

Source code

```

class Dimension
{
    public double[,] _10DMatrix; // Points of 10D graphic
    public double[,] _10DMatrixDistances; // Distances between points of 10D graphic
    public double[,] _2DMatrix; // Points of 2D graphic
    public double[,] _2DMatrixDistances; // Distances between points of 2D graphic
    public List<double[,]> List_2DMatrix; // Generation
    public double[,] fitnessMatrix; // Difference between 10D and 2D distances
    public List<double> GenerationFitnessList; // Fitnesses of whole generation
    public List<double> MinGenerationFitness; // Minimum fitness in every generation
    public List<double> AvgGenerationFitness; // Average fitness in every
generation
    const int _10D = 10;
    const int _2D = 2;
    public readonly int numberOfPoints = 30;
    const int ChromosomesNumber = 100;

    #region initialization and finding distance matrix

    public void GenerateMatrix()
    {
        Random rnd = new Random();
        _10DMatrix = new double[numberOfPoints, _10D];
        for(int i = 0; i < numberOfPoints; i++)
            for(int j = 0; j < _10D; j++)
                _10DMatrix[i, j] = rnd.Next(0, 2001);
        _10DMatrix = NormalizeRandom(_10DMatrix, _10D);
        _10DMatrixDistances = new double[numberOfPoints, numberOfPoints];
        for (int i = 0; i < numberOfPoints; i++)
            for (int j = 0; j < numberOfPoints; j++)
                _10DMatrixDistances[i, j] = FindDistanceND(i, j, _10D, _10DMatrix);
        _10DMatrixDistances = NormalizeDistance(_10DMatrixDistances);
        GenerationFitnessList = new List<double>();
        List_2DMatrix = new List<double[,]>();
        int iter = 0;
        while (iter < 100)
        {
            Generate2DMatrix();
            FindFitness();
            List_2DMatrix.Add(_2DMatrix);
            iter++;
        }
        MinGenerationFitness = new List<double>();
        AvgGenerationFitness = new List<double>();
        MinGenerationFitness.Add(GenerationFitnessList.Min());
        AvgGenerationFitness.Add(GenerationFitnessList.Average());
    }

    void Generate2DMatrix()
    {
        Random rnd = new Random();
        _2DMatrix = new double[numberOfPoints, _2D];
        for (int i = 0; i < numberOfPoints; i++)
            for (int j = 0; j < _2D; j++)
                _2DMatrix[i, j] = rnd.Next(0, 2001);
        _2DMatrix = NormalizeRandom(_2DMatrix, _2D);
        _2DMatrixDistances = new double[numberOfPoints, numberOfPoints];
        for (int i = 0; i < numberOfPoints; i++)
            for (int j = 0; j < numberOfPoints; j++)
                _2DMatrixDistances[i, j] = FindDistanceND(i, j, _2D, _2DMatrix);
        _2DMatrixDistances = NormalizeDistance(_2DMatrixDistances);
    }
}

```

```

    }

    int FindDistanceND(int i , int j, int size, double[,] _10DMatr)
    {
        if (i == j) return 0;
        int num = 0;
        for (int k = 0; k < size; k++)
            num += (int)Math.Pow(_10DMatr[i, k] - _10DMatr[j, k], 2);
        return (int)Math.Sqrt(num);
    }

    double[,] NormalizeDistance(double[,] mat)
    {
        double max = mat.Cast<double>().Max();
        for(int i = 0; i < numberOfPoints; i++)
            for(int j = 0; j < numberOfPoints; j++)
                mat[i, j] /= max;
        return mat;
    }

    double[,] NormalizeRandom(double[,] mat, int size)
    {
        for(int i = 0; i < numberOfPoints; i++)
            for(int j = 0; j < size; j++)
            {
                mat[i, j] /= 1000;
                if (mat[i, j] == 1) mat[i, j] = 0;
                else if (mat[i, j] < 1) mat[i, j] = -mat[i, j];
                else mat[i, j] -= 1;
            }
        return mat;
    }

    public void FindFitness()
    {
        fitnessMatrix = new double[numberOfPoints, numberOfPoints];
        double fitness = 0;
        for (int i = 0; i < numberOfPoints; i++)
            for (int j = 0; j < numberOfPoints; j++)
                fitnessMatrix[i, j] = _10DMatrixDistances[i, j] -
                _2DMatrixDistances[i, j];
        for (int i = 0; i < numberOfPoints; i++)
            for (int j = 0; j < numberOfPoints; j++)
                fitness += fitnessMatrix[i, j];
        GenerationFitnessList.Add(fitness);
    }

    #endregion

    public void MainCicle()
    {
        SortPopulation();
    }

    #region Sorting

    public void SortPopulation()
    {
        int[] keys = new int[ChromosomesNumber];
        for (int i = 0; i < ChromosomesNumber; i++) keys[i] = i;
        Array.Sort(GenerationFitnessList.ToArray(), keys);
        List<double[,]> sorted2DMatrix = new List<double[,]>();
        for (int i = 0; i < ChromosomesNumber; i++)
            sorted2DMatrix.Add(FindKMatrix(keys[i]));
        List_2DMatrix = null;
    }

```

```

        List_2DMatrix = sorted2DMatrix;
    }

    public double[,] FindKMatrix(int k)
    {
        for (int i = 0; i < numberOfPoints; i++)
            if (i == k) return List_2DMatrix[i];
        return null;
    }

    #endregion

    public void UniformCrossover()
    {
    }

    public bool Uslovie()
    {
        if (MinGenerationFitness.Last() == 0) return false;
        return true;
    }
}

class Program
{
    static void Main(string[] args)
    {
        Dimension obj = new Dimension();
        obj.GenerateMatrix();
        for (int i = 0; i < obj.numberOfPoints; i++)
        {
            for (int j = 0; j < obj.numberOfPoints; j++)
                Console.Write("{0} ", obj._1DMatrixDistances[i, j]);
            Console.WriteLine("\n");
        }
    }
}

```