TRAVELING SALESMAN PROBLEM USING ANT COLONY OPTIMIZATION using new parameter

#include <vector>

#include <conio.h>

#include <cstdio>

#include <ctime>

#include <cmath>

#include <algorithm>

#include <set>

#include <time.h>

#include <iostream>

using namespace std;

class Data {

public:

vector< pair<double,double> > cityCoords;

vector< vector<double> > cost;

vector< vector<double> > visibility;

vector< vector<double> > T;

int N;

char s[30];

Data() { // N cities

scanf("%s",s);

scanf("%d",&N);

// index from 1

cityCoords.push\_back(make\_pair(0,0));

vector< double> L,M,U;

cost.push\_back(L);

T.push\_back(M);

visibility.push\_back(U);

for(int i=0;i<N;i++) {

double x,y;

scanf("%lf%lf",&x,&y);

cityCoords.push\_back(make\_pair(x,y));

}

for(int i=0;i<N;i++) {

vector<double> V(N+1);

vector<double> t(N+1);

vector<double> l(N+1);

for(int j = 1; j<=N ; j++) {

scanf("%lf",&V[j]);

t[j] = 1.0;

if(V[j] != 0) {

l[j] = 1/ V[j];

}

}

cost.push\_back(V);

T.push\_back(t);

visibility.push\_back(l);

}

}

double tourCost(vector<int> C) {

int l = C.size();

double tourCost = 0.0;

l = l-1;

for (int i=0;i <l; i++) {

tourCost += cost[C[i]][C[i+1]];

}

tourCost += cost[C[l]][C[0]];

return tourCost;

}

void print(vector<int> C) {

for(int i=0;i<C.size();i++)

printf("%d ",C[i]);

printf("\n");

}

};

Data d; // global data construct

class Ant {

public:

vector<int> trail;

set<int> available;

double alpha; // importance of the pheromone level

double beta; // importance of the visibility

Ant(double a,double b) {

alpha = a;

beta = b;

trail.push\_back(1); // always start from the nest (1)

for(int i=2;i<=d.N;i++) {

available.insert(i);

}

}

void reset() {

vector<int> L;

L.push\_back(1);

trail = L; // reset to nest.

for(int i=2;i<=d.N;i++) {

available.insert(i);

}

}

void deposit() {

double tourCost = d.tourCost(trail);

int Q = 100;

double depositAmount = Q / tourCost;

int l = trail.size();

l = l-1;

for (int i=0;i <l; i++) {

d.T[trail[i]][trail[i+1]] += depositAmount;

}

d.T[trail[l]][trail[0]] += depositAmount;

}

vector<int> stop() {

deposit();

vector<int> temp = trail;

reset();

return temp;

}

void step() {

int currentCity = trail[trail.size()-1];

double norm = probabilityNorm(currentCity);

double p,gp;

bool moved = false;

double highestProb = 0;

double cityHighest = 0;

for(set<int>::iterator i=available.begin(); i != available.end() ; i++) {

p = moveProbability(currentCity,\*i,norm);

if (p > highestProb) {

cityHighest = \*i;

highestProb = p;

}

gp = getRand();

if (gp <= p) { // move

moved = true;

trail.push\_back(\*i);

available.erase(i);

break;

}

}

if(!moved) {

// make a move to the highest available prob city

// move to cityHighest

trail.push\_back(cityHighest);

available.erase(cityHighest);

}

}

double getRand() {

double p = (rand() / (RAND\_MAX + 1.0));

return p;

}

double moveProbability(int i,int j,double norm) {

double p = (pow(d.T[i][j],alpha))\*(pow(d.visibility[i][j],beta));

p /= norm;

return p;

}

double probabilityNorm(int currentCity) {

int size = available.size();

double norm = 0.0;

for(set<int>::iterator i=available.begin(); i != available.end() ; i++) {

norm += (pow(d.T[currentCity][\*i],alpha))\*(pow(d.visibility[currentCity][\*i],beta));

}

return norm;

}

};

class ACO {

public:

int N; // cities

int M; // no.of ants

vector<Ant> ant; // ants

double evaporation; // evaporation rate

double alpha; // importance of the pheromone level

double beta; // importance of the visibility

ACO(double a,double b,double e) {

alpha = a;

beta = b;

evaporation = e;

N = d.N;

M = 30; // ants

for(int i=0; i<M; i++) {

Ant a(alpha,beta);

ant.push\_back(a);

}

}

void run() {

vector<int> PATH;

double minTour,tourC;

for(int n=0;n < 30; n++) {

for(int p=0; p<(N-1); p++){

for(int i=0;i<M;i++) {

ant[i].step();

}

}

for(int i=0;i<M;i++) {

vector<int> p = ant[i].stop();

if(!PATH.size()) {

PATH = p;

minTour = d.tourCost(p);

continue;

}

tourC = d.tourCost(p);

if(tourC < minTour) {

minTour = tourC;

PATH = p;

}

}

for(int i=1;i<=N;i++) {

for(int j=1;j<=N;j++) {

d.T[i][j] \*= evaporation;

}

}

}

printf("%lf\n",minTour);

d.print(PATH);

}

};

main() {

time\_t t;

srand((unsigned)time(&t));

double alpha = 1; // pheromone importance

double beta = 2; // visibility importance

double evaporation = 0.1 ; // evaporation rate

ACO colony(alpha,beta,evaporation);

colony.run();

getch();

}