using System;

using System.Collections.Generic;

using System.Text;

namespace AntSystem

{

public class AA

{

private int alpha; 对信息量的重视程度

private int beta; 启发式信息的受重视程度

private double lo; 信息素的挥发速度

private double[,,]City; 城市距离矩阵

private double[,,] Message; 信息素矩阵

private Queue<int> openList=new **Queue**<int> ();opneList用于存放下一步可行城市

private Queue<int> closedList=new Queue<int> ();closedList用于存放已经访问过的城市

private Queue <int> BestList=new Queue<int> ();储存较好的路径

private int Pro\_time = 0;

/\*\*//////////////////////////////////////////////////////////

/// 构造函数形成城市距离和信息素矩阵

/// <param name="city">城市距离矩阵</param>

/// <param name="Lo"> 信息素的挥发速度</param>

public AA(double[,,] city,double Lo,int Alpha,int Beta)

{

alpha = Alpha;

beta = Beta;

lo=Lo;

int temp = Convert.ToInt32( Math.Sqrt(city.Length/3));

City=new double [temp,temp,3];

Message=new double [temp,temp,3];

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

{

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

{

for(int k = 0; k < 3; k++)

{

City[i, j,3] = city[i, j,3];

}

}

}

//初始化信息素矩阵

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

{

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

{

if (i != j)

{

for(int k = 0; k < 3; k++)

{

**Message[i, j,k]** = (double)1 / (temp \* temp - temp);

}

}

}

}

}

/\*\*/////////////////////////////////////////////////////////////

/// 改变信息素矩阵，closed\_list为a较好的路径

/// <param name="closed\_list"></param>

private void Change\_Message(Queue<int> closed\_list)

{

lock (this)

{

int[] temp\_Array = new int[closed\_list.Count];

temp\_Array = closed\_list.ToArray();

for (int i = 0; i < closed\_list.Count - 1; i++)

{

for(int k = 0; k < 3; k++)

{

Message[temp\_Array[i], temp\_Array[i + 1],k] = Message[temp\_Array[i], temp\_Array[i + 1],k] + lo / ((1 - lo) \*Convert.ToInt32(Get\_Weight(closed\_list)+1));

}

}

for(int k = 0; k < 3; k++)

{

Message[temp\_Array[temp\_Array.Length - 1], temp\_Array[0],k] = Message[temp\_Array[temp\_Array.Length – 1], temp\_Array[0],k] + lo / ((1 - lo) \*Convert.ToInt32(Get\_Weight(closed\_list)));

}

for (int i = 0; i < closed\_list.Count; i++)

{

for (int j = 0; j < closed\_list.Count; j++)

{

for(int k = 0; k < 3; k++)

{

Message[i, j] = (1 - lo) \* Message[i, j];

}

}

}

}

}

/\*\*////////////////////////////////////////////////////////////////

/// 输入一个链表，计算出其对应的总路径?

/// <param name="closed\_list"></param>

public double Get\_Weight(Queue <int> closed\_list)

{

lock (this)

{

double sum = 0;

int[] temp\_Array = new int[closed\_list.Count];

temp\_Array = closed\_list.ToArray();

for (int i = 0; i < Convert.ToInt32(temp\_Array.Length) - 1; i++)

{

for(int k = 0; k < 3; k++)

{

sum = sum + City[temp\_Array[i], temp\_Array[i + 1],k];

}

}

for(int k = 0; k < 3; k++)

{

sum = sum + City[temp\_Array[temp\_Array.Length - 1], temp\_Array[0],k];

}

return sum;

}

}

/\*\*///////////////////////////////////////////////////////////////

/// 产生到城市后下一个可走城市的集合。并将城市编号加入到openList中

/// 产生的城市不可以已经存在closedList中

/// <param name="i"></param>

private void NextCity()

{

openList.Clear();

int temp\_int=Convert.ToInt32(Math.Sqrt(City.Length));

for (int i = 0; i < temp\_int; i++)

{

if (closedList.Contains(i) ==false)

{

openList.Enqueue(i);

}

}

}

/\*\*///////////////////////////////////////////////////////////////

/// 选择应该走那条¬路，选择完路后，清空openList，再把加入到openList中

/// <returns></returns>

private int choiceRoute()

{

int index = 0;//记录选择的城市

Random random = new Random();

double random\_value =(double) random.NextDouble();//随机选择的概率

int[] temp\_Array=new int [openList.Count];

temp\_Array=openList.ToArray();

double sum\_Message = 0;//openList所有节点的总信息量

for (int i = 0; i < openList.Count; i++)

{

for(int k = 0; k < 3; k++)

{

double eta = 1 / City[Pro\_time, temp\_Array[i],k];

sum\_Message = sum\_Message +Math.Pow(Message[Pro\_time, temp\_Array[i],k],alpha)\*Math.Pow (eta,beta);

}

}

double temp=0;

for (int j = 0; j < openList.Count; j++)

{

for(int k = 0; k < 3; k++)

{

double eta = 1 / City[Pro\_time, temp\_Array[j],k];

temp=temp+Math.Pow(Message[Pro\_time,temp\_Array[j],k],alpha)\*Math.Pow(eta,beta)/sum\_Message;

if (temp > random\_value)

{

index = temp\_Array [j];

break;

}

}

openList.Clear();

openList.Enqueue(index);

return index;

}

/\*\*//////////////////////////////////////////////////////////////

public Queue<int> Main\_DW()

{

BestList.Clear();

/\*\*////共循环20次

for (int i = 0; i < 20; i++)

{

/\*\*////共有n只蚂蚁n=City'number Convert.ToInt32(Math.Sqrt(City.Length))

for (int j = 0; j < Convert.ToInt32(Math.Sqrt(City.Length/3)); j++)

{

openList.Enqueue(0);

closedList.Clear();

while (openList.Count != 0 && closedList.Count != Convert.ToInt32(Math.Sqrt(City.Length/3)))

{

int temp = openList.Dequeue();

Pro\_time = temp;

closedList.Enqueue(temp);

if (openList.Count == 0 && closedList.Count == Convert.ToInt32(Math.Sqrt(City.Length)))

{

if (BestList.Count == 0)

{

int[] temp\_Array = new int[Convert.ToInt32(Math.Sqrt(City.Length/3))];

temp\_Array = closedList.ToArray();

for (int k = 0; k < Convert.ToInt32(Math.Sqrt(City.Length/3)); k++)

{

BestList.Enqueue(temp\_Array[k]);

}

}

if (Get\_Weight(BestList) > Get\_Weight(closedList))

{

BestList.Clear();

int[] temp\_Array = new int[Convert.ToInt32(Math.Sqrt(City.Length/3))];

temp\_Array = closedList.ToArray();

for (int k = 0; k < Convert.ToInt32(Math.Sqrt(City.Length/3)); k++)

{

BestList.Enqueue(temp\_Array[k]);

}

}

}

NextCity();

choiceRoute();

}

}

Change\_Message(BestList);//修改信息量

}

return BestList;

}

}

}