**Pei Shi 1379396**

namespace TSPForm

{

public partial class FormTSP : Form

{

private int Cities;

private double[] xMinMax;

private double[] yMinMax;

private double[,] xy;

private double a;

private static double b;

private int[] goodPathIndex;

private double[,] CityDistance;

private int[] CityTabu;

private double shortesPathLenth = double.PositiveInfinity;

private double BeforeShortPathLegth = double.PositiveInfinity;

int[,] CityAntHaveTrave;

double[,] PheromoneTau;

double[,] etaCij;

double[,] deltaTau;

int[,] CitiesTabu;

int[,] CityNoTabu;

double[] mAntShortestDistance;

int AntCount;

int NcMax = 3000;

double Q = 100;

double alpha = 2.0;

double beta = 7.0;

double rho = 0.5;

int iHaveNoChange = 50;

int Nc = 0; //the current number of iterations

double PhenomenonZero = 1e-5;

double PhenomenonRangeMax =30;

public FormTSP()

{

InitializeComponent();

}

/// <summary>

///

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void readDataToolStripMenuItem\_Click(object sender, EventArgs e)

{

labelCTLength.Text = "";

labelDimension.Text = "";

labelName.Text = "";

labelOPTLength.Text = "";

labelImprovement.Text = "";

labelProgress.Text = "";

OpenFileDialog myFileDialog = new OpenFileDialog();

myFileDialog.InitialDirectory = Application.StartupPath + "\\Data";

xMinMax = new double[2] { double.MaxValue, 0 };

yMinMax = new double[2] { double.MaxValue, 0 };

if (myFileDialog.ShowDialog() == DialogResult.OK)

{

string strPath = myFileDialog.FileName;

StreamReader streamReader = new StreamReader(strPath);

string stringData;

string[] stringArray;

string dataFlag = "no";

int i = 0;

while ((stringData = streamReader.ReadLine()) != "EOF")

{

if (stringData.Contains("NAME"))

{

stringArray = stringData.Split(':');

labelName.Text = stringArray[1];

}

if (stringData.Contains("Optimum"))

{

int iLenthStartIndex;

iLenthStartIndex = stringData.IndexOf("is") + 2;

labelOPTLength.Text = stringData.Substring(iLenthStartIndex);

}

if (stringData.Contains("DIMENSION"))

{

stringArray = stringData.Split(':');

labelDimension.Text = stringArray[1] + " cities";

Cities = int.Parse(stringArray[1].Trim());

xy = new double[Cities, 2];

goodPathIndex = new int[Cities];

CityDistance = new double[Cities, Cities];

CityTabu = new int[Cities];

setBarValue(0, Cities);

}

if (dataFlag == "yes")

{

stringArray = stringData.Split(' ');

xy[i, 0] = double.Parse(stringArray[1]);

xy[i, 1] = double.Parse(stringArray[2]);

if (xMinMax[0] > xy[i, 0])

{

xMinMax[0] = xy[i, 0];

}

if (xMinMax[1] < xy[i, 0])

{

xMinMax[1] = xy[i, 0];

}

if (yMinMax[0] > xy[i, 1])

{

yMinMax[0] = xy[i, 1];

}

if (yMinMax[1] < xy[i, 1])

{

yMinMax[1] = xy[i, 1];

}

i++;

UpadateProgressBar(i);

}

if (stringData.Contains("NODE\_COORD\_SECTION"))

{

dataFlag = "yes";

}

}

streamReader.Close();

DrawCityPoint();

CalculateDistance();

}

}

/// <summary>

///

/// </summary>

private void CalculateDistance()

{

setBarValue(0, Cities);

for (int j = 0; j < Cities; j++)

{

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

{

CityDistance[j, k] = Math.Sqrt((xy[j, 0] - xy[k, 0]) \* (xy[j, 0] - xy[k, 0]) + (xy[j, 1] - xy[k, 1]) \* (xy[j, 1] - xy[k, 1]));

}

UpadateProgressBar(j + 1);

}

}

/// <summary>

///

/// </summary>

/// <param name="iBarValue"></param>

public void UpadateProgressBar(int iBarValue)

{

if (progressBarTSP.InvokeRequired)

{

Action<int> actionUpdateProgressBar = delegate(int iTemp)

{

progressBarTSP.Value = iTemp;

};

progressBarTSP.Invoke(actionUpdateProgressBar, iBarValue > progressBarTSP.Maximum ? progressBarTSP.Maximum : iBarValue);

}

else

{

progressBarTSP.Value = iBarValue;

}

}

/// <summary>

///

/// </summary>

/// <param name="minValue"></param>

/// <param name="maxValue"></param>

public void setBarValue(int minValue, int maxValue)

{

if (progressBarTSP.InvokeRequired)

{

Action<int, int> actionSetBarValue = delegate(int minTemp, int maxTemp)

{

progressBarTSP.Minimum = minTemp;

progressBarTSP.Maximum = maxTemp;

};

progressBarTSP.Invoke(actionSetBarValue, minValue, maxValue);

}

else

{

progressBarTSP.Minimum = minValue;

progressBarTSP.Maximum = maxValue;

}

}

/// <summary>

///

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void FormTSP\_Load(object sender, EventArgs e)

{

labelCTLength.Text = "";

labelDimension.Text = "";

labelName.Text = "";

labelOPTLength.Text = "";

labelImprovement.Text = "";

labelProgress.Text = "";

}

/// <summary>

///

/// </summary>

private void DrawCityPoint()

{

setBarValue(0, Cities);

if (pictureBoxCity.InvokeRequired)

{

Action actionDrawCityPoint = delegate()

{

double fPicboxWidth = pictureBoxCity.Width;

double fPicboxheight = pictureBoxCity.Height;

double PicAxisX, PicAxisY;

PicAxisX = (double)(fPicboxWidth - 30) / (xMinMax[1] - xMinMax[0]);

PicAxisY = (double)(fPicboxheight - 30) / (yMinMax[1] - yMinMax[0]);

SolidBrush RedBrush = new SolidBrush(Color.Red);

pictureBoxCity.Image = null;

Image imageTSP = new Bitmap(pictureBoxCity.Width, pictureBoxCity.Height);

Graphics GraphicsTSP = Graphics.FromImage(imageTSP);

GraphicsTSP.Clear(Color.White);

double cityPointX;

double cityPointY;

for (int j = 0; j < Cities; j++)

{

cityPointX = (xy[j, 0] - xMinMax[0]) \* PicAxisX;

cityPointY = (xy[j, 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.FillEllipse(RedBrush, new RectangleF((float)cityPointX, (float)cityPointY, 3, 3));

UpadateProgressBar(j + 1);

}

pictureBoxCity.Image = imageTSP;

};

pictureBoxCity.Invoke(actionDrawCityPoint);

}

else

{

double fPicboxWidth = pictureBoxCity.Width;

double fPicboxheight = pictureBoxCity.Height;

double PicAxisX, PicAxisY;

PicAxisX = (double)(fPicboxWidth - 30) / (xMinMax[1] - xMinMax[0]);

PicAxisY = (double)(fPicboxheight - 30) / (yMinMax[1] - yMinMax[0]);

SolidBrush RedBrush = new SolidBrush(Color.Red);

pictureBoxCity.Image = null;

Image imageTSP = new Bitmap(pictureBoxCity.Width, pictureBoxCity.Height);

Graphics GraphicsTSP = Graphics.FromImage(imageTSP);

GraphicsTSP.Clear(Color.White);

double cityPointX;

double cityPointY;

for (int j = 0; j < Cities; j++)

{

cityPointX = (xy[j, 0] - xMinMax[0]) \* PicAxisX;

cityPointY = (xy[j, 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.FillEllipse(RedBrush, new RectangleF((float)cityPointX, (float)cityPointY, 3, 3));

UpadateProgressBar(j + 1);

}

pictureBoxCity.Image = imageTSP;

}

}

/// <summary>

///

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void exitToolStripMenuItem\_Click(object sender, EventArgs e)

{

Application.Exit();

}

/// <summary>

///

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void greedySolverToolStripMenuItem\_Click(object sender, EventArgs e)

{

labelProgress.Text = "Greedy algorithm progress";

shortesPathLenth = double.PositiveInfinity;

BeforeShortPathLegth = double.PositiveInfinity;

Thread threadGreedy = new Thread(new ThreadStart(GreedyAlgorithm));

threadGreedy.IsBackground = true;

threadGreedy.Start();

}

/// <summary>

///

/// </summary>

private void GreedyAlgorithm()

{

int[] GreedyhaveTrave = new int[Cities];

double cityShortestDistance = double.PositiveInfinity;

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

{

goodPathIndex[i] = 0;

CityTabu[i] = 0;

GreedyhaveTrave[i] = -1;

}

GreedyhaveTrave[0] = 0;

CityTabu[GreedyhaveTrave[0]] = 1;

int minCityIndex = GreedyhaveTrave[0];

setBarValue(1, Cities - 1);

for (int i = 1; i < Cities; i++)

{

cityShortestDistance = double.PositiveInfinity;

for (int j = 0; j < Cities; j++)

{

if (CityDistance[GreedyhaveTrave[i - 1], j] < cityShortestDistance && CityTabu[j] == 0)

{

minCityIndex = j;

cityShortestDistance = CityDistance[GreedyhaveTrave[i - 1], j];

}

}

UpadateProgressBar(i);

CityTabu[minCityIndex] = 1;

GreedyhaveTrave[i] = minCityIndex;

WriteProgressInfo(i, cityShortestDistance);

}

shortesPathLenth = 0;

for (int k = 0; k < Cities - 1; k++)

{

shortesPathLenth += CityDistance[GreedyhaveTrave[k], GreedyhaveTrave[k + 1]];

}

shortesPathLenth += CityDistance[GreedyhaveTrave[Cities - 1], GreedyhaveTrave[0]];

WriteProgressInfo(Cities, shortesPathLenth);

goodPathIndex = GreedyhaveTrave;

DrawShortestPath();

}

/// <summary>

///

/// </summary>

/// <param name="iterationCount"></param>

/// <param name="minimumLength"></param>

public void WriteProgressInfo(int iterationCount, double minimumLength)

{

double fImprovement = 0;

if (labelImprovement.InvokeRequired)

{

Action<int, double> actionIteration = delegate(int iterationCountTemp, double minimumLengthTemp)

{

fImprovement = BeforeShortPathLegth == double.PositiveInfinity ? 0 : BeforeShortPathLegth - minimumLengthTemp;

labelImprovement.Text = "(t=" + iterationCountTemp.ToString() + ")," + fImprovement.ToString("F0");

};

labelImprovement.Invoke(actionIteration, iterationCount, minimumLength);

}

else

{

fImprovement = BeforeShortPathLegth == double.PositiveInfinity ? 0 : BeforeShortPathLegth - minimumLength;

labelImprovement.Text = "(t=" + iterationCount.ToString() + ")," + fImprovement.ToString("F0");

}

if (labelCTLength.InvokeRequired)

{

Action<double> actionLength = delegate(double shortLengh)

{

labelCTLength.Text = shortLengh == 0 ? "" : shortLengh.ToString("F0");

};

labelCTLength.Invoke(actionLength, minimumLength);

}

else

{

labelCTLength.Text = minimumLength == 0 ? "" : minimumLength.ToString("F0");

}

}

/// <summary>

///

/// </summary>

public void DrawShortestPath()

{

if (pictureBoxCity.InvokeRequired)

{

Action actionDraw = delegate()

{

double fPicboxWidth = pictureBoxCity.Width;

double fPicboxheight = pictureBoxCity.Height;

double PicAxisX, PicAxisY;

PicAxisX = (double)(fPicboxWidth - 30) / (xMinMax[1] - xMinMax[0]);

PicAxisY = (double)(fPicboxheight - 30) / (yMinMax[1] - yMinMax[0]);

SolidBrush RedBrush = new SolidBrush(Color.Red);

pictureBoxCity.Image = null;

Image imageTSP = new Bitmap(pictureBoxCity.Width, pictureBoxCity.Height);

Graphics GraphicsTSP = Graphics.FromImage(imageTSP);

GraphicsTSP.Clear(Color.White);

double cityPointX;

double cityPointY;

setBarValue(0, Cities - 1);

Font myFont1 = new Font("Hacttenschweiler", 7);

for (int j = 0; j < Cities; j++)

{

cityPointX = (xy[j, 0] - xMinMax[0]) \* PicAxisX;

cityPointY = (xy[j, 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.FillEllipse(RedBrush, new RectangleF((float)cityPointX, (float)cityPointY, 3, 3));

UpadateProgressBar(j);

}

double FirstPointX, FirstPointY, SecondPointX, SecondPointY;

Pen penLine = new Pen(Color.Black, 1);

setBarValue(1, Cities - 1);

for (int i = 1; i < Cities; i++)

{

FirstPointX = (xy[goodPathIndex[i - 1], 0] - xMinMax[0]) \* PicAxisX;

FirstPointY = (xy[goodPathIndex[i - 1], 1] - yMinMax[0]) \* PicAxisY;

SecondPointX = (xy[goodPathIndex[i], 0] - xMinMax[0]) \* PicAxisX;

SecondPointY = (xy[goodPathIndex[i], 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.DrawLine(penLine, new PointF((float)FirstPointX, (float)FirstPointY), new PointF((float)SecondPointX, (float)SecondPointY));

UpadateProgressBar(i);

}

FirstPointX = (xy[goodPathIndex[0], 0] - xMinMax[0]) \* PicAxisX;

FirstPointY = (xy[goodPathIndex[0], 1] - yMinMax[0]) \* PicAxisY;

SecondPointX = (xy[goodPathIndex[Cities - 1], 0] - xMinMax[0]) \* PicAxisX;

SecondPointY = (xy[goodPathIndex[Cities - 1], 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.DrawLine(penLine, new PointF((float)FirstPointX, (float)FirstPointY), new PointF((float)SecondPointX, (float)SecondPointY));

pictureBoxCity.Image = imageTSP;

};

pictureBoxCity.Invoke(actionDraw);

}

else

{

double fPicboxWidth = pictureBoxCity.Width;

double fPicboxheight = pictureBoxCity.Height;

double PicAxisX, PicAxisY;

PicAxisX = (double)(fPicboxWidth - 30) / (xMinMax[1] - xMinMax[0]);

PicAxisY = (double)(fPicboxheight - 30) / (yMinMax[1] - yMinMax[0]);

SolidBrush RedBrush = new SolidBrush(Color.Red);

pictureBoxCity.Image = null;

Image imageTSP = new Bitmap(pictureBoxCity.Width, pictureBoxCity.Height);

Graphics GraphicsTSP = Graphics.FromImage(imageTSP);

GraphicsTSP.Clear(Color.White);

double cityPointX;

double cityPointY;

setBarValue(0, Cities - 1);

for (int j = 0; j < Cities; j++)

{

cityPointX = (xy[j, 0] - xMinMax[0]) \* PicAxisX;

cityPointY = (xy[j, 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.FillEllipse(RedBrush, new RectangleF((float)cityPointX, (float)cityPointY, 3, 3));

UpadateProgressBar(j);

}

double FirstPointX, FirstPointY, SecondPointX, SecondPointY;

Pen penLine = new Pen(Color.Black, 1);

setBarValue(1, Cities - 1);

for (int i = 1; i < Cities; i++)

{

FirstPointX = (xy[goodPathIndex[i - 1], 0] - xMinMax[0]) \* PicAxisX;

FirstPointY = (xy[goodPathIndex[i - 1], 1] - yMinMax[0]) \* PicAxisY;

SecondPointX = (xy[goodPathIndex[i], 0] - xMinMax[0]) \* PicAxisX;

SecondPointY = (xy[goodPathIndex[i], 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.DrawLine(penLine, new PointF((float)FirstPointX, (float)FirstPointY), new PointF((float)SecondPointX, (float)SecondPointY));

UpadateProgressBar(i);

}

FirstPointX = (xy[goodPathIndex[0], 0] - xMinMax[0]) \* PicAxisX;

FirstPointY = (xy[goodPathIndex[0], 1] - yMinMax[0]) \* PicAxisY;

SecondPointX = (xy[goodPathIndex[Cities - 1], 0] - xMinMax[0]) \* PicAxisX;

SecondPointY = (xy[goodPathIndex[Cities - 1], 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.DrawLine(penLine, new PointF((float)FirstPointX, (float)FirstPointY), new PointF((float)SecondPointX, (float)SecondPointY));

pictureBoxCity.Image = imageTSP;

}

}

private void aCOSolverToolStripMenuItem\_Click(object sender, EventArgs e)

{

labelProgress.Text = "ACO algorithm progress";

shortesPathLenth = double.PositiveInfinity;

BeforeShortPathLegth = double.PositiveInfinity;

Nc = 0;

Thread threadACO = new Thread(new ThreadStart(ACOAlgrorithm));

threadACO.IsBackground = true;

threadACO.Start();

}

/// <summary>

///

/// </summary>

private void ACOAlgrorithm()

{

AntCount = Cities;

PheromoneTau = new double[Cities, Cities];

etaCij = new double[Cities, Cities];

deltaTau = new double[Cities, Cities];

CitiesTabu = new int[AntCount, Cities];

CityNoTabu = new int[AntCount, Cities];

CityAntHaveTrave = new int[AntCount, Cities];

mAntShortestDistance = new double[AntCount];

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

{

for (int j = 0; j < Cities; j++)

{

PheromoneTau[i, j] = 1.0;

if (i != j)

{

etaCij[i, j] = 1.0 / CityDistance[i, j];

}

deltaTau[i, j] = 0.0;

}

}

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

{

for (int j = 0; j < Cities; j++)

{

CityAntHaveTrave[i, j] = -1;

CitiesTabu[i, j] = 0;

}

CityAntHaveTrave[i, 0] = 0;

CitiesTabu[i, 0] = 1;

}

for (int iterationCount = 1; iterationCount <= NcMax; iterationCount++)

{

startAntMove();

int minLenthAntNo = 0;

BeforeShortPathLegth = shortesPathLenth;

setBarValue(0, AntCount);

for (int antNo = 0; antNo < AntCount; antNo++)

{

mAntShortestDistance[antNo] = calculateMinDistance(antNo);

if (mAntShortestDistance[antNo] < shortesPathLenth)

{

shortesPathLenth = mAntShortestDistance[antNo];

minLenthAntNo = antNo;

}

UpadateProgressBar(antNo + 1);

}

setBarValue(0, Cities);

for (int cityNO = 0; cityNO < Cities; cityNO++)

{

goodPathIndex[cityNO] = CityAntHaveTrave[minLenthAntNo, cityNO];

UpadateProgressBar(cityNO + 1);

}

UpdatePathIncrementPheromone();

setBarValue(0, Cities);

for (int icityNo = 0; icityNo < Cities; icityNo++)

{

for (int jcityNo = 0; jcityNo < Cities; jcityNo++)

{

PheromoneTau[icityNo, jcityNo] = (1 - rho) \* PheromoneTau[icityNo, jcityNo] + deltaTau[icityNo, jcityNo];

if (PheromoneTau[icityNo, jcityNo] < PhenomenonZero)

PheromoneTau[icityNo, jcityNo] = PhenomenonZero;

if (PheromoneTau[icityNo, jcityNo] > PhenomenonRangeMax)

PheromoneTau[icityNo, jcityNo] = PhenomenonRangeMax;

}

UpadateProgressBar(icityNo + 1);

}

setBarValue(0, AntCount);

for (int antNo = 0; antNo < AntCount; antNo++)

{

for (int cityNo = 0; cityNo < Cities; cityNo++)

{

CitiesTabu[antNo, cityNo] = 0;

if (cityNo > 0)

{

CityAntHaveTrave[antNo, cityNo] = -1;

}

}

CitiesTabu[antNo, CityAntHaveTrave[antNo, 0]] = 1;

UpadateProgressBar(antNo + 1);

}

WriteProgressInfo(iterationCount, shortesPathLenth);

DrawShortestPath();

}

}

/// <summary>

///

/// </summary>

private void startAntMove()

{

double antPij = 0;

double PijSum = 0;

double randomPij = 0;

Random randData = new Random((int)DateTime.Now.Ticks & 0x0000FFFF);

for (int haveTraveCity = 1; haveTraveCity < Cities; haveTraveCity++)

{

setBarValue(0, AntCount);

for (int iAnt = 0; iAnt < AntCount; iAnt++)

{

antPij = 0.0;

PijSum = 0.0;

randomPij = randData.Next(3000) / 3000.0;

for (int iCity = 0; iCity < Cities; iCity++)

{

if (CitiesTabu[iAnt, iCity] == 0)

{

try

{

PijSum += Math.Pow(PheromoneTau[CityAntHaveTrave[iAnt, haveTraveCity - 1], iCity], alpha) \* Math.Pow(etaCij[CityAntHaveTrave[iAnt, haveTraveCity - 1], iCity], beta);

}

catch (System.Exception ex)

{

continue;

}

}

}

for (int iCity = 0; iCity < Cities; iCity++)

{

if (CitiesTabu[iAnt, iCity] == 0)

{

try

{

antPij += Math.Pow(PheromoneTau[CityAntHaveTrave[iAnt, haveTraveCity - 1], iCity], alpha) \* Math.Pow(etaCij[CityAntHaveTrave[iAnt, haveTraveCity - 1], iCity], beta) / PijSum;

if (antPij > randomPij)

{

CitiesTabu[iAnt, iCity] = 1;

CityAntHaveTrave[iAnt, haveTraveCity] = iCity;

break;

}

}

catch (System.Exception ex)

{

continue;

}

}

}

UpadateProgressBar(iAnt + 1);

}

}

}

/// <summary>

///

/// </summary>

private void UpdatePathIncrementPheromone()

{

setBarValue(0, AntCount);

for (int antNo = 0; antNo < AntCount; antNo++)

{

try

{

for (int cityNo = 0; cityNo < Cities - 1; cityNo++)

{

deltaTau[CityAntHaveTrave[antNo, cityNo], CityAntHaveTrave[antNo, cityNo + 1]] += 100 / mAntShortestDistance[antNo];

}

deltaTau[CityAntHaveTrave[antNo, Cities - 1], CityAntHaveTrave[antNo, 0]] += 100 / mAntShortestDistance[antNo];

}

catch (System.Exception ex)

{

continue;

}

UpadateProgressBar(antNo + 1);

}

}

/// <summary>

///

/// </summary>

/// <param name="antNo"></param>

/// <returns></returns>

private double calculateMinDistance(int antNo)

{

double currentMinDis = 0;

setBarValue(0, Cities - 1);

for (int cityNO = 0; cityNO < Cities - 1; cityNO++)

{

UpadateProgressBar(cityNO + 1);

currentMinDis += CityDistance[CityAntHaveTrave[antNo, cityNO], CityAntHaveTrave[antNo, cityNO + 1]];

}

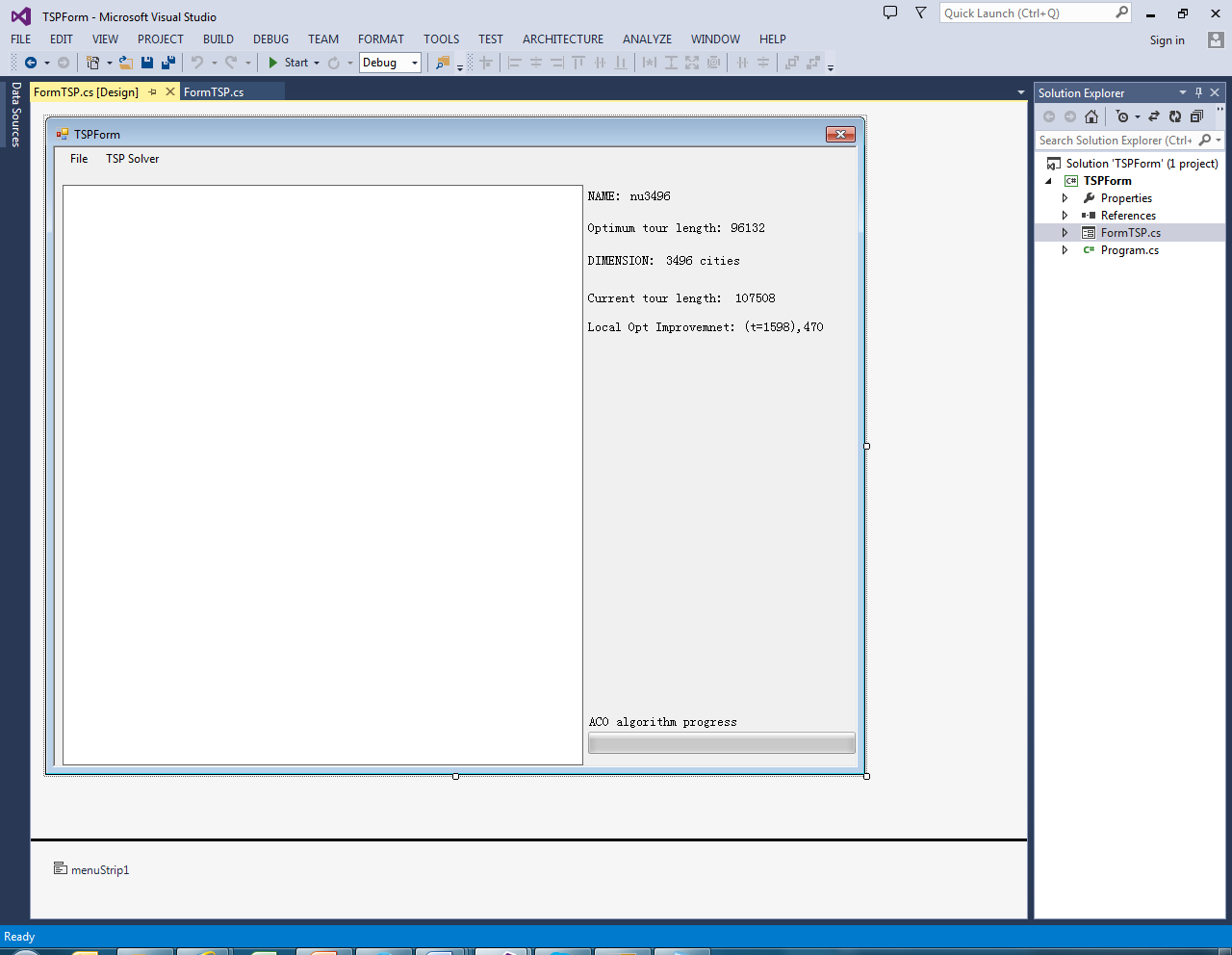
currentMinDis += CityDistance[CityAntHaveTrave[antNo, Cities - 1], CityAntHaveTrave[antNo, 0]];

return currentMinDis;

}

}

}



**Guanshen Yan 1481476**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Windows.Forms;

namespace TSPForm

{

public partial class FormTSP : Form

{

private int City;

private double[] xMinMax;

private double[] yMinMax;

private double[,] xy;

private int[] m; // the index of the good path

private double[,] d; //distance between each city

private int[] CityList;

private double optl = double.PositiveInfinity;

private double boptl = double.PositiveInfinity;

int[,] VisitedCity; //store the city which has been visited by ant

int[,] CitiesTabu;

int[,] CityNoTabu;

int n; //ant number

int TMax = 3000; //the maximum iteration number

double[,] Pheromone; //the amount of pheromone

double[,] eta; // reciprocal of the length of arc(i,j)

double[,] delta;

double[] dk; //the shortest length of the tour which finished by ant k

double alpha = 2.0;

double beta = 7.0; //the relative influence of the arc length over the previously accumulated pheromone deposit

double r = 0.5; //evoraption rate

double PheromoneMin = 1e-5; // the range of the density of pheromone

double PheromoneMax =30;

public FormTSP()

{

InitializeComponent();

}

////////////////////////////////////////////////////////Greedy/////////////////////////////////////////////////////////////////////

private void Greedy()

{

int[] travelledcity = new int[City]; // vector of visited city

double q = double.PositiveInfinity; //the shortest distance between each city

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

{

m[i] = 0;

CityList[i] = 0;

travelledcity[i] = -1;

}

travelledcity[0] = 0;

CityList[travelledcity[0]] = 1;

int minIndex = travelledcity[0];

Value(1, City - 1);

for (int i = 1; i < City; i++)

{

q = double.PositiveInfinity;

for (int j = 0; j < City; j++)

{

if (d[travelledcity[i - 1], j] < q && CityList[j] == 0)

{

minIndex = j;

q = d[travelledcity[i - 1], j];

}

}

ProgressBar(i);

CityList[minIndex] = 1;

travelledcity[i] = minIndex;

displayinfo(i, q);

}

optl = 0;

for (int k = 0; k < City - 1; k++)

{

optl += d[travelledcity[k], travelledcity[k + 1]];

}

optl += d[travelledcity[City - 1], travelledcity[0]];

displayinfo(City, optl);

m = travelledcity;

if (pictureBox1.InvokeRequired)

{

Action actionDraw = delegate()

{

double W = pictureBox1.Width;

double H = pictureBox1.Height;

double PicX, PicY;

PicX = (double)(W - 30) / (xMinMax[1] - xMinMax[0]);

PicY = (double)(H - 30) / (yMinMax[1] - yMinMax[0]);

SolidBrush RedBrush = new SolidBrush(Color.Red);

pictureBox1.Image = null;

Image imageTSP = new Bitmap(pictureBox1.Width, pictureBox1.Height);

Graphics GraphicsTSP = Graphics.FromImage(imageTSP);

GraphicsTSP.Clear(Color.White);

double cityX;

double cityY;

Value(0, City - 1);

Font myFont1 = new Font("Hacttenschweiler", 7);

for (int j = 0; j < City; j++)

{

cityX = (xy[j, 0] - xMinMax[0]) \* PicX;

cityY = (xy[j, 1] - yMinMax[0]) \* PicY;

GraphicsTSP.FillEllipse(RedBrush, new RectangleF((float)cityX, (float)cityY, 3, 3));

ProgressBar(j);

}

double X1, Y1, X2, Y2;

Pen penLine = new Pen(Color.Black, 1);

Value(1, City - 1);

for (int i = 1; i < City; i++)

{

X1 = (xy[m[i - 1], 0] - xMinMax[0]) \* PicX;

Y1 = (xy[m[i - 1], 1] - yMinMax[0]) \* PicY;

X2 = (xy[m[i], 0] - xMinMax[0]) \* PicX;

Y2 = (xy[m[i], 1] - yMinMax[0]) \* PicY;

GraphicsTSP.DrawLine(penLine, new PointF((float)X1, (float)Y1), new PointF((float)X2, (float)Y2));

ProgressBar(i);

}

X1 = (xy[m[0], 0] - xMinMax[0]) \* PicX;

Y1 = (xy[m[0], 1] - yMinMax[0]) \* PicY;

X2 = (xy[m[City - 1], 0] - xMinMax[0]) \* PicX;

Y2 = (xy[m[City - 1], 1] - yMinMax[0]) \* PicY;

GraphicsTSP.DrawLine(penLine, new PointF((float)X1, (float)Y1), new PointF((float)X2, (float)Y2));

pictureBox1.Image = imageTSP;

};

pictureBox1.Invoke(actionDraw);

}

else

{

double W = pictureBox1.Width;

double H = pictureBox1.Height;

double PicX, PicY;

PicX = (double)(W - 30) / (xMinMax[1] - xMinMax[0]);

PicY = (double)(H - 30) / (yMinMax[1] - yMinMax[0]);

SolidBrush RedBrush = new SolidBrush(Color.Red);

pictureBox1.Image = null;

Image imageTSP = new Bitmap(pictureBox1.Width, pictureBox1.Height);

Graphics GraphicsTSP = Graphics.FromImage(imageTSP);

GraphicsTSP.Clear(Color.White);

double cityX;

double cityY;

Value(0, City - 1);

for (int j = 0; j < City; j++)

{

cityX = (xy[j, 0] - xMinMax[0]) \* PicX;

cityY = (xy[j, 1] - yMinMax[0]) \* PicY;

GraphicsTSP.FillEllipse(RedBrush, new RectangleF((float)cityX, (float)cityY, 3, 3));

ProgressBar(j);

}

double X1, Y1, X2, Y2;

Pen penLine = new Pen(Color.Black, 1);

Value(1, City - 1);

for (int i = 1; i < City; i++)

{

X1 = (xy[m[i - 1], 0] - xMinMax[0]) \* PicX;

Y1 = (xy[m[i - 1], 1] - yMinMax[0]) \* PicY;

X2 = (xy[m[i], 0] - xMinMax[0]) \* PicX;

Y2 = (xy[m[i], 1] - yMinMax[0]) \* PicY;

GraphicsTSP.DrawLine(penLine, new PointF((float)X1, (float)Y1), new PointF((float)X2, (float)Y2));

ProgressBar(i);

}

X1 = (xy[m[0], 0] - xMinMax[0]) \* PicX;

Y1 = (xy[m[0], 1] - yMinMax[0]) \* PicY;

X2 = (xy[m[City - 1], 0] - xMinMax[0]) \* PicX;

Y2 = (xy[m[City - 1], 1] - yMinMax[0]) \* PicY;

GraphicsTSP.DrawLine(penLine, new PointF((float)X1, (float)Y1), new PointF((float)X2, (float)Y2));

pictureBox1.Image = imageTSP;

}

}

////////////////////////////////////////////////////////////////ACO/////////////////////////////////////////////////////////

private void ACOAlgrorithm()

{

n = City;

Pheromone = new double[City, City];

eta = new double[City, City];

delta = new double[City, City];

CitiesTabu = new int[n, City];

CityNoTabu = new int[n, City];

VisitedCity = new int[n, City];

dk = new double[n];

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

{

for (int j = 0; j < City; j++)

{

Pheromone[i, j] = 1.0;

if (i != j)

{

eta[i, j] = 1.0 / d[i, j];

}

delta[i, j] = 0.0;

}

}

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

{

for (int j = 0; j < City; j++)

{

VisitedCity[i, j] = -1;

CitiesTabu[i, j] = 0;

}

VisitedCity[i, 0] = 0;

CitiesTabu[i, 0] = 1;

}

for (int ic = 1; ic <= TMax; ic++)

{

double Pij = 0;

double PijSum = 0;

double randomPij = 0;

Random randData = new Random((int)DateTime.Now.Ticks & 0x0000FFFF);

for (int Tc = 1; Tc < City; Tc++)

{

Value(0, n);

for (int iAnt = 0; iAnt < n; iAnt++)

{

Pij = 0.0;

PijSum = 0.0;

randomPij = randData.Next(3000) / 3000.0;

for (int iCity = 0; iCity < City; iCity++)

{

if (CitiesTabu[iAnt, iCity] == 0)

{

try

{

PijSum += Math.Pow(Pheromone[VisitedCity[iAnt, Tc - 1], iCity], alpha) \* Math.Pow(eta[VisitedCity[iAnt, Tc - 1], iCity], beta);

}

catch (System.Exception ex)

{

continue;

}

}

}

for (int iCity = 0; iCity < City; iCity++)

{

if (CitiesTabu[iAnt, iCity] == 0)

{

try

{

Pij += Math.Pow(Pheromone[VisitedCity[iAnt, Tc - 1], iCity], alpha) \* Math.Pow(eta[VisitedCity[iAnt, Tc - 1], iCity], beta) / PijSum;

if (Pij > randomPij)

{

CitiesTabu[iAnt, iCity] = 1;

VisitedCity[iAnt, Tc] = iCity;

break;

}

}

catch (System.Exception ex)

{

continue;

}

}

}

ProgressBar(iAnt + 1);

}

}

int mk = 0;

boptl = optl;

Value(0, n);

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

{

dk[k] = cd(k);

if (dk[k] < optl)

{

optl = dk[k];

mk = k;

}

ProgressBar(k + 1);

}

Value(0, City);

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

{

m[i] = VisitedCity[mk, i];

ProgressBar(i + 1);

}

Value(0, n);

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

{

try

{

for (int i = 0; i < City - 1; i++)

{

delta[VisitedCity[k, i], VisitedCity[k, i + 1]] += 100 / dk[k];

}

delta[VisitedCity[k, City - 1], VisitedCity[k, 0]] += 100 / dk[k];

}

catch (System.Exception ex)

{

continue;

}

ProgressBar(k + 1);

}

Value(0, City);

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

{

for (int j = 0; j < City; j++)

{

Pheromone[i, j] = (1 - r) \* Pheromone[i, j] + delta[i, j];

if (Pheromone[i, j] < PheromoneMin)

Pheromone[i, j] = PheromoneMin;

if (Pheromone[i, j] > PheromoneMax)

Pheromone[i, j] = PheromoneMax;

}

ProgressBar(i + 1);

}

Value(0, n);

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

{

for (int c = 0; c < City; c++)

{

CitiesTabu[k, c] = 0;

if (c > 0)

{

VisitedCity[k, c] = -1;

}

}

CitiesTabu[k, VisitedCity[k, 0]] = 1;

ProgressBar(k + 1);

}

displayinfo(ic, optl);

if (pictureBox1.InvokeRequired)

{

Action actionDraw = delegate()

{

double W = pictureBox1.Width;

double H = pictureBox1.Height;

double PicX, PicY;

PicX = (double)(W - 30) / (xMinMax[1] - xMinMax[0]);

PicY = (double)(H - 30) / (yMinMax[1] - yMinMax[0]);

SolidBrush RedBrush = new SolidBrush(Color.Red);

pictureBox1.Image = null;

Image imageTSP = new Bitmap(pictureBox1.Width, pictureBox1.Height);

Graphics GraphicsTSP = Graphics.FromImage(imageTSP);

GraphicsTSP.Clear(Color.White);

double cityX;

double cityY;

Value(0, City - 1);

Font myFont1 = new Font("Hacttenschweiler", 7);

for (int j = 0; j < City; j++)

{

cityX = (xy[j, 0] - xMinMax[0]) \* PicX;

cityY = (xy[j, 1] - yMinMax[0]) \* PicY;

GraphicsTSP.FillEllipse(RedBrush, new RectangleF((float)cityX, (float)cityY, 3, 3));

ProgressBar(j);

}

double X1, Y1, X2, Y2;

Pen penLine = new Pen(Color.Black, 1);

Value(1, City - 1);

for (int i = 1; i < City; i++)

{

X1 = (xy[m[i - 1], 0] - xMinMax[0]) \* PicX;

Y1 = (xy[m[i - 1], 1] - yMinMax[0]) \* PicY;

X2 = (xy[m[i], 0] - xMinMax[0]) \* PicX;

Y2 = (xy[m[i], 1] - yMinMax[0]) \* PicY;

GraphicsTSP.DrawLine(penLine, new PointF((float)X1, (float)Y1), new PointF((float)X2, (float)Y2));

ProgressBar(i);

}

X1 = (xy[m[0], 0] - xMinMax[0]) \* PicX;

Y1 = (xy[m[0], 1] - yMinMax[0]) \* PicY;

X2 = (xy[m[City - 1], 0] - xMinMax[0]) \* PicX;

Y2 = (xy[m[City - 1], 1] - yMinMax[0]) \* PicY;

GraphicsTSP.DrawLine(penLine, new PointF((float)X1, (float)Y1), new PointF((float)X2, (float)Y2));

pictureBox1.Image = imageTSP;

};

pictureBox1.Invoke(actionDraw);

}

else

{

double W = pictureBox1.Width;

double H = pictureBox1.Height;

double PicX, PicY;

PicX = (double)(W - 30) / (xMinMax[1] - xMinMax[0]);

PicY = (double)(H - 30) / (yMinMax[1] - yMinMax[0]);

SolidBrush RedBrush = new SolidBrush(Color.Red);

pictureBox1.Image = null;

Image imageTSP = new Bitmap(pictureBox1.Width, pictureBox1.Height);

Graphics GraphicsTSP = Graphics.FromImage(imageTSP);

GraphicsTSP.Clear(Color.White);

double cityX;

double cityY;

Value(0, City - 1);

for (int j = 0; j < City; j++)

{

cityX = (xy[j, 0] - xMinMax[0]) \* PicX;

cityY = (xy[j, 1] - yMinMax[0]) \* PicY;

GraphicsTSP.FillEllipse(RedBrush, new RectangleF((float)cityX, (float)cityY, 3, 3));

ProgressBar(j);

}

double X1, Y1, X2, Y2;

Pen penLine = new Pen(Color.Black, 1);

Value(1, City - 1);

for (int i = 1; i < City; i++)

{

X1 = (xy[m[i - 1], 0] - xMinMax[0]) \* PicX;

Y1 = (xy[m[i - 1], 1] - yMinMax[0]) \* PicY;

X2 = (xy[m[i], 0] - xMinMax[0]) \* PicX;

Y2 = (xy[m[i], 1] - yMinMax[0]) \* PicY;

GraphicsTSP.DrawLine(penLine, new PointF((float)X1, (float)Y1), new PointF((float)X2, (float)Y2));

ProgressBar(i);

}

X1 = (xy[m[0], 0] - xMinMax[0]) \* PicX;

Y1 = (xy[m[0], 1] - yMinMax[0]) \* PicY;

X2 = (xy[m[City - 1], 0] - xMinMax[0]) \* PicX;

Y2 = (xy[m[City - 1], 1] - yMinMax[0]) \* PicY;

GraphicsTSP.DrawLine(penLine, new PointF((float)X1, (float)Y1), new PointF((float)X2, (float)Y2));

pictureBox1.Image = imageTSP;

}

}

}

private double cd(int k) //function for calculating the current minimum distance

{

double currentd = 0;

Value(0, City - 1);

for (int i = 0; i < City - 1; i++)

{

ProgressBar(i + 1);

currentd += d[VisitedCity[k, i], VisitedCity[k, i + 1]];

}

currentd += d[VisitedCity[k, City - 1], VisitedCity[k, 0]];

return currentd;

}

/////////////////////////////////////////////////////////form///////////////////////////////////////////////

//function of the 'File' Button

private void readDataStripMenuItem\_Click(object sender, EventArgs e)

{

label6.Text = ""; // current route length

label8.Text = ""; //dimension (number of city)

label9.Text = ""; //name

label7.Text = ""; //optimum route length

label10.Text = ""; //status of solver

OpenFileDialog myFileDialog = new OpenFileDialog();

myFileDialog.InitialDirectory = Application.StartupPath + "\\Data";

xMinMax = new double[2] { double.MaxValue, 0 };

yMinMax = new double[2] { double.MaxValue, 0 };

if (myFileDialog.ShowDialog() == DialogResult.OK)

{

string strPath = myFileDialog.FileName;

StreamReader streamReader = new StreamReader(strPath);

string Data;

string[] Array;

string dataFlag = "no";

int i = 0;

while ((Data = streamReader.ReadLine()) != "EOF")

{

if (Data.Contains("NAME"))

{

Array = Data.Split(':');

label9.Text = Array[1];

}

if (Data.Contains("Optimum"))

{

int si; //starting index

si = Data.IndexOf("is") + 2;

label7.Text = Data.Substring(si);

}

if (Data.Contains("DIMENSION"))

{

Array = Data.Split(':');

label8.Text = Array[1] + " cities";

City = int.Parse(Array[1].Trim());

xy = new double[City, 2];

m = new int[City];

d = new double[City, City];

CityList = new int[City];

Value(0, City);

}

if (dataFlag == "yes")

{

Array = Data.Split(' ');

xy[i, 0] = double.Parse(Array[1]);

xy[i, 1] = double.Parse(Array[2]);

if (xMinMax[0] > xy[i, 0])

{

xMinMax[0] = xy[i, 0];

}

if (xMinMax[1] < xy[i, 0])

{

xMinMax[1] = xy[i, 0];

}

if (yMinMax[0] > xy[i, 1])

{

yMinMax[0] = xy[i, 1];

}

if (yMinMax[1] < xy[i, 1])

{

yMinMax[1] = xy[i, 1];

}

i++;

ProgressBar(i);

}

if (Data.Contains("NODE\_COORD\_SECTION"))

{

dataFlag = "yes";

}

}

streamReader.Close();

//Draw city points

Value(0, City);

if (pictureBox1.InvokeRequired)

{

Action actionDrawCityPoint = delegate()

{

double W = pictureBox1.Width;

double H = pictureBox1.Height;

double PicX, PicY;

PicX = (double)(W - 30) / (xMinMax[1] - xMinMax[0]);

PicY = (double)(H - 30) / (yMinMax[1] - yMinMax[0]);

SolidBrush RedBrush = new SolidBrush(Color.Red);

pictureBox1.Image = null;

Image imageTSP = new Bitmap(pictureBox1.Width, pictureBox1.Height);

Graphics GraphicsTSP = Graphics.FromImage(imageTSP);

GraphicsTSP.Clear(Color.White);

double cityX;

double cityY;

for (int j = 0; j < City; j++)

{

cityX = (xy[j, 0] - xMinMax[0]) \* PicX;

cityY = (xy[j, 1] - yMinMax[0]) \* PicY;

GraphicsTSP.FillEllipse(RedBrush, new RectangleF((float)cityX, (float)cityY, 3, 3));

ProgressBar(j + 1);

}

pictureBox1.Image = imageTSP;

};

pictureBox1.Invoke(actionDrawCityPoint);

}

else

{

double W = pictureBox1.Width;

double H = pictureBox1.Height;

double PicX, PicY;

PicX = (double)(W - 30) / (xMinMax[1] - xMinMax[0]);

PicY = (double)(H - 30) / (yMinMax[1] - yMinMax[0]);

SolidBrush RedBrush = new SolidBrush(Color.Red);

pictureBox1.Image = null;

Image imageTSP = new Bitmap(pictureBox1.Width, pictureBox1.Height);

Graphics GraphicsTSP = Graphics.FromImage(imageTSP);

GraphicsTSP.Clear(Color.White);

double cityX;

double cityY;

for (int j = 0; j < City; j++)

{

cityX = (xy[j, 0] - xMinMax[0]) \* PicX;

cityY = (xy[j, 1] - yMinMax[0]) \* PicY;

GraphicsTSP.FillEllipse(RedBrush, new RectangleF((float)cityX, (float)cityY, 3, 3));

ProgressBar(j + 1);

}

pictureBox1.Image = imageTSP;

}

Value(0, City);

for (int j = 0; j < City; j++)

{

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

{

d[j, k] = Math.Sqrt((xy[j, 0] - xy[k, 0]) \* (xy[j, 0] - xy[k, 0]) + (xy[j, 1] - xy[k, 1]) \* (xy[j, 1] - xy[k, 1]));

}

ProgressBar(j + 1);

}

}

}

//define the progress bar

public void ProgressBar(int BarValue)

{

if (progressBarTSP.InvokeRequired)

{

Action<int> actionUpdateProgressBar = delegate(int Temp)

{

progressBarTSP.Value = Temp;

};

progressBarTSP.Invoke(actionUpdateProgressBar, BarValue > progressBarTSP.Maximum ? progressBarTSP.Maximum : BarValue);

}

else

{

progressBarTSP.Value = BarValue;

}

}

//set the value of the progress bar

public void Value(int minValue, int maxValue)

{

if (progressBarTSP.InvokeRequired)

{

Action<int, int> actionSetBarValue = delegate(int minTemp, int maxTemp)

{

progressBarTSP.Minimum = minTemp;

progressBarTSP.Maximum = maxTemp;

};

progressBarTSP.Invoke(actionSetBarValue, minValue, maxValue);

}

else

{

progressBarTSP.Minimum = minValue;

progressBarTSP.Maximum = maxValue;

}

}

private void FormTSP\_Load(object sender, EventArgs e)

{

label6.Text = "";

label8.Text = "";

label9.Text = "";

label7.Text = "";

label10.Text = "";

}

//function of the 'Greedy Solver' Button

private void greedySolverToolStripMenuItem\_Click(object sender, EventArgs e)

{

label10.Text = "Greedy algorithm progress";

optl = double.PositiveInfinity;

boptl = double.PositiveInfinity;

Thread threadGreedy = new Thread(new ThreadStart(Greedy));

threadGreedy.IsBackground = true;

threadGreedy.Start();

}

//function of the 'ACO Solver' Button

private void ACOSolverToolStripMenuItem\_Click(object sender, EventArgs e)

{

label10.Text = "ACO algorithm progress";

optl = double.PositiveInfinity;

boptl = double.PositiveInfinity;

int T = 0;

Thread threadACO = new Thread(new ThreadStart(ACOAlgrorithm));

threadACO.IsBackground = true;

threadACO.Start();

}

public void displayinfo(int p, double ml)

{

double fImprovement = 0;

if (label6.InvokeRequired)

{

Action<double> l = delegate(double sl)

{

label6.Text = sl == 0 ? "" : sl.ToString("F0");

};

label6.Invoke(l, ml);

}

else

{

label6.Text = ml == 0 ? "" : ml.ToString("F0");

}

}

////function of the 'Exit' Button

private void exitToolStripMenuItem\_Click\_1(object sender, EventArgs e)

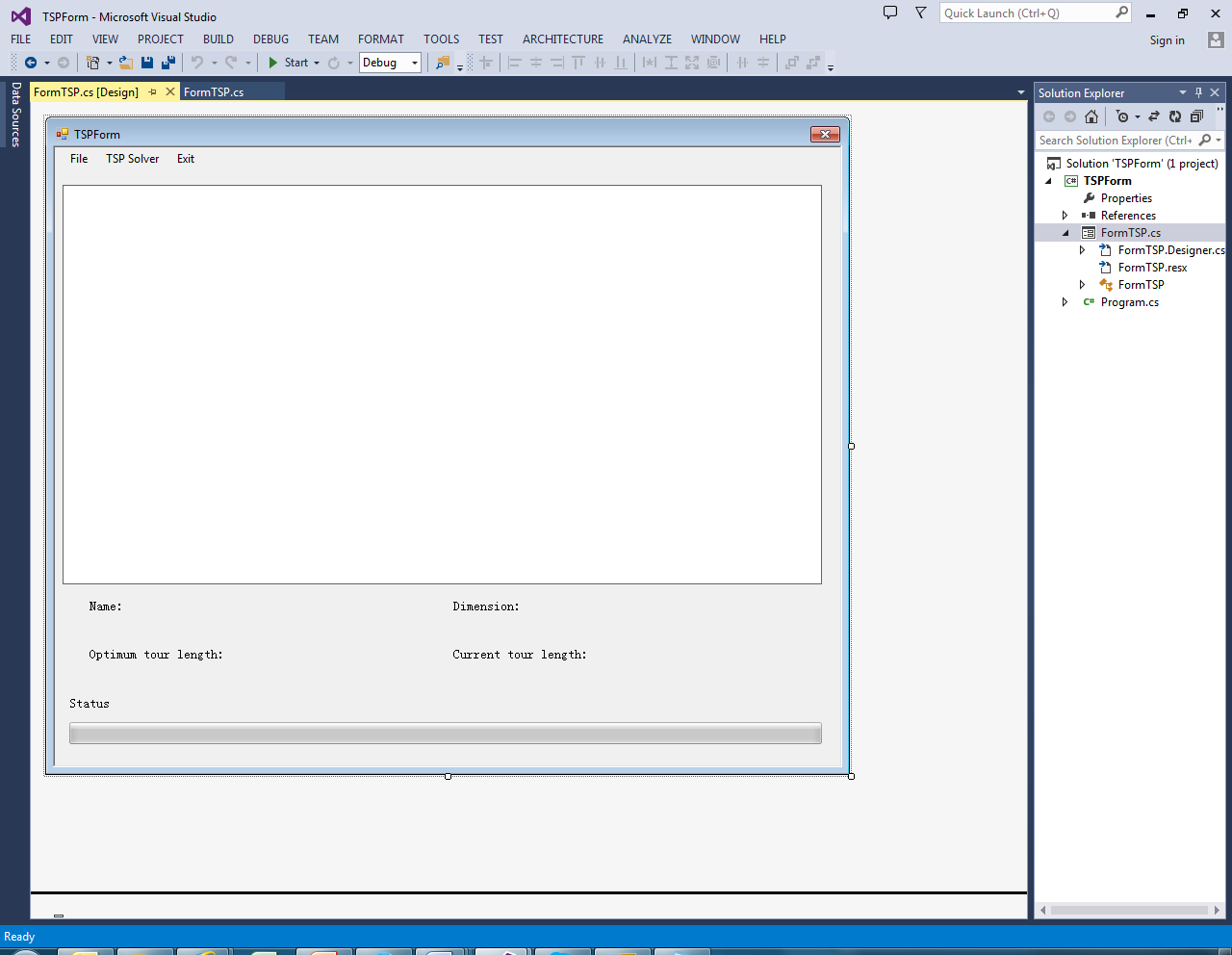
{

Application.Exit();

}

}

}



**Yechen Tan 1337236**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Windows.Forms;

namespace TSPForm

{

public partial class FormTSP : Form

{

private int Wholecities;

private double[] xMinMax;

private double[] yMinMax;

private double[,] xy;

private double a;

private static double b;

private int[] goodPathIndex;

private double[,] CityDistance;

private int[] CItytuopu;

private double shortesPathLenth = double.PositiveInfinity;

private double BeforeShortPathLegth = double.PositiveInfinity;

int[,] CityAntHaveTrave;

double[,] PheromoneTau;

double[,] etaCij;

double[,] deltaTau;

int[,] WholecitiesTabu;

int[,] CityNoTabu;

double[] mAntShortestDistance;

int AntCount;

int NcMax = 3000;

double Q = 100;

double afa = 2.0;

double beita = 7.0;

double rho = 0.5;

int iHaveNoChange = 50;

int Nc = 0; //the current number of iterations

double PhenomenonZero = 1e-5;

double PhenomenonRangeMax =30;

public FormTSP()

{

InitializeComponent();

}

/// <summary>

///

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void readDataToolStripMenuItem\_Click(object sender, EventArgs e)

{

labelCTLength.Text = "";

labelDimension.Text = "";

labelName.Text = "";

labelOPTLength.Text = "";

labelImprovement.Text = "";

labelProgress.Text = "";

OpenFileDialog myFileDialog = new OpenFileDialog();

myFileDialog.InitialDirectory = Application.StartupPath + "\\Data";

xMinMax = new double[2] { double.MaxValue, 0 };

yMinMax = new double[2] { double.MaxValue, 0 };

if (myFileDialog.ShowDialog() == DialogResult.OK)

{

string strPath = myFileDialog.FileName;

StreamReader streamReader = new StreamReader(strPath);

string stringData;

string[] stringArray;

string dataFlag = "no";

int i = 0;

while ((stringData = streamReader.ReadLine()) != "EOF")

{

if (stringData.Contains("NAME"))

{

stringArray = stringData.Split(':');

labelName.Text = stringArray[1];

}

if (stringData.Contains("Optimum"))

{

int iLenthStartIndex;

iLenthStartIndex = stringData.IndexOf("is") + 2;

labelOPTLength.Text = stringData.Substring(iLenthStartIndex);

}

if (stringData.Contains("DIMENSION"))

{

stringArray = stringData.Split(':');

labelDimension.Text = stringArray[1] + " Wholecities";

Wholecities = int.Parse(stringArray[1].Trim());

xy = new double[Wholecities, 2];

goodPathIndex = new int[Wholecities];

CityDistance = new double[Wholecities, Wholecities];

CItytuopu = new int[Wholecities];

setBarValue(0, Wholecities);

}

if (dataFlag == "yes")

{

stringArray = stringData.Split(' ');

xy[i, 0] = double.Parse(stringArray[1]);

xy[i, 1] = double.Parse(stringArray[2]);

if (xMinMax[0] > xy[i, 0])

{

xMinMax[0] = xy[i, 0];

}

if (xMinMax[1] < xy[i, 0])

{

xMinMax[1] = xy[i, 0];

}

if (yMinMax[0] > xy[i, 1])

{

yMinMax[0] = xy[i, 1];

}

if (yMinMax[1] < xy[i, 1])

{

yMinMax[1] = xy[i, 1];

}

i++;

UpadateProgressBar(i);

}

if (stringData.Contains("NODE\_COORD\_SECTION"))

{

dataFlag = "yes";

}

}

streamReader.Close();

DrawCityPoint();

CalculateDistance();

}

}

/// <summary>

///

/// </summary>

private void CalculateDistance()

{

setBarValue(0, Wholecities);

for (int j = 0; j < Wholecities; j++)

{

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

{

CityDistance[j, k] = Math.Sqrt((xy[j, 0] - xy[k, 0]) \* (xy[j, 0] - xy[k, 0]) + (xy[j, 1] - xy[k, 1]) \* (xy[j, 1] - xy[k, 1]));

}

UpadateProgressBar(j + 1);

}

}

/// <summary>

///

/// </summary>

/// <param name="iBarValue"></param>

public void UpadateProgressBar(int iBarValue)

{

if (progressBarTSP.InvokeRequired)

{

Action<int> actionUpdateProgressBar = delegate(int iTemp)

{

progressBarTSP.Value = iTemp;

};

progressBarTSP.Invoke(actionUpdateProgressBar, iBarValue > progressBarTSP.Maximum ? progressBarTSP.Maximum : iBarValue);

}

else

{

progressBarTSP.Value = iBarValue;

}

}

/// <summary>

///

/// </summary>

/// <param name="minValue"></param>

/// <param name="maxValue"></param>

public void setBarValue(int minValue, int maxValue)

{

if (progressBarTSP.InvokeRequired)

{

Action<int, int> actionSetBarValue = delegate(int minTemp, int maxTemp)

{

progressBarTSP.Minimum = minTemp;

progressBarTSP.Maximum = maxTemp;

};

progressBarTSP.Invoke(actionSetBarValue, minValue, maxValue);

}

else

{

progressBarTSP.Minimum = minValue;

progressBarTSP.Maximum = maxValue;

}

}

/// <summary>

///

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void FormTSP\_Load(object sender, EventArgs e)

{

labelCTLength.Text = "";

labelDimension.Text = "";

labelName.Text = "";

labelOPTLength.Text = "";

labelImprovement.Text = "";

labelProgress.Text = "";

}

/// <summary>

///

/// </summary>

private void DrawCityPoint()

{

setBarValue(0, Wholecities);

if (pictureBoxCity.InvokeRequired)

{

Action actionDrawCityPoint = delegate()

{

double fPicboxWidth = pictureBoxCity.Width;

double fPicboxheight = pictureBoxCity.Height;

double PicAxisX, PicAxisY;

PicAxisX = (double)(fPicboxWidth - 30) / (xMinMax[1] - xMinMax[0]);

PicAxisY = (double)(fPicboxheight - 30) / (yMinMax[1] - yMinMax[0]);

SolidBrush RedBrush = new SolidBrush(Color.Red);

pictureBoxCity.Image = null;

Image imageTSP = new Bitmap(pictureBoxCity.Width, pictureBoxCity.Height);

Graphics GraphicsTSP = Graphics.FromImage(imageTSP);

GraphicsTSP.Clear(Color.White);

double cityPointX;

double cityPointY;

for (int j = 0; j < Wholecities; j++)

{

cityPointX = (xy[j, 0] - xMinMax[0]) \* PicAxisX;

cityPointY = (xy[j, 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.FillEllipse(RedBrush, new RectangleF((float)cityPointX, (float)cityPointY, 3, 3));

UpadateProgressBar(j + 1);

}

pictureBoxCity.Image = imageTSP;

};

pictureBoxCity.Invoke(actionDrawCityPoint);

}

else

{

double fPicboxWidth = pictureBoxCity.Width;

double fPicboxheight = pictureBoxCity.Height;

double PicAxisX, PicAxisY;

PicAxisX = (double)(fPicboxWidth - 30) / (xMinMax[1] - xMinMax[0]);

PicAxisY = (double)(fPicboxheight - 30) / (yMinMax[1] - yMinMax[0]);

SolidBrush RedBrush = new SolidBrush(Color.Red);

pictureBoxCity.Image = null;

Image imageTSP = new Bitmap(pictureBoxCity.Width, pictureBoxCity.Height);

Graphics GraphicsTSP = Graphics.FromImage(imageTSP);

GraphicsTSP.Clear(Color.White);

double cityPointX;

double cityPointY;

for (int j = 0; j < Wholecities; j++)

{

cityPointX = (xy[j, 0] - xMinMax[0]) \* PicAxisX;

cityPointY = (xy[j, 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.FillEllipse(RedBrush, new RectangleF((float)cityPointX, (float)cityPointY, 3, 3));

UpadateProgressBar(j + 1);

}

pictureBoxCity.Image = imageTSP;

}

}

/// <summary>

///

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void exitToolStripMenuItem\_Click(object sender, EventArgs e)

{

Application.Exit();

}

/// <summary>

///

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void greedySolverToolStripMenuItem\_Click(object sender, EventArgs e)

{

labelProgress.Text = "Greedy algorithm progress";

shortesPathLenth = double.PositiveInfinity;

BeforeShortPathLegth = double.PositiveInfinity;

Thread threadGreedy = new Thread(new ThreadStart(GreedyAlgorithm));

threadGreedy.IsBackground = true;

threadGreedy.Start();

}

/// <summary>

///

/// </summary>

private void GreedyAlgorithm()

{

int[] GreedyhaveTrave = new int[Wholecities];

double cityShortestDistance = double.PositiveInfinity;

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

{

goodPathIndex[i] = 0;

CItytuopu[i] = 0;

GreedyhaveTrave[i] = -1;

}

GreedyhaveTrave[0] = 0;

CItytuopu[GreedyhaveTrave[0]] = 1;

int minCityIndex = GreedyhaveTrave[0];

setBarValue(1, Wholecities - 1);

for (int i = 1; i < Wholecities; i++)

{

cityShortestDistance = double.PositiveInfinity;

for (int j = 0; j < Wholecities; j++)

{

if (CityDistance[GreedyhaveTrave[i - 1], j] < cityShortestDistance && CItytuopu[j] == 0)

{

minCityIndex = j;

cityShortestDistance = CityDistance[GreedyhaveTrave[i - 1], j];

}

}

UpadateProgressBar(i);

CItytuopu[minCityIndex] = 1;

GreedyhaveTrave[i] = minCityIndex;

WriteProgressInfo(i, cityShortestDistance);

}

shortesPathLenth = 0;

for (int k = 0; k < Wholecities - 1; k++)

{

shortesPathLenth += CityDistance[GreedyhaveTrave[k], GreedyhaveTrave[k + 1]];

}

shortesPathLenth += CityDistance[GreedyhaveTrave[Wholecities - 1], GreedyhaveTrave[0]];

WriteProgressInfo(Wholecities, shortesPathLenth);

goodPathIndex = GreedyhaveTrave;

DrawShortestPath();

}

/// <summary>

///

/// </summary>

/// <param name="iterationCount"></param>

/// <param name="minimumLength"></param>

public void WriteProgressInfo(int iterationCount, double minimumLength)

{

double fImprovement = 0;

if (labelImprovement.InvokeRequired)

{

Action<int, double> actionIteration = delegate(int iterationCountTemp, double minimumLengthTemp)

{

fImprovement = BeforeShortPathLegth == double.PositiveInfinity ? 0 : BeforeShortPathLegth - minimumLengthTemp;

labelImprovement.Text = "(t=" + iterationCountTemp.ToString() + ")," + fImprovement.ToString("F0");

};

labelImprovement.Invoke(actionIteration, iterationCount, minimumLength);

}

else

{

fImprovement = BeforeShortPathLegth == double.PositiveInfinity ? 0 : BeforeShortPathLegth - minimumLength;

labelImprovement.Text = "(t=" + iterationCount.ToString() + ")," + fImprovement.ToString("F0");

}

if (labelCTLength.InvokeRequired)

{

Action<double> actionLength = delegate(double shortLengh)

{

labelCTLength.Text = shortLengh == 0 ? "" : shortLengh.ToString("F0");

};

labelCTLength.Invoke(actionLength, minimumLength);

}

else

{

labelCTLength.Text = minimumLength == 0 ? "" : minimumLength.ToString("F0");

}

}

/// <summary>

///

/// </summary>

public void DrawShortestPath()

{

if (pictureBoxCity.InvokeRequired)

{

Action actionDraw = delegate()

{

double fPicboxWidth = pictureBoxCity.Width;

double fPicboxheight = pictureBoxCity.Height;

double PicAxisX, PicAxisY;

PicAxisX = (double)(fPicboxWidth - 30) / (xMinMax[1] - xMinMax[0]);

PicAxisY = (double)(fPicboxheight - 30) / (yMinMax[1] - yMinMax[0]);

SolidBrush RedBrush = new SolidBrush(Color.Red);

pictureBoxCity.Image = null;

Image imageTSP = new Bitmap(pictureBoxCity.Width, pictureBoxCity.Height);

Graphics GraphicsTSP = Graphics.FromImage(imageTSP);

GraphicsTSP.Clear(Color.White);

double cityPointX;

double cityPointY;

setBarValue(0, Wholecities - 1);

Font myFont1 = new Font("Hacttenschweiler", 7);

for (int j = 0; j < Wholecities; j++)

{

cityPointX = (xy[j, 0] - xMinMax[0]) \* PicAxisX;

cityPointY = (xy[j, 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.FillEllipse(RedBrush, new RectangleF((float)cityPointX, (float)cityPointY, 3, 3));

UpadateProgressBar(j);

}

double FirstPointX, FirstPointY, SecondPointX, SecondPointY;

Pen penLine = new Pen(Color.Black, 1);

setBarValue(1, Wholecities - 1);

for (int i = 1; i < Wholecities; i++)

{

FirstPointX = (xy[goodPathIndex[i - 1], 0] - xMinMax[0]) \* PicAxisX;

FirstPointY = (xy[goodPathIndex[i - 1], 1] - yMinMax[0]) \* PicAxisY;

SecondPointX = (xy[goodPathIndex[i], 0] - xMinMax[0]) \* PicAxisX;

SecondPointY = (xy[goodPathIndex[i], 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.DrawLine(penLine, new PointF((float)FirstPointX, (float)FirstPointY), new PointF((float)SecondPointX, (float)SecondPointY));

UpadateProgressBar(i);

}

FirstPointX = (xy[goodPathIndex[0], 0] - xMinMax[0]) \* PicAxisX;

FirstPointY = (xy[goodPathIndex[0], 1] - yMinMax[0]) \* PicAxisY;

SecondPointX = (xy[goodPathIndex[Wholecities - 1], 0] - xMinMax[0]) \* PicAxisX;

SecondPointY = (xy[goodPathIndex[Wholecities - 1], 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.DrawLine(penLine, new PointF((float)FirstPointX, (float)FirstPointY), new PointF((float)SecondPointX, (float)SecondPointY));

pictureBoxCity.Image = imageTSP;

};

pictureBoxCity.Invoke(actionDraw);

}

else

{

double fPicboxWidth = pictureBoxCity.Width;

double fPicboxheight = pictureBoxCity.Height;

double PicAxisX, PicAxisY;

PicAxisX = (double)(fPicboxWidth - 30) / (xMinMax[1] - xMinMax[0]);

PicAxisY = (double)(fPicboxheight - 30) / (yMinMax[1] - yMinMax[0]);

SolidBrush RedBrush = new SolidBrush(Color.Red);

pictureBoxCity.Image = null;

Image imageTSP = new Bitmap(pictureBoxCity.Width, pictureBoxCity.Height);

Graphics GraphicsTSP = Graphics.FromImage(imageTSP);

GraphicsTSP.Clear(Color.White);

double cityPointX;

double cityPointY;

setBarValue(0, Wholecities - 1);

for (int j = 0; j < Wholecities; j++)

{

cityPointX = (xy[j, 0] - xMinMax[0]) \* PicAxisX;

cityPointY = (xy[j, 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.FillEllipse(RedBrush, new RectangleF((float)cityPointX, (float)cityPointY, 3, 3));

UpadateProgressBar(j);

}

double FirstPointX, FirstPointY, SecondPointX, SecondPointY;

Pen penLine = new Pen(Color.Black, 1);

setBarValue(1, Wholecities - 1);

for (int i = 1; i < Wholecities; i++)

{

FirstPointX = (xy[goodPathIndex[i - 1], 0] - xMinMax[0]) \* PicAxisX;

FirstPointY = (xy[goodPathIndex[i - 1], 1] - yMinMax[0]) \* PicAxisY;

SecondPointX = (xy[goodPathIndex[i], 0] - xMinMax[0]) \* PicAxisX;

SecondPointY = (xy[goodPathIndex[i], 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.DrawLine(penLine, new PointF((float)FirstPointX, (float)FirstPointY), new PointF((float)SecondPointX, (float)SecondPointY));

UpadateProgressBar(i);

}

FirstPointX = (xy[goodPathIndex[0], 0] - xMinMax[0]) \* PicAxisX;

FirstPointY = (xy[goodPathIndex[0], 1] - yMinMax[0]) \* PicAxisY;

SecondPointX = (xy[goodPathIndex[Wholecities - 1], 0] - xMinMax[0]) \* PicAxisX;

SecondPointY = (xy[goodPathIndex[Wholecities - 1], 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.DrawLine(penLine, new PointF((float)FirstPointX, (float)FirstPointY), new PointF((float)SecondPointX, (float)SecondPointY));

pictureBoxCity.Image = imageTSP;

}

}

private void aCOSolverToolStripMenuItem\_Click(object sender, EventArgs e)

{

labelProgress.Text = "ACO algorithm progress";

shortesPathLenth = double.PositiveInfinity;

BeforeShortPathLegth = double.PositiveInfinity;

Nc = 0;

Thread threadACO = new Thread(new ThreadStart(ACOAlgrorithm));

threadACO.IsBackground = true;

threadACO.Start();

}

/// <summary>

///

/// </summary>

private void ACOAlgrorithm()

{

AntCount = Wholecities;

PheromoneTau = new double[Wholecities, Wholecities];

etaCij = new double[Wholecities, Wholecities];

deltaTau = new double[Wholecities, Wholecities];

WholecitiesTabu = new int[AntCount, Wholecities];

CityNoTabu = new int[AntCount, Wholecities];

CityAntHaveTrave = new int[AntCount, Wholecities];

mAntShortestDistance = new double[AntCount];

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

{

for (int j = 0; j < Wholecities; j++)

{

PheromoneTau[i, j] = 1.0;

if (i != j)

{

etaCij[i, j] = 1.0 / CityDistance[i, j];

}

deltaTau[i, j] = 0.0;

}

}

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

{

for (int j = 0; j < Wholecities; j++)

{

CityAntHaveTrave[i, j] = -1;

WholecitiesTabu[i, j] = 0;

}

CityAntHaveTrave[i, 0] = 0;

WholecitiesTabu[i, 0] = 1;

}

for (int iterationCount = 1; iterationCount <= NcMax; iterationCount++)

{

startAntMove();

int minLenthAntNo = 0;

BeforeShortPathLegth = shortesPathLenth;

setBarValue(0, AntCount);

for (int antNo = 0; antNo < AntCount; antNo++)

{

mAntShortestDistance[antNo] = calculateMinDistance(antNo);

if (mAntShortestDistance[antNo] < shortesPathLenth)

{

shortesPathLenth = mAntShortestDistance[antNo];

minLenthAntNo = antNo;

}

UpadateProgressBar(antNo + 1);

}

setBarValue(0, Wholecities);

for (int cityNO = 0; cityNO < Wholecities; cityNO++)

{

goodPathIndex[cityNO] = CityAntHaveTrave[minLenthAntNo, cityNO];

UpadateProgressBar(cityNO + 1);

}

UpdatePathIncrementPheromone();

setBarValue(0, Wholecities);

for (int icityNo = 0; icityNo < Wholecities; icityNo++)

{

for (int jcityNo = 0; jcityNo < Wholecities; jcityNo++)

{

PheromoneTau[icityNo, jcityNo] = (1 - rho) \* PheromoneTau[icityNo, jcityNo] + deltaTau[icityNo, jcityNo];

if (PheromoneTau[icityNo, jcityNo] < PhenomenonZero)

PheromoneTau[icityNo, jcityNo] = PhenomenonZero;

if (PheromoneTau[icityNo, jcityNo] > PhenomenonRangeMax)

PheromoneTau[icityNo, jcityNo] = PhenomenonRangeMax;

}

UpadateProgressBar(icityNo + 1);

}

setBarValue(0, AntCount);

for (int antNo = 0; antNo < AntCount; antNo++)

{

for (int cityNo = 0; cityNo < Wholecities; cityNo++)

{

WholecitiesTabu[antNo, cityNo] = 0;

if (cityNo > 0)

{

CityAntHaveTrave[antNo, cityNo] = -1;

}

}

WholecitiesTabu[antNo, CityAntHaveTrave[antNo, 0]] = 1;

UpadateProgressBar(antNo + 1);

}

WriteProgressInfo(iterationCount, shortesPathLenth);

DrawShortestPath();

}

}

/// <summary>

///

/// </summary>

private void startAntMove()

{

double antPij = 0;

double PijSum = 0;

double randomPij = 0;

Random randData = new Random((int)DateTime.Now.Ticks & 0x0000FFFF);

for (int haveTraveCity = 1; haveTraveCity < Wholecities; haveTraveCity++)

{

setBarValue(0, AntCount);

for (int iAnt = 0; iAnt < AntCount; iAnt++)

{

antPij = 0.0;

PijSum = 0.0;

randomPij = randData.Next(3000) / 3000.0;

for (int iCity = 0; iCity < Wholecities; iCity++)

{

if (WholecitiesTabu[iAnt, iCity] == 0)

{

try

{

PijSum += Math.Pow(PheromoneTau[CityAntHaveTrave[iAnt, haveTraveCity - 1], iCity], afa) \* Math.Pow(etaCij[CityAntHaveTrave[iAnt, haveTraveCity - 1], iCity], beita);

}

catch (System.Exception ex)

{

continue;

}

}

}

for (int iCity = 0; iCity < Wholecities; iCity++)

{

if (WholecitiesTabu[iAnt, iCity] == 0)

{

try

{

antPij += Math.Pow(PheromoneTau[CityAntHaveTrave[iAnt, haveTraveCity - 1], iCity], afa) \* Math.Pow(etaCij[CityAntHaveTrave[iAnt, haveTraveCity - 1], iCity], beita) / PijSum;

if (antPij > randomPij)

{

WholecitiesTabu[iAnt, iCity] = 1;

CityAntHaveTrave[iAnt, haveTraveCity] = iCity;

break;

}

}

catch (System.Exception ex)

{

continue;

}

}

}

UpadateProgressBar(iAnt + 1);

}

}

}

/// <summary>

///

/// </summary>

private void UpdatePathIncrementPheromone()

{

setBarValue(0, AntCount);

for (int antNo = 0; antNo < AntCount; antNo++)

{

try

{

for (int cityNo = 0; cityNo < Wholecities - 1; cityNo++)

{

deltaTau[CityAntHaveTrave[antNo, cityNo], CityAntHaveTrave[antNo, cityNo + 1]] += 100 / mAntShortestDistance[antNo];

}

deltaTau[CityAntHaveTrave[antNo, Wholecities - 1], CityAntHaveTrave[antNo, 0]] += 100 / mAntShortestDistance[antNo];

}

catch (System.Exception ex)

{

continue;

}

UpadateProgressBar(antNo + 1);

}

}

/// <summary>

///

/// </summary>

/// <param name="antNo"></param>

/// <returns></returns>

private double calculateMinDistance(int antNo)

{

double currentMinDis = 0;

setBarValue(0, Wholecities - 1);

for (int cityNO = 0; cityNO < Wholecities - 1; cityNO++)

{

UpadateProgressBar(cityNO + 1);

currentMinDis += CityDistance[CityAntHaveTrave[antNo, cityNO], CityAntHaveTrave[antNo, cityNO + 1]];

}

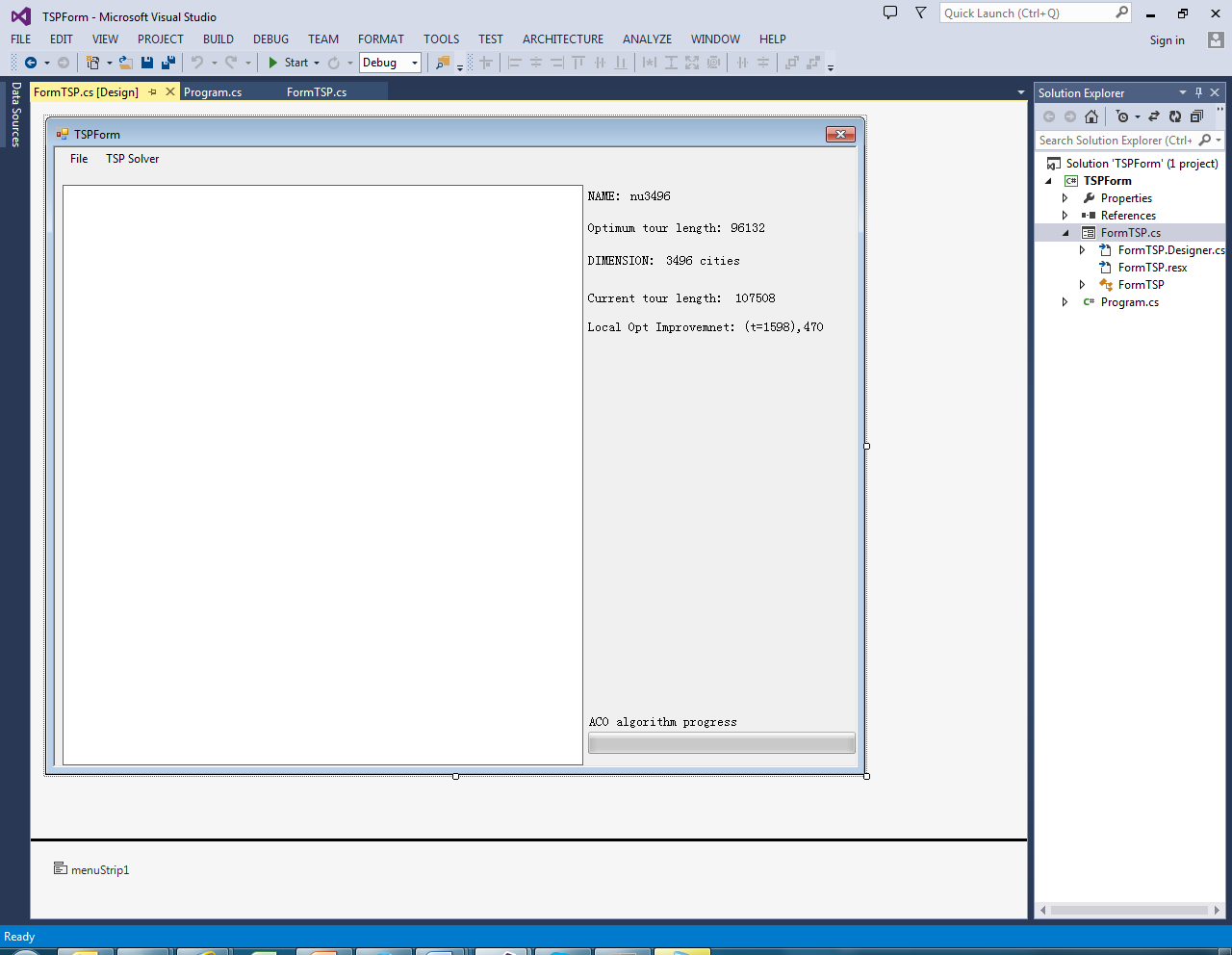
currentMinDis += CityDistance[CityAntHaveTrave[antNo, Wholecities - 1], CityAntHaveTrave[antNo, 0]];

return currentMinDis;

}

}

}



**Yang Song 1417886**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Windows.Forms;

namespace TSPForm

{

public partial class FormTSP : Form

{

private int Cities;

private double[] xMinMax;

private double[] yMinMax;

private double[,] xy;

private double a;

private static double b;

private int[] goodPathIndex;

private double[,] CityDistance;

private int[] CityTabu;

private double shortesPathLenth = double.PositiveInfinity;

private double BeforeShortPathLegth = double.PositiveInfinity;

int[,] CityAntHaveTrave;

double[,] PheromoneTau;

double[,] etaCij;

double[,] deltaTau;

int[,] CitiesTabu;

int[,] CityNoTabu;

double[] mAntShortestDistance;

int AntCount;

int NcMax = 3000;

double Q = 100;

double alpha = 2.0;

double beta = 7.0;

double rho = 0.5;

int iHaveNoChange = 50;

int Nc = 0; //当前迭代次数

double PhenomenonZero = 1e-5;

double PhenomenonRangeMax =30;

public FormTSP()

{

InitializeComponent();

}

/// <summary>

///

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void readDataToolStripMenuItem\_Click(object sender, EventArgs e)

{

labelCTLength.Text = "";

labelDimension.Text = "";

labelName.Text = "";

labelOPTLength.Text = "";

labelImprovement.Text = "";

labelProgress.Text = "";

OpenFileDialog myFileDialog = new OpenFileDialog();

myFileDialog.InitialDirectory = Application.StartupPath + "\\Data";

xMinMax = new double[2] { double.MaxValue, 0 };

yMinMax = new double[2] { double.MaxValue, 0 };

if (myFileDialog.ShowDialog() == DialogResult.OK)

{

string strPath = myFileDialog.FileName;

StreamReader streamReader = new StreamReader(strPath);

string stringData;

string[] stringArray;

string dataFlag = "no";

int i = 0;

while ((stringData = streamReader.ReadLine()) != "EOF")

{

if (stringData.Contains("NAME"))

{

stringArray = stringData.Split(':');

labelName.Text = stringArray[1];

}

if (stringData.Contains("Optimum"))

{

int iLenthStartIndex;

iLenthStartIndex = stringData.IndexOf("is") + 2;

labelOPTLength.Text = stringData.Substring(iLenthStartIndex);

}

if (stringData.Contains("DIMENSION"))

{

stringArray = stringData.Split(':');

labelDimension.Text = stringArray[1] + " cities";

Cities = int.Parse(stringArray[1].Trim());

xy = new double[Cities, 2];

goodPathIndex = new int[Cities];

CityDistance = new double[Cities, Cities];

CityTabu = new int[Cities];

setBarValue(0, Cities);

}

if (dataFlag == "yes")

{

stringArray = stringData.Split(' ');

xy[i, 0] = double.Parse(stringArray[1]);

xy[i, 1] = double.Parse(stringArray[2]);

if (xMinMax[0] > xy[i, 0])

{

xMinMax[0] = xy[i, 0];

}

if (xMinMax[1] < xy[i, 0])

{

xMinMax[1] = xy[i, 0];

}

if (yMinMax[0] > xy[i, 1])

{

yMinMax[0] = xy[i, 1];

}

if (yMinMax[1] < xy[i, 1])

{

yMinMax[1] = xy[i, 1];

}

i++;

UpadateProgressBar(i);

}

if (stringData.Contains("NODE\_COORD\_SECTION"))

{

dataFlag = "yes";

}

}

streamReader.Close();

DrawCityPoint();

CalculateDistance();

}

}

/// <summary>

///

/// </summary>

private void CalculateDistance()

{

setBarValue(0, Cities);

for (int j = 0; j < Cities; j++)

{

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

{

CityDistance[j, k] = Math.Sqrt((xy[j, 0] - xy[k, 0]) \* (xy[j, 0] - xy[k, 0]) + (xy[j, 1] - xy[k, 1]) \* (xy[j, 1] - xy[k, 1]));

}

UpadateProgressBar(j + 1);

}

}

/// <summary>

///

/// </summary>

/// <param name="iBarValue"></param>

public void UpadateProgressBar(int iBarValue)

{

if (progressBarTSP.InvokeRequired)

{

Action<int> actionUpdateProgressBar = delegate(int iTemp)

{

progressBarTSP.Value = iTemp;

};

progressBarTSP.Invoke(actionUpdateProgressBar, iBarValue > progressBarTSP.Maximum ? progressBarTSP.Maximum : iBarValue);

}

else

{

progressBarTSP.Value = iBarValue;

}

}

/// <summary>

///

/// </summary>

/// <param name="minValue"></param>

/// <param name="maxValue"></param>

public void setBarValue(int minValue, int maxValue)

{

if (progressBarTSP.InvokeRequired)

{

Action<int, int> actionSetBarValue = delegate(int minTemp, int maxTemp)

{

progressBarTSP.Minimum = minTemp;

progressBarTSP.Maximum = maxTemp;

};

progressBarTSP.Invoke(actionSetBarValue, minValue, maxValue);

}

else

{

progressBarTSP.Minimum = minValue;

progressBarTSP.Maximum = maxValue;

}

}

/// <summary>

///

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void FormTSP\_Load(object sender, EventArgs e)

{

labelCTLength.Text = "";

labelDimension.Text = "";

labelName.Text = "";

labelOPTLength.Text = "";

labelImprovement.Text = "";

labelProgress.Text = "";

}

/// <summary>

///

/// </summary>

private void DrawCityPoint()

{

setBarValue(0, Cities);

if (pictureBoxCity.InvokeRequired)

{

Action actionDrawCityPoint = delegate()

{

double fPicboxWidth = pictureBoxCity.Width;

double fPicboxheight = pictureBoxCity.Height;

double PicAxisX, PicAxisY;

PicAxisX = (double)(fPicboxWidth - 30) / (xMinMax[1] - xMinMax[0]);

PicAxisY = (double)(fPicboxheight - 30) / (yMinMax[1] - yMinMax[0]);

SolidBrush RedBrush = new SolidBrush(Color.Red);

pictureBoxCity.Image = null;

Image imageTSP = new Bitmap(pictureBoxCity.Width, pictureBoxCity.Height);

Graphics GraphicsTSP = Graphics.FromImage(imageTSP);

GraphicsTSP.Clear(Color.White);

double cityPointX;

double cityPointY;

for (int j = 0; j < Cities; j++)

{

cityPointX = (xy[j, 0] - xMinMax[0]) \* PicAxisX;

cityPointY = (xy[j, 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.FillEllipse(RedBrush, new RectangleF((float)cityPointX, (float)cityPointY, 3, 3));

UpadateProgressBar(j + 1);

}

pictureBoxCity.Image = imageTSP;

};

pictureBoxCity.Invoke(actionDrawCityPoint);

}

else

{

double fPicboxWidth = pictureBoxCity.Width;

double fPicboxheight = pictureBoxCity.Height;

double PicAxisX, PicAxisY;

PicAxisX = (double)(fPicboxWidth - 30) / (xMinMax[1] - xMinMax[0]);

PicAxisY = (double)(fPicboxheight - 30) / (yMinMax[1] - yMinMax[0]);

SolidBrush RedBrush = new SolidBrush(Color.Red);

pictureBoxCity.Image = null;

Image imageTSP = new Bitmap(pictureBoxCity.Width, pictureBoxCity.Height);

Graphics GraphicsTSP = Graphics.FromImage(imageTSP);

GraphicsTSP.Clear(Color.White);

double cityPointX;

double cityPointY;

for (int j = 0; j < Cities; j++)

{

cityPointX = (xy[j, 0] - xMinMax[0]) \* PicAxisX;

cityPointY = (xy[j, 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.FillEllipse(RedBrush, new RectangleF((float)cityPointX, (float)cityPointY, 3, 3));

UpadateProgressBar(j + 1);

}

pictureBoxCity.Image = imageTSP;

}

}

/// <summary>

///

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void exitToolStripMenuItem\_Click(object sender, EventArgs e)

{

Application.Exit();

}

/// <summary>

///

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void greedySolverToolStripMenuItem\_Click(object sender, EventArgs e)

{

labelProgress.Text = "Greedy algorithm progress";

shortesPathLenth = double.PositiveInfinity;

BeforeShortPathLegth = double.PositiveInfinity;

Thread threadGreedy = new Thread(new ThreadStart(GreedyAlgorithm));

threadGreedy.IsBackground = true;

threadGreedy.Start();

}

/// <summary>

///

/// </summary>

private void GreedyAlgorithm()

{

int[] GreedyhaveTrave = new int[Cities];

double cityShortestDistance = double.PositiveInfinity;

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

{

goodPathIndex[i] = 0;

CityTabu[i] = 0;

GreedyhaveTrave[i] = -1;

}

GreedyhaveTrave[0] = 0;

CityTabu[GreedyhaveTrave[0]] = 1;

int minCityIndex = GreedyhaveTrave[0];

setBarValue(1, Cities - 1);

for (int i = 1; i < Cities; i++)

{

cityShortestDistance = double.PositiveInfinity;

for (int j = 0; j < Cities; j++)

{

if (CityDistance[GreedyhaveTrave[i - 1], j] < cityShortestDistance && CityTabu[j] == 0)

{

minCityIndex = j;

cityShortestDistance = CityDistance[GreedyhaveTrave[i - 1], j];

}

}

UpadateProgressBar(i);

CityTabu[minCityIndex] = 1;

GreedyhaveTrave[i] = minCityIndex;

WriteProgressInfo(i, cityShortestDistance);

}

shortesPathLenth = 0;

for (int k = 0; k < Cities - 1; k++)

{

shortesPathLenth += CityDistance[GreedyhaveTrave[k], GreedyhaveTrave[k + 1]];

}

shortesPathLenth += CityDistance[GreedyhaveTrave[Cities - 1], GreedyhaveTrave[0]];

WriteProgressInfo(Cities, shortesPathLenth);

goodPathIndex = GreedyhaveTrave;

DrawShortestPath();

}

/// <summary>

///

/// </summary>

/// <param name="iterationCount"></param>

/// <param name="minimumLength"></param>

public void WriteProgressInfo(int iterationCount, double minimumLength)

{

double fImprovement = 0;

if (labelImprovement.InvokeRequired)

{

Action<int, double> actionIteration = delegate(int iterationCountTemp, double minimumLengthTemp)

{

fImprovement = BeforeShortPathLegth == double.PositiveInfinity ? 0 : BeforeShortPathLegth - minimumLengthTemp;

labelImprovement.Text = "(t=" + iterationCountTemp.ToString() + ")," + fImprovement.ToString("F0");

};

labelImprovement.Invoke(actionIteration, iterationCount, minimumLength);

}

else

{

fImprovement = BeforeShortPathLegth == double.PositiveInfinity ? 0 : BeforeShortPathLegth - minimumLength;

labelImprovement.Text = "(t=" + iterationCount.ToString() + ")," + fImprovement.ToString("F0");

}

if (labelCTLength.InvokeRequired)

{

Action<double> actionLength = delegate(double shortLengh)

{

labelCTLength.Text = shortLengh == 0 ? "" : shortLengh.ToString("F0");

};

labelCTLength.Invoke(actionLength, minimumLength);

}

else

{

labelCTLength.Text = minimumLength == 0 ? "" : minimumLength.ToString("F0");

}

}

/// <summary>

///

/// </summary>

public void DrawShortestPath()

{

if (pictureBoxCity.InvokeRequired)

{

Action actionDraw = delegate()

{

double fPicboxWidth = pictureBoxCity.Width;

double fPicboxheight = pictureBoxCity.Height;

double PicAxisX, PicAxisY;

PicAxisX = (double)(fPicboxWidth - 30) / (xMinMax[1] - xMinMax[0]);

PicAxisY = (double)(fPicboxheight - 30) / (yMinMax[1] - yMinMax[0]);

SolidBrush RedBrush = new SolidBrush(Color.Red);

pictureBoxCity.Image = null;

Image imageTSP = new Bitmap(pictureBoxCity.Width, pictureBoxCity.Height);

Graphics GraphicsTSP = Graphics.FromImage(imageTSP);

GraphicsTSP.Clear(Color.White);

double cityPointX;

double cityPointY;

setBarValue(0, Cities - 1);

Font myFont1 = new Font("Hacttenschweiler", 7);

for (int j = 0; j < Cities; j++)

{

cityPointX = (xy[j, 0] - xMinMax[0]) \* PicAxisX;

cityPointY = (xy[j, 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.FillEllipse(RedBrush, new RectangleF((float)cityPointX, (float)cityPointY, 3, 3));

UpadateProgressBar(j);

}

double FirstPointX, FirstPointY, SecondPointX, SecondPointY;

Pen penLine = new Pen(Color.Black, 1);

setBarValue(1, Cities - 1);

for (int i = 1; i < Cities; i++)

{

FirstPointX = (xy[goodPathIndex[i - 1], 0] - xMinMax[0]) \* PicAxisX;

FirstPointY = (xy[goodPathIndex[i - 1], 1] - yMinMax[0]) \* PicAxisY;

SecondPointX = (xy[goodPathIndex[i], 0] - xMinMax[0]) \* PicAxisX;

SecondPointY = (xy[goodPathIndex[i], 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.DrawLine(penLine, new PointF((float)FirstPointX, (float)FirstPointY), new PointF((float)SecondPointX, (float)SecondPointY));

UpadateProgressBar(i);

}

FirstPointX = (xy[goodPathIndex[0], 0] - xMinMax[0]) \* PicAxisX;

FirstPointY = (xy[goodPathIndex[0], 1] - yMinMax[0]) \* PicAxisY;

SecondPointX = (xy[goodPathIndex[Cities - 1], 0] - xMinMax[0]) \* PicAxisX;

SecondPointY = (xy[goodPathIndex[Cities - 1], 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.DrawLine(penLine, new PointF((float)FirstPointX, (float)FirstPointY), new PointF((float)SecondPointX, (float)SecondPointY));

pictureBoxCity.Image = imageTSP;

};

pictureBoxCity.Invoke(actionDraw);

}

else

{

double fPicboxWidth = pictureBoxCity.Width;

double fPicboxheight = pictureBoxCity.Height;

double PicAxisX, PicAxisY;

PicAxisX = (double)(fPicboxWidth - 30) / (xMinMax[1] - xMinMax[0]);

PicAxisY = (double)(fPicboxheight - 30) / (yMinMax[1] - yMinMax[0]);

SolidBrush RedBrush = new SolidBrush(Color.Red);

pictureBoxCity.Image = null;

Image imageTSP = new Bitmap(pictureBoxCity.Width, pictureBoxCity.Height);

Graphics GraphicsTSP = Graphics.FromImage(imageTSP);

GraphicsTSP.Clear(Color.White);

double cityPointX;

double cityPointY;

setBarValue(0, Cities - 1);

for (int j = 0; j < Cities; j++)

{

cityPointX = (xy[j, 0] - xMinMax[0]) \* PicAxisX;

cityPointY = (xy[j, 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.FillEllipse(RedBrush, new RectangleF((float)cityPointX, (float)cityPointY, 3, 3));

UpadateProgressBar(j);

}

double FirstPointX, FirstPointY, SecondPointX, SecondPointY;

Pen penLine = new Pen(Color.Black, 1);

setBarValue(1, Cities - 1);

for (int i = 1; i < Cities; i++)

{

FirstPointX = (xy[goodPathIndex[i - 1], 0] - xMinMax[0]) \* PicAxisX;

FirstPointY = (xy[goodPathIndex[i - 1], 1] - yMinMax[0]) \* PicAxisY;

SecondPointX = (xy[goodPathIndex[i], 0] - xMinMax[0]) \* PicAxisX;

SecondPointY = (xy[goodPathIndex[i], 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.DrawLine(penLine, new PointF((float)FirstPointX, (float)FirstPointY), new PointF((float)SecondPointX, (float)SecondPointY));

UpadateProgressBar(i);

}

FirstPointX = (xy[goodPathIndex[0], 0] - xMinMax[0]) \* PicAxisX;

FirstPointY = (xy[goodPathIndex[0], 1] - yMinMax[0]) \* PicAxisY;

SecondPointX = (xy[goodPathIndex[Cities - 1], 0] - xMinMax[0]) \* PicAxisX;

SecondPointY = (xy[goodPathIndex[Cities - 1], 1] - yMinMax[0]) \* PicAxisY;

GraphicsTSP.DrawLine(penLine, new PointF((float)FirstPointX, (float)FirstPointY), new PointF((float)SecondPointX, (float)SecondPointY));

pictureBoxCity.Image = imageTSP;

}

}

private void aCOSolverToolStripMenuItem\_Click(object sender, EventArgs e)

{

labelProgress.Text = "ACO algorithm progress";

shortesPathLenth = double.PositiveInfinity;

BeforeShortPathLegth = double.PositiveInfinity;

Nc = 0;

Thread threadACO = new Thread(new ThreadStart(ACOAlgrorithm));

threadACO.IsBackground = true;

threadACO.Start();

}

/// <summary>

///

/// </summary>

private void ACOAlgrorithm()

{

AntCount = Cities;

PheromoneTau = new double[Cities, Cities];

etaCij = new double[Cities, Cities];

deltaTau = new double[Cities, Cities];

CitiesTabu = new int[AntCount, Cities];

CityNoTabu = new int[AntCount, Cities];

CityAntHaveTrave = new int[AntCount, Cities];

mAntShortestDistance = new double[AntCount];

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

{

for (int j = 0; j < Cities; j++)

{

PheromoneTau[i, j] = 1.0;

if (i != j)

{

etaCij[i, j] = 1.0 / CityDistance[i, j];

}

deltaTau[i, j] = 0.0;

}

}

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

{

for (int j = 0; j < Cities; j++)

{

CityAntHaveTrave[i, j] = -1;

CitiesTabu[i, j] = 0;

}

CityAntHaveTrave[i, 0] = 0;

CitiesTabu[i, 0] = 1;

}

for (int iterationCount = 1; iterationCount <= NcMax; iterationCount++)

{

startAntMove();

int minLenthAntNo = 0;

BeforeShortPathLegth = shortesPathLenth;

setBarValue(0, AntCount);

for (int antNo = 0; antNo < AntCount; antNo++)

{

mAntShortestDistance[antNo] = calculateMinDistance(antNo);

if (mAntShortestDistance[antNo] < shortesPathLenth)

{

shortesPathLenth = mAntShortestDistance[antNo];

minLenthAntNo = antNo;

}

UpadateProgressBar(antNo + 1);

}

setBarValue(0, Cities);

for (int cityNO = 0; cityNO < Cities; cityNO++)

{

goodPathIndex[cityNO] = CityAntHaveTrave[minLenthAntNo, cityNO];

UpadateProgressBar(cityNO + 1);

}

UpdatePathIncrementPheromone();

setBarValue(0, Cities);

for (int icityNo = 0; icityNo < Cities; icityNo++)

{

for (int jcityNo = 0; jcityNo < Cities; jcityNo++)

{

PheromoneTau[icityNo, jcityNo] = (1 - rho) \* PheromoneTau[icityNo, jcityNo] + deltaTau[icityNo, jcityNo];

if (PheromoneTau[icityNo, jcityNo] < PhenomenonZero)

PheromoneTau[icityNo, jcityNo] = PhenomenonZero;

if (PheromoneTau[icityNo, jcityNo] > PhenomenonRangeMax)

PheromoneTau[icityNo, jcityNo] = PhenomenonRangeMax;

}

UpadateProgressBar(icityNo + 1);

}

setBarValue(0, AntCount);

for (int antNo = 0; antNo < AntCount; antNo++)

{

for (int cityNo = 0; cityNo < Cities; cityNo++)

{

CitiesTabu[antNo, cityNo] = 0;

if (cityNo > 0)

{

CityAntHaveTrave[antNo, cityNo] = -1;

}

}

CitiesTabu[antNo, CityAntHaveTrave[antNo, 0]] = 1;

UpadateProgressBar(antNo + 1);

}

WriteProgressInfo(iterationCount, shortesPathLenth);

DrawShortestPath();

}

}

/// <summary>

///

/// </summary>

private void startAntMove()

{

double antPij = 0;

double PijSum = 0;

double randomPij = 0;

Random randData = new Random((int)DateTime.Now.Ticks & 0x0000FFFF);

for (int haveTraveCity = 1; haveTraveCity < Cities; haveTraveCity++)

{

setBarValue(0, AntCount);

for (int iAnt = 0; iAnt < AntCount; iAnt++)

{

antPij = 0.0;

PijSum = 0.0;

randomPij = randData.Next(3000) / 3000.0;

for (int iCity = 0; iCity < Cities; iCity++)

{

if (CitiesTabu[iAnt, iCity] == 0)

{

try

{

PijSum += Math.Pow(PheromoneTau[CityAntHaveTrave[iAnt, haveTraveCity - 1], iCity], alpha) \* Math.Pow(etaCij[CityAntHaveTrave[iAnt, haveTraveCity - 1], iCity], beta);

}

catch (System.Exception ex)

{

continue;

}

}

}

for (int iCity = 0; iCity < Cities; iCity++)

{

if (CitiesTabu[iAnt, iCity] == 0)

{

try

{

antPij += Math.Pow(PheromoneTau[CityAntHaveTrave[iAnt, haveTraveCity - 1], iCity], alpha) \* Math.Pow(etaCij[CityAntHaveTrave[iAnt, haveTraveCity - 1], iCity], beta) / PijSum;

if (antPij > randomPij)

{

CitiesTabu[iAnt, iCity] = 1;

CityAntHaveTrave[iAnt, haveTraveCity] = iCity;

break;

}

}

catch (System.Exception ex)

{

continue;

}

}

}

UpadateProgressBar(iAnt + 1);

}

}

}

/// <summary>

///

/// </summary>

private void UpdatePathIncrementPheromone()

{

setBarValue(0, AntCount);

for (int antNo = 0; antNo < AntCount; antNo++)

{

try

{

for (int cityNo = 0; cityNo < Cities - 1; cityNo++)

{

deltaTau[CityAntHaveTrave[antNo, cityNo], CityAntHaveTrave[antNo, cityNo + 1]] += 100 / mAntShortestDistance[antNo];

}

deltaTau[CityAntHaveTrave[antNo, Cities - 1], CityAntHaveTrave[antNo, 0]] += 100 / mAntShortestDistance[antNo];

}

catch (System.Exception ex)

{

continue;

}

UpadateProgressBar(antNo + 1);

}

}

/// <summary>

///

/// </summary>

/// <param name="antNo"></param>

/// <returns></returns>

private double calculateMinDistance(int antNo)

{

double currentMinDis = 0;

setBarValue(0, Cities - 1);

for (int cityNO = 0; cityNO < Cities - 1; cityNO++)

{

UpadateProgressBar(cityNO + 1);

currentMinDis += CityDistance[CityAntHaveTrave[antNo, cityNO], CityAntHaveTrave[antNo, cityNO + 1]];

}

currentMinDis += CityDistance[CityAntHaveTrave[antNo, Cities - 1], CityAntHaveTrave[antNo, 0]];

return currentMinDis;

}

}

}

