
            for (int I = 0; I < 20; I++)
            {
                Gen temp = new Gen(rnd, mypnt);
                generation.Add(temp);
                fitness += (double)temp.get_fitness()/20;
            }

            for (int I = 0; I < 20; I++)
```

```
    {
        calcSherFitness(generation, generation[I]);
        shfitness += (double)generation[I].get_shfitness() / 20;
    }
    this.shsort();
}
public void shsort()
{
    for (int i = 0; i < 20; i++)
        for (int j = 0; j < 19; j++)
            if (generation[j].get_shfitness() > generation[j + 1].get_shfitness())
            {
                Gen temp = new Gen(generation[j]);
                generation[j] += generation[j + 1];
                generation[j + 1] += temp;
            }
}
public List<Gen> get_parents(Random rnd)
{
    List<Gen> TEMP = new List<Gen>();
    TEMP.Add(generation[rnd.Next() % 10]);
    TEMP.Add(generation[rnd.Next() % 10]);
    return TEMP;
}
public Gen get_child(Random rnd, List<Gen> par, Points mypnt)
{
    List<int> TEMP = new List<int>();
    for (int i = 0; i < max; i++)
        if (rnd.Next() % 2 == 0)
            TEMP.Add(par[0].get()[i]);
        else
            TEMP.Add(par[1].get()[i]);

    Gen child = new Gen(TEMP, mypnt);
    child.mutation(rnd);

    return child;
}
public void calcSherFitness(List<Gen> generation,Gen obj)
{
    double shFitness = 0;
    for (int i = 0; i < 20; i++)
    {
        int cor_1 = Math.Abs(obj.corOfEnd()[0] - generation[i].corOfEnd()[0]);
        int cor_2 = Math.Abs(obj.corOfEnd()[1] - generation[i].corOfEnd()[1]);
        if (cor_1 + cor_2 <= 50)
            shFitness += 1 - (cor_1 + cor_2)/50;
    }

    if (shFitness == 0)
        obj.set_shfitness(obj.get_fitness());
    else
        obj.set_shfitness(obj.get_fitness() / shFitness);
}
public void new_generation(Random rnd, Points mypnt, int generationCount)
{
    List<Gen> ngeneration = new List<Gen>();
    double nfitness = 0;
    double nshfitness = 0;
    for (int i = 0; i < 20; i++)
    {
        ngeneration.Add(this.get_child(rnd, this.get_parents(rnd), mypnt));
        nfitness += (double)ngeneration[i].get_fitness()/20;
    }

    for (int i = 0; i < 20; i++)
    {
        calcSherFitness(ngeneration, ngeneration[i]);
        this.generation[i] += ngeneration[i];
```

```
                nshfitness += (double)ngeneration[i].get_shfitness() / 20;
            }
            this.fitness = nfitness;
            this.shfitness = nshfitness;

            this.shsort();
        }

        public double get_fitness()
        {
            return fitness;
        }
        public double get_shfitness()
        {
            return shfitness;
        }
        public List<Gen> get()
        {
            return generation;
        }
        public Gen get(int i)
        {
            return generation[i];
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LakyDog
{
    public class Point
    {
        public int x;
        public int y;

        public Point(int a, int b)
        {
            x = a;
            y = b;
        }
        public List<int> getPoints()
        {
            List<int> temp = new List<int>();
            temp.Add(x);
            temp.Add(y);
            return temp;
        }
    }
    public class Points
    {
        private List<Point> list;

        public Points()
        {
            list = new List<Point>();
            list.Add(new Point(200, 200));
            list.Add(new Point(200, 800));
            list.Add(new Point(800, 200));
            list.Add(new Point(800, 800));
        }
        public int getCount()
        {
            return list.Count;
        }
```

```
        public Point getPoint(int i)
        {
            return list[i];
        }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;

namespace LakyDog
{
    public partial class Form1 : Form
    {
        int generationCount;
        Random rnd ;
        Generation work ;
        Points wPoint;
        public Form1()
        {
            InitializeComponent();
            generationCount = 0;
            wPoint = new Points();
            rnd = new Random(DateTime.Now.Millisecond);
            work = new Generation(rnd, wPoint);
            pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
            File.WriteAllText("shFitnessBest.txt","");
            File.WriteAllText("shFitnessAverege.txt", "");
            File.WriteAllText("FitnessBest.txt", "");
            File.WriteAllText("FitnessAverege.txt", "");
        }

        private void button1_Click(object sender, EventArgs e)
        {
            generationCount++;
            Bitmap new_map = new Bitmap(1000,1000);
            Pen myPen = new Pen(Color.Red);
            Graphics graph = Graphics.FromImage(new_map);

            work.new_generation(rnd, wPoint,generationCount);
            for (int i = 0; i < 20; i++)
            {
                int x = 500, y = 500;
                for (int j = 0; j < 1000; j++)
                {
                    int nx = x, ny = y;
                    switch (work.get(i).get()[j])
                    {
                        case 0: nx--; break;
                        case 1: nx++; break;
                        case 2: ny++; break;
                        case 3: ny--; break;
                    }
                    graph.DrawLine(myPen, x, y, nx, ny);
                    x = nx; y = ny;
                }
            }
            myPen.Color = Color.Green;
            myPen.Width = 8;

            for (int i = 0; i < wPoint.getCount(); i++)
```

```
                graph.DrawEllipse(myPen, wPoint.getPoint(i).getPoints()[0] - 2,
wPoint.getPoint(i).getPoints()[1] - 2, 4, 4);
            pictureBox1.Image = new_map;

            textBox1.Text = Math.Round(work.get_shfitness(),2).ToString();
            textBox2.Text = generationCount.ToString();

            File.AppendAllText("shFitnessBest.txt", Math.Round(work.get_shfitness(), 2).ToString() +
"\r\n");
            File.AppendAllText("shFitnessAverege.txt", Math.Round(work.get()[0].get_shfitness(),
2).ToString() + "\r\n");
            File.AppendAllText("FitnessBest.txt", Math.Round(work.get_fitness(), 2).ToString() +
"\r\n");
            File.AppendAllText("FitnessAverege.txt", Math.Round(work.get()[0].get_fitness(),
2).ToString() + "\r\n");



            File.WriteAllText(".\\Info\\shFitness.txt", "");
            File.WriteAllText(".\\Info\\Fitness.txt", "");
            for (int i = 0; i < 20; i++)
            {
                File.AppendAllText(".\\Info\\shFitness.txt",
work.get()[i].get_shfitness().ToString()+"\r\n");
                File.AppendAllText(".\\Info\\Fitness.txt", work.get()[i].get_fitness().ToString() +
"\r\n");
            }
        }
    }
}
```

MY RESULT

Best Fitness	Averege Fitness
580	570,3
562	569,1
560	572,6
568	570,6
546	568,2
560	570,6
556	567,5
554	567,6
550	570,2
546	563,7
528	551,7
530	547,4
526	541,4
508	533,4
512	526
502	524,7
498	518,5
490	512
484	502,1
482	496,9
478	496,5
472	490,9
470	480,4
460	473,3

Best Shared Fitness	Averege Shared Fitness
30,53	40,78
31,22	42,14
28	31,05
28,4	33,07
27,3	30,02
28	30,6
27,8	29,37
27,7	28,84
27,5	28,98
27,3	28,65
26,4	28,22
26,5	28,31
26,3	27,21
25,4	26,67
25,6	26,44
25,1	26,37
24,9	25,92
24,5	25,73
24,2	25,1
24,1	24,85
23,9	24,83
23,6	24,54
23,5	24,02
23	23,67

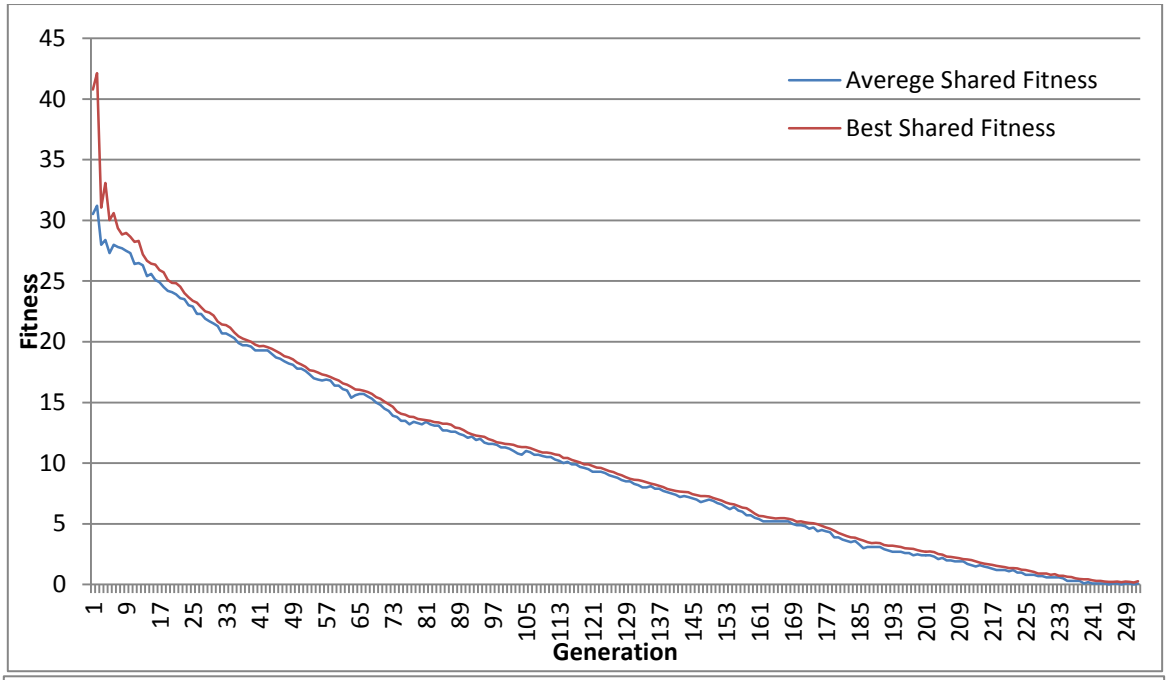
458	467,6	22,9	23,38
446	464,3	22,3	23,22
446	457,3	22,3	22,86
438	450	21,9	22,5
434	447,9	21,7	22,4
430	443,3	21,5	22,16
426	433,4	21,3	21,67
414	428,4	20,7	21,42
414	427,5	20,7	21,37
410	423,4	20,5	21,17
406	415,5	20,3	20,77
398	409,2	19,9	20,46
394	405,2	19,7	20,26
394	402,8	19,7	20,14
392	399,8	19,6	19,99
386	395,3	19,3	19,76
386	392,5	19,3	19,62
386	393,3	19,3	19,66
386	391,2	19,3	19,56
380	388,5	19	19,42
374	384,8	18,7	19,24
372	381,2	18,6	19,06
368	376,4	18,4	18,82
364	374,3	18,2	18,72
362	370,9	18,1	18,54
356	365,9	17,8	18,29
356	362,8	17,8	18,14
352	358,7	17,6	17,94
346	353,2	17,3	17,66
340	351,8	17	17,59
338	349,4	16,9	17,47
336	346	16,8	17,3
338	344,5	16,9	17,22
336	342	16,8	17,1
328	338,8	16,4	16,94
328	336	16,4	16,8
322	331,4	16,1	16,57
320	329,1	16	16,46
308	325,9	15,4	16,29
312	321,3	15,6	16,07
314	320,9	15,7	16,05
314	319,6	15,7	15,98
310	317,4	15,5	15,87
306	314,2	15,3	15,71
300	309	15	15,45
296	305,9	14,8	15,3
290	301,5	14,5	15,08
286	297,2	14,3	14,86
278	292,9	13,9	14,64

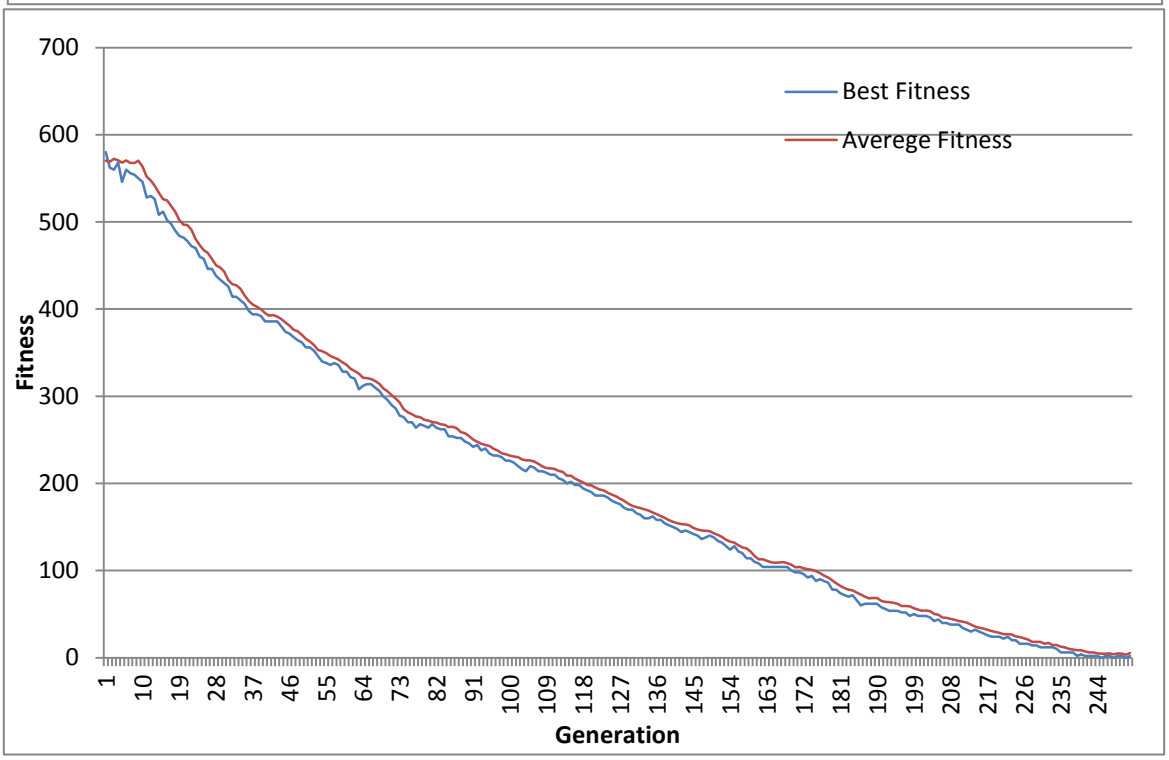
276	285,2	13,8	14,26
270	281,5	13,5	14,08
270	279,5	13,5	13,98
264	276,8	13,2	13,84
268	276,2	13,4	13,81
266	273	13,3	13,65
264	272	13,2	13,6
268	270,5	13,4	13,53
264	269,7	13,2	13,48
262	267,8	13,1	13,39
262	267,2	13,1	13,36
254	264,8	12,7	13,24
254	264,9	12,7	13,24
252	263,2	12,6	13,16
252	258,6	12,6	12,93
248	257,8	12,4	12,89
246	254,3	12,3	12,72
242	250,3	12,1	12,51
244	247,7	12,2	12,38
238	245,6	11,9	12,28
240	244,2	12	12,21
234	243,1	11,7	12,16
232	239,7	11,6	11,99
232	237,7	11,6	11,88
230	234,4	11,5	11,72
226	233,4	11,3	11,67
226	232,1	11,3	11,6
224	231	11,2	11,55
220	230,3	11	11,52
216	227,7	10,8	11,38
214	226,4	10,7	11,32
220	226,5	11	11,32
218	224,8	10,9	11,24
214	222,5	10,7	11,12
214	219,6	10,7	10,98
212	217,7	10,6	10,89
210	217,3	10,5	10,87
210	216,4	10,5	10,82
206	214,2	10,3	10,71
204	213,4	10,2	10,67
200	208,8	10	10,44
202	208,8	10,1	10,44
198	205,5	9,9	10,28
198	203,3	9,9	10,16
194	201,1	9,7	10,06
192	198,2	9,6	9,91
190	197,9	9,5	9,9
186	195,3	9,3	9,77
186	193	9,3	9,65

186	192	9,3	9,6
184	189,4	9,2	9,47
180	187,2	9	9,36
178	185,3	8,9	9,26
176	182,2	8,8	9,11
172	180,1	8,6	9,01
170	176,9	8,5	8,85
170	174,3	8,5	8,71
166	172,7	8,3	8,64
164	171,9	8,2	8,6
160	170,4	8	8,52
160	168,7	8	8,43
162	166,6	8,1	8,33
158	164,7	7,9	8,24
158	162,7	7,9	8,14
154	160,4	7,7	8,02
152	157,7	7,6	7,88
150	155,9	7,5	7,79
148	154,4	7,4	7,72
144	153,2	7,2	7,66
146	152,9	7,3	7,64
144	152	7,2	7,6
142	149	7,1	7,45
140	147,2	7	7,36
136	146	6,8	7,3
138	145,6	6,9	7,28
140	145,2	7	7,26
138	142,8	6,9	7,14
134	140,7	6,7	7,03
132	138,6	6,6	6,93
128	135,3	6,4	6,76
124	133,2	6,2	6,66
128	132,1	6,4	6,6
122	129,1	6,1	6,46
120	126,5	6	6,33
114	125,6	5,7	6,28
114	121,7	5,7	6,08
110	116,7	5,5	5,84
108	112,9	5,4	5,65
104	112,8	5,2	5,64
104	111,2	5,2	5,56
104	109,8	5,2	5,49
104	109	5,2	5,45
104	109,1	5,2	5,46
104	109,5	5,2	5,48
104	108,6	5,2	5,43
100	106,8	5	5,34
98	103,8	4,9	5,19
98	104	4,9	5,2

96	102,8	4,8	5,14
92	101,5	4,6	5,07
94	100,7	4,7	5,04
88	99,2	4,4	4,96
90	97	4,5	4,85
88	94,2	4,4	4,71
86	92,1	4,3	4,61
78	88,7	3,9	4,43
78	85,3	3,9	4,26
74	82,4	3,7	4,12
72	79,8	3,6	3,99
70	77,9	3,5	3,9
72	77,3	3,6	3,86
66	74,6	3,3	3,73
60	72,4	3	3,62
62	69,8	3,1	3,49
62	68,3	3,1	3,41
62	68,6	3,1	3,43
62	68,4	3,1	3,42
58	65,1	2,9	3,26
56	64	2,8	3,2
54	63,8	2,7	3,19
54	63	2,7	3,15
54	61,8	2,7	3,09
52	59,5	2,6	2,98
52	59,4	2,6	2,97
48	59,1	2,4	2,95
50	56,8	2,5	2,84
48	55,2	2,4	2,76
48	54	2,4	2,7
48	54,3	2,4	2,72
46	53,6	2,3	2,68
42	50,3	2,1	2,51
44	49,3	2,2	2,47
40	46	2	2,3
40	45,7	2	2,28
38	44,8	1,9	2,24
38	43,4	1,9	2,17
38	42	1,9	2,1
34	41,4	1,7	2,07
32	40,1	1,6	2,01
30	37,9	1,5	1,9
32	35,9	1,6	1,8
30	34,3	1,5	1,72
28	33,6	1,4	1,68
26	32,3	1,3	1,62
24	30,7	1,2	1,54
24	29,5	1,2	1,48
24	28,6	1,2	1,43

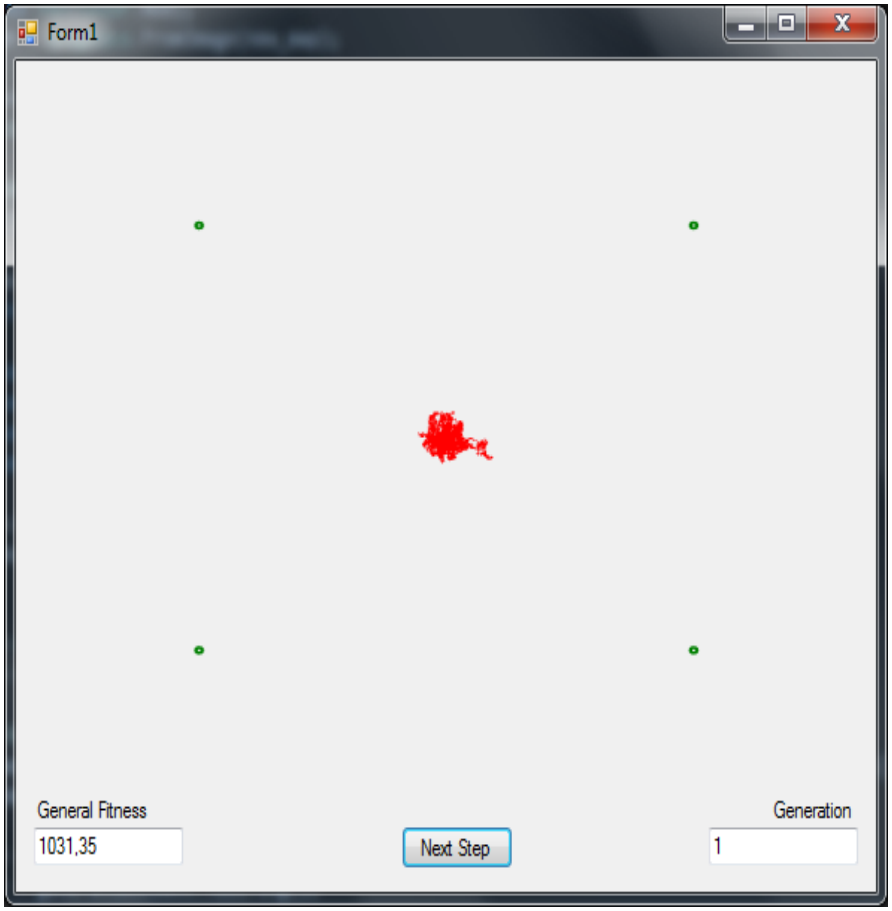
22	27	1,1	1,35
24	27	1,2	1,35
20	26,7	1	1,33
20	24,6	1	1,23
16	23,8	0,8	1,19
16	22,3	0,8	1,12
16	20,6	0,8	1,03
14	18,1	0,7	0,91
14	18,2	0,7	0,91
12	18,2	0,6	0,91
12	16	0,6	0,8
12	17,1	0,6	0,86
12	14,6	0,6	0,73
10	14,4	0,5	0,72
6	12,7	0,3	0,64
6	12,1	0,3	0,61
6	10,1	0,3	0,51
6	9,2	0,3	0,46
2	8,6	0,1	0,43
4	8,5	0,2	0,43
2	7	0,1	0,35
2	6,1	0,1	0,31
2	5,9	0,1	0,3
2	5,1	0,1	0,26
0	4,5	0	0,22
2	4,6	0,1	0,23
2	4,9	0,1	0,25
0	3,7	0	0,19
2	5	0,1	0,25
2	4,5	0,1	0,23
0	3,5	0	0,18
2	5,4	0,1	0,27


Averege Shared Fitness
Best Shared Fitness
Fitness
Generation
0
5
10
15
20
25
30
35
40
45
1
9
17
25
33
41
49
57
65
73
81
89
97
105
113
121
129
137
145
153
161
169
177
185
193
201
209
217
225
233
241
249

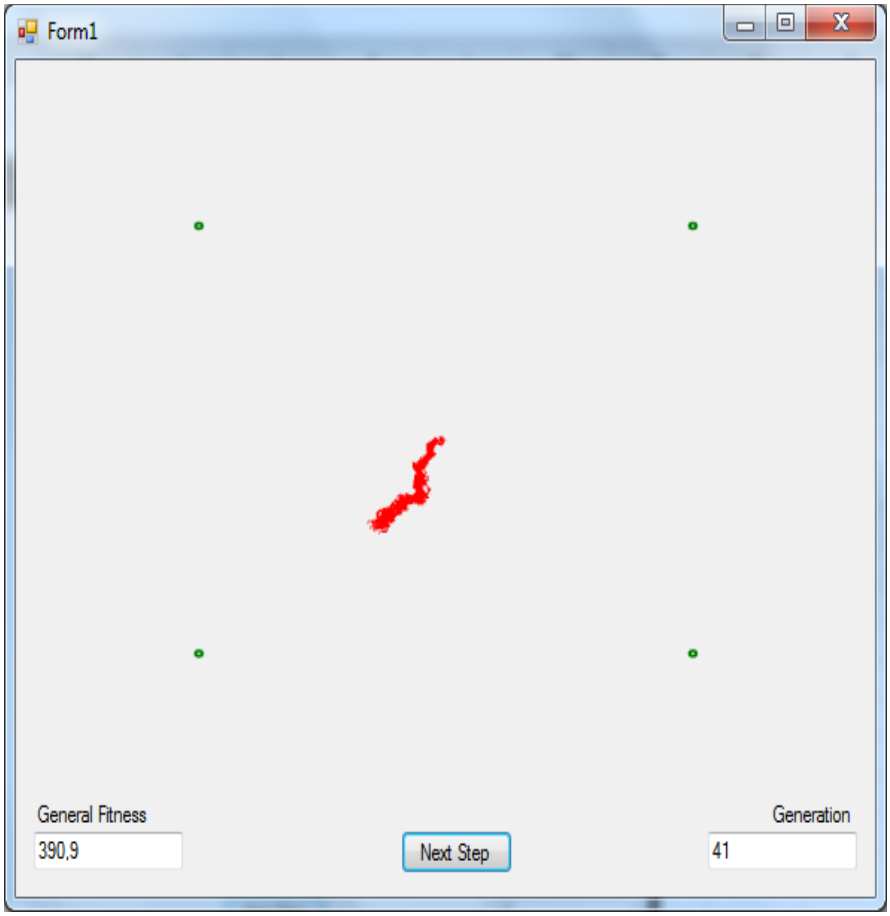

Best Fitness
Averege Fitness
Fitness
Generation
0
100
200
300
400
500
600
700
1
10
19
28
37
46
55
64
73
82
91
100
109
118
127
136
145
154
163
172
181
190
199
208
217
226
235
244

GENERATION INFORATION

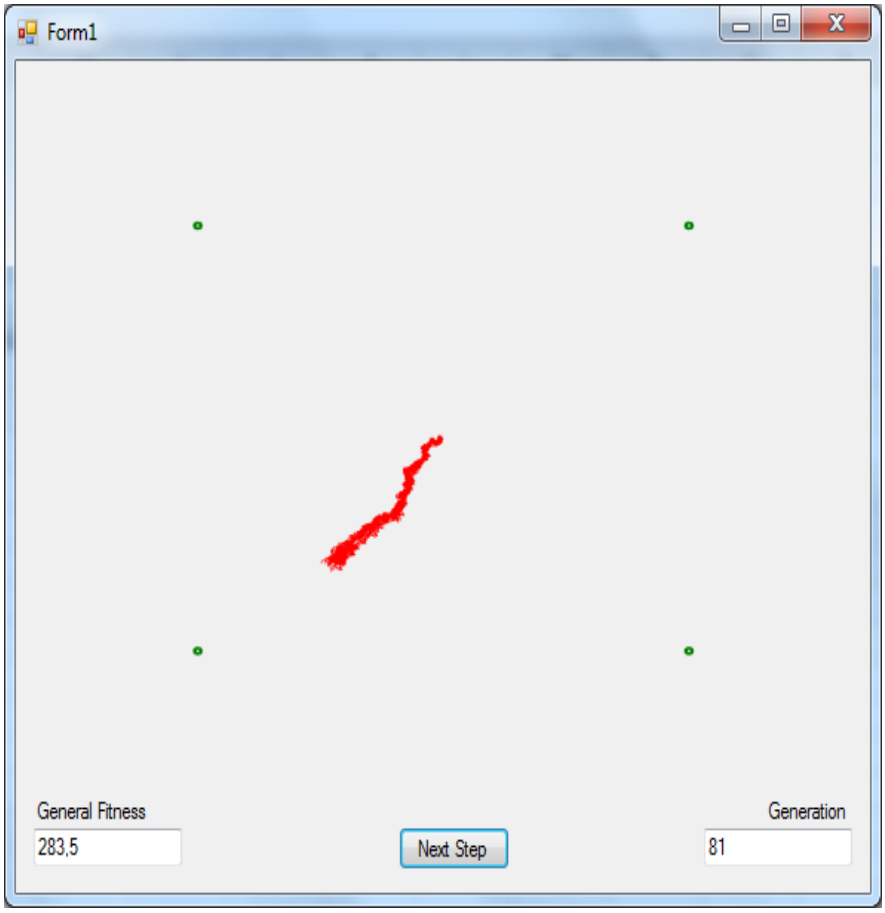
№	Fitness	Shared Fitness
1	578	32,1
2	592	32,9
3	596	33,1
4	578	34,0
5	586	34,5
6	588	34,6
7	570	35,6
8	576	36,0
9	582	36,4
10	584	36,5
11	586	36,6
12	564	37,6
13	578	38,5
14	578	41,3
15	564	43,4
16	564	51,3
17	564	56,4
18	554	69,3
19	552	138,0
20	520	173,3



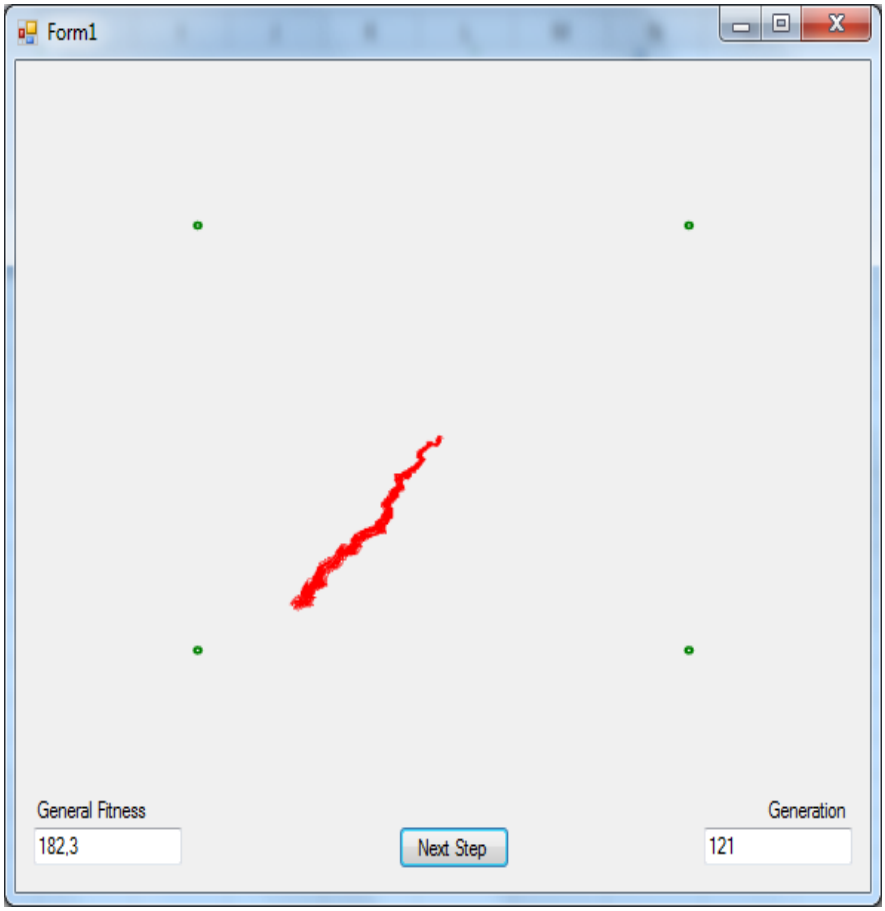
№	Fitness	Shared Fitness
1	384	19,2
2	384	19,2
3	386	19,3
4	386	19,3
5	388	19,4
6	388	19,4
7	388	19,4
8	388	19,4
9	388	19,4
10	390	19,5
11	392	19,6
12	392	19,6
13	392	19,6
14	394	19,7
15	394	19,7
16	394	19,7
17	394	19,7
18	396	19,8
19	396	19,8
20	404	20,2



№	Fitness	Shared Fitness
1	276	13,8
2	278	13,9
3	280	14
4	280	14
5	280	14
6	282	14,1
7	282	14,1
8	282	14,1
9	284	14,2
10	284	14,2
11	284	14,2
12	284	14,2
13	284	14,2
14	286	14,3
15	286	14,3
16	286	14,3
17	288	14,4
18	288	14,4
19	288	14,4
20	288	14,4



№	Fitness	Shared Fitness
1	176	8,8
2	176	8,8
3	180	9
4	180	9
5	180	9
6	180	9
7	180	9
8	180	9
9	182	9,1
10	182	9,1
11	182	9,1
12	182	9,1
13	182	9,1
14	184	9,2
15	184	9,2
16	184	9,2
17	184	9,2
18	186	9,3
19	188	9,4
20	194	9,7



№	Fitness	Shared Fitness
1	0	0
2	0	0
3	2	0,1
4	2	0,1
5	2	0,1
6	4	0,2
7	4	0,2
8	4	0,2
9	4	0,2
10	4	0,2
11	6	0,3
12	6	0,3
13	6	0,3
14	6	0,3
15	8	0,4
16	8	0,4
17	8	0,4
18	10	0,5
19	10	0,5
20	12	0,6