This program was designed for study purposes so no self-serving or trading intentions are not provided and highly disapproved

This program is designated to solve TSP problem by means of elastic net method.

Short description

This program finds out one of the good (not nececcarily the best) ways between set points(e.g. cities)

There's an option to change some values i.e. learning coefficients,

total iterations quantity, screen refresh rate etc.

Short instrucrion

Nodes per city - amount of normalizing ring nodes dedicated for single point(city)

The normalizing ring generated would contain (amount of cities \* nodes per city) nodes

Max iterations - maximal amount of iterations

(if the limit reached, program will terminate regardless if algorithm either found the solution or not)

Max Radius - number of nodes on each side of point is moved forward to the city which are decided to follow that one in gaussian progression

Screen Refresh Rate - [times per second];it's believed there is no need to explain more what this parameter does

Learning Rate Decrease speed - the higher the number the slower the learning rate will decrease

KEEP IN MIND!!!! If it's high value would be combined with a large number of iterations, an error can occur and program will terminate within a few hundred of iterations

Max Learning Rate - learning rate starts from it's value

ComboBox - there's an opportunity to choose where the normalizing ring will be drawn from

Write downn intermediate data to console - it prints all the data(step by step) if chosen

PAY ATTENTION!!! There is no sense to choose this parameter if the program was ran not from .bat 'cause the visual console wasn't opened and the program just would be slowed down greatly writing the data to the virtual one

Write to file - writes the result of current program into the file(see the 'output' folder) numerated from zero. Each next run with file writing will generate new file so there is no risk for the previous data to be lost

Select file with data - this method helps to choose file with point's(city's) data on which the map generation will be based

You should know that even not any .tsp file could run. It's because some of them were made in incomprehensible for this program author way so make sure data is stored in way any of example data stored

User wanting create their own map should use data.txt as a layout

Inner data format for .tsp files can be seen placing all the file's contents to Microsoft Word Document

Program working principle

The algorithm execution itself can be broken on three essential blocks:

**I.<Initialization of starting parameters>;**

**II.<Calculating the result in iterative way>;**

**III.<Definition if the algorithm has got the solution on the current iteration>;**

which can be represented as such an algorithm:

**I.<Initialization of starting parameters>;**

1.Define the smallest and the biggest x’s and y’s from the cities array;

2. a) For map to generate set the smallest x value as 0 on X axis and the biggest x value as 1;

b) set the smallest y value as 0 on Y axis and the biggest as 1

3. Set variables Xrange and Yrange, where

Xrange = Xmax – Xmin;

Yrange = Ymax – Ymin;

4.For each city (with (x;y) coordinates) set scaled analogues accordingly to this formula: thisCityNormalizedX =

thisCityNormalizedY =

5. Set the center to start drawing normalizing ring from:

a) if a parameter ‘start from cities mass center’ was selected, calculate

XesSum =;

YsSum = ;

Xcenter = , Ycenter = ;

b) if it wasn’t, set: Xcenter = 0.5, Ycenter = 0.5;

6.Create new array for normalizing ring node elements named nodesArray and set array size:

arraySize = citiesQuantity \* Nodes per Cities;

7. step =

8.radius = 0.5

9. for(int i = 0; i < arraySize; i++){

nodesArray[i]:

}

in other words it means the broken line of nodes is set in a circle-like way;

**II.<Calculating the result in iterative way>;**

if (!(not)(final conditions are done){

}

else{

1. print out currentIteration;

2. print out currentLearningRate;

3. print out currentNodesRingLength, which is calculated by formula set below:

currentNodesRingLength =

= ;

4. print out currentRadius;

5. print out normalizedRingLength and trueRingLength, which are calculated by formulas:

1) sort normalizedCities and trueCities accordingly to tiedUpNode orders

if (cityA.tiedUpNode.index > cityB.tiedUpNode.index) then

cityB.index = cityA.index, cityA.index = cityB.index;

else do nothing;

2) normalizedRingLength =

=;

3) trueRingLength = ;

6. print out executionTime , which equals to difference endTime – startTime;

}

1.

**III.<Definition if the algorithm has got the solution on the current iteration>;**

set finalConditionsAreDone = true, if at least one of three conditions became true:

{

1. currentIteration ≥ maximalIteration;
2. currentRadius ≤ minimumAvailableRadius;
3. ifSolutionCloseEnough is true;(

ifSolutionCloseEnough = true, if distances bettwen each City and the nearest to it node less than minimumLearningRate;

else ifSolutionIsCloseEnough = false;)

}

else finalConditionsAreDone = false;