# 基于蚁群优化求解TSP问题

**摘要：**本文实现了基于蚁群优化的TSP问题求解，并分析了对于不同挥发因子对算法的影响，此外，本文还实现了min-max优化的蚁群系统，并分析在不同挥发因子的情况下，优化的效果。

**Abstract：**This paper implemented ASO algorithm and applied it to the TSP problems. We

analyze the influence of different evaporation factors to the algorithm. And also, we have an implementation using the max-min technique. We then analyze the efficiency of the technique when factor changed.

## 1,引言：

虽然已经有很多经典方法求解组合和优化问题，但是有一大类问题还是不能通过这些方法在多项式问题求解，即它们是难问题。旅行商问题就是一个典型的例子。旅行商问题可以被建模成一个图，图中的点代表城市，边代表城市间的距离。现代启发式算法可以很快地求解这一类组合优化问题。虽然这样地算法可能求出的最优解与真实最优解有一定误差，但通常是可接受的。

启发式算法自出现以来就受到很多研究者的注意，在过去的几十年里，研究者提出了很多启发式算法。如蚁群优化[1]，模拟退火[2]，遗传算法[3]等等。这些算法对于求解不同的优化问题各有利弊，针对特定问题需要做一定的优化。本文通过实现基于蚁群优化求解TSP问题，探讨蚁群优化算法参数的选取以及一些优化。

本文的结构如下，第2章我们将简介TSP问题，第3章将简介TSP算法，第4章将介绍我们的实现细节，第5章将给出实验结果以及对比，第6章将介绍max-min优化以及实验结果，第7章是本文的结论。

### 2,问题描述

旅行商问题[4]：一个商人欲到n个城市推销商品，每两个城市间距离为，选择一条路径使得商人路过每个城市并回到起点距离最短。对于旅行商问题又可以进一步分为：对称；旅行商问题，非对称旅行商问题。这两者的差别在于相应两城市距离是否与方向有关。旅行商问题具有重要的实际意义和工程背景。它一开始是为交通运输而提出的，比如飞机航线安排、送邮件、快递服务、设计校车行进路线等等。可以将旅行商问题按如下的数学模型表述：

(1)

s.t.  (2)

(3)

(4)

(5)

(5)中的决策变量表示商人行走路径中包含，表示商人没有走这条路径。（1）表示最优路径是使得距离之和最小的路径。（2）限制了商人只能从一个城市出来一次，（3）限制了商人只能进入一个城市一次。限制（4）保证了整个解没有子回路产生（即长度小于n的回路）。

### 3,算法描述

蚁群优化是一种模拟蚂蚁需找食物的算法。它的基本思想是：对于一个目标点，如果路径较短，则蚂蚁到这里就比较快，于是单位时间通过的蚂蚁较多，留下的信息素就比较多，而对于路径较长的，蚂蚁到这里就比较慢，信息素挥发比较快，于是随着时间的推进，最短路劲上的信息素就会越来越强，于是更多的蚂蚁到这条路径。

蚁群优化的算法具体过程如下：

Step1：对n城市的TSP问题， ， ，城市间的距离 ，为TSP图中的每一条弧 赋信息素痕迹初值 ,假设有m只蚂蚁在工作，所有蚂蚁从同一城市出发。。当前最优解为。

Step2：如果满足算法的停止规则，停止计算并输出计算得到的最好解。否则，让蚂蚁s从起点出发，用表示蚂蚁s行走的城市集合，初始为空集，。

Step3：按蚂蚁 的顺序分别计算。当蚂蚁在城市,若或完成第s只蚂蚁的计算。否则，若且,则以概率

（6）

到达，;

若且，到达,;重复本步骤。

Step4：对，若，按中城市的顺序计算路径长度；

若，路径长度是一个充分大的数。比较m只蚂蚁中的路径长度，记走最短路径的蚂蚁为t若，则。用（2）对W路径上的弧信息素痕迹加强，对其他弧的信息素痕迹挥发，

（7）

得到新的,重复step2。

其中

4,算法实现

我们将城市编号,我们用一个向量s表示解,设s=表示最短路径为。我们的实现选择n-1只蚂蚁，即只有n-1个向量，并且运行过程中蚂蚁数量恒定。当蚂蚁到了第j点时，假设前面已经选了m个点，则我们设置一个tabu表，将已经路过的点禁忌掉，这就避免了回路（注意，到了第j点的时候就需要将j加入到tabu中）。我们使用随机数生成函数randV()(实现参考后面的代码)生成的实数p,按照这个数的大小去选择下一个点，我们将j的所有可行的下一个城市的信息素求和得到每个城市的访问概率，同时将它们按照一定顺序排列起来，按照它们的概率组成[0,1]的区间，当p落到哪个点，就选哪个点。

这样继续下去，就能求出一个解。重复上述步骤n-1次，我们就得到了一个S，S是所有向量s的集合，从中选出路径最短的，按照公式（7）更新。这里我们需要选取一个挥发因子，具体细节见第5章。我们将这个最短路径记录下来，与上一次结果做对比，如果相等，则相等计数count+1，若不相等，则count计数归零。直到若干次之后，count叨叨我们预设的一个值cons。

### 5,实验结果

我们选取固定的,并使用count作为停机条件，即当最短路径的值在count次不再变化停止计算。我们将我们的算法在[5]上的数据集pr107上实验。我们的实现选取了下表的参数，得到结果如下：

Table1,最短路径

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| Count=3 | 248455 | 170254 | 97165.5 | 71241.5 | 59071.9 | 56572.8 |
| Count=5 | 267958 | 151915 | 110433 | 76054.6 | 60403.6 | 54016.9 |
| Count=7 | 288162 | 148372 | 108763 | 78797.8 | 60899.8 | 56962.9 |

Table2,计算时间

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| Count=3 | 2074ms | 5380ms | 14428ms | 52359ms | 194497ms | 744555ms |
| Count=5 | 1795ms | 5551ms | 14838ms | 45962ms | 226713ms | 570979ms |
| Count=7 | 1985ms | 6463ms | 16755ms | 40516ms | 194172ms | 627243ms |

经过表格1,2我们容易观察到，对于来说，当取得比较大的时候，算法收敛比较快，但是得到的解质量比较差，这是因为当较大的时候，其他路径的信息素挥发过快，使得开始选取的那条路径被选中的概率增大，但是取得比较小的时候，算法收敛比较慢，解较好，但是时间消耗很大。同时，经过对比，count值对于算法的贡献比较小，而且几乎不影响计算时间。

### 6,max-min优化及其效果分析

Stützle , Hoos [5]提出MAX-MIN算法，旨在控制信息素痕迹的挥发，具体公式如下：

其中实数。

为了减少信息素增强的幅度，我们对于最优路径上面的信息素也做限定，具体参数为最大值不超过0.6，最小值不低于0.000000003：

下面是相应的实验结果（注，这些结果是多次实验的均值，实验时会有一些误差，主要原因在于蚂蚁选择路径的随机性）。

|  |  |  |  |
| --- | --- | --- | --- |
|  | ACO | MAX-MIN ACO | 提升比率 |
| =0.5 | 293887 | 199144 | 147% |
| =0.2 | 151895 | 136435 | 111% |
| =0.1 | 110636 | 92656 | 119% |
| =0.05 | 79969 | 74980 | 107% |
| =0.02 | 60451 | 61682 | 98% |

我们可以看出来，当比较大的时候，此时最短路径很快就收敛了，容易陷入局部最优，我们通过MAX-MIN算法可以有很大的提高，但是当逐渐减小的时候，MAX-MIN算法对于解的质量提高的效果逐渐减弱。

### 7,总结

我们实现了基于蚁群优化求解TSP的算法，并通过调整参数来观察大小对于解的性能的影响，我们发现当挥发较慢的时候，能够避免过早陷入局部最优，但是会造成算法耗时过多。由于只有一次循环的最优路径被增强，使得算法过快陷入局部最优，我们使用MAX-MIN来改善这个问题。经过实验得出，MAX-MIN对于原来的解有的提高，但是当逐渐减小的时候，这种提高不明显甚至比原来的解还要差。后续的工作将专注于提高求解的速度和利用其他算法来改善蚁群优化中最优路径增强导致的局部最优以及研究如何选取合适的MAX-MIN参数。

### 8,参考文献

[1] M. Dorigo, C. Blum, Ant colony optimization theory: a survey, Theoretical Computer Science 344 (2005) 243–278.

[2]S.Kirkpatrick, C.Gelatt, M.Vecchi, Optimization by simulated annealing, Science 220(1983)671-680.

[3] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, USA, 1975.

[4]邢文训，谢金星，现代优化计算方法，清华大学出版社（第二版），（2006），p2.

[5] Stützle T, Hoos HH. MAX-MIN ant system. Future Generation Computing Systems, 2000,16:889-914.

[6] <http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/>

### 9,C++代码：

#include<iostream>

#include<vector>

#include<random>

#include<string>

#include<math.h>

#include<sys/time.h>

#include<cstdlib>

#include<fstream>

#define MAX 165332

using namespace std;

typedef struct P{

double possibility;

int city;

}P;

double power2(double x){

return x\*x;

}

int strToInt(string s){

int sum=0;

for(int i=0;i!=s.length();i++){

sum\*=10;

sum+=(s[i]-'0');

}

return sum;

}

double randV(){

default\_random\_engine generator(random\_device{}());

uniform\_real\_distribution<double> fun(0.0,1.0);

return fun(generator);

}

void update(double\* phrom,int \*path,double ro,int min,int dim){

double\* tmp;

tmp=new double [dim\*dim];

for(int i=0;i!=dim;i++)

for(int j=0;j!=dim;j++)

tmp[i\*dim+j]=phrom[i\*dim+j];

for(int i=0;i!=dim;i++)

for(int j=0;j!=dim;j++){

phrom[i\*dim+j]=(1-ro)\*tmp[i\*dim+j];

}

for(int i=0;i!=dim;i++){

phrom[i\*dim+i]=0;

}

for(int t=1;t!=dim+1;t++){

int i=path[t-1];

int j=path[t];

phrom[i\*dim+j]=(1-ro)\*tmp[i\*dim+j]+ro/min;

}

}

void update(double\* phrom,int \*path,double ro,int min,int dim,double maxP,double minP){

double\* tmp;

tmp=new double [dim\*dim];

for(int i=0;i!=dim;i++)

for(int j=0;j!=dim;j++)

tmp[i\*dim+j]=phrom[i\*dim+j];

for(int i=0;i!=dim;i++)

for(int j=0;j!=dim;j++){

phrom[i\*dim+j]=(1-ro)\*tmp[i\*dim+j];

if(phrom[i\*dim+j]<minP){

//cout<<"po";

phrom[i\*dim+j]=minP;

}

}

for(int i=0;i!=dim;i++){

phrom[i\*dim+i]=0;

}

for(int t=1;t!=dim+1;t++){

int i=path[t-1];

int j=path[t];

phrom[i\*dim+j]=(1-ro)\*tmp[i\*dim+j]+ro/min;

if(phrom[i\*dim+j]>maxP){

cout<<"oops";

phrom[i\*dim+j]=maxP;

}

}

}

void init(double \*mat,double \*dis,int dim){

for(int i=0;i!=dim;i++)

for(int j=0;j!=dim;j++)

dis[i\*dim+j]=mat[i\*dim+j];

}

void init(double \*dis,int dim,ifstream &file){

string s;

for(int i=0;i!=18;i++){

file>>s;

}

int\* city;

city=new int[dim\*3];

for(int i=0;i!=dim;i++){

for(int j=0;j!=3;j++){

file>>s;

city[i\*3+j]=strToInt(s);

}

}

for(int i=0;i!=dim;i++){

for(int j=i;j!=dim;j++){

double tmp=power2(city[i\*3+1]-city[j\*3+1])+power2(city[i\*3+2]-city[j\*3+2]);

dis[i\*dim+j]=sqrt(tmp);

dis[j\*dim+i]=sqrt(tmp);

}

}

for(int i=0;i!=dim;i++)

dis[i\*dim+i]=0;

delete [] city;

}

double maxColSum(double \*dis,int dim){

double maxSum=0;

for(int i=0;i!=dim;i++){

double max=0;

for(int j=0;j!=dim;j++){

if(dis[i\*dim+j]>max){

max=dis[i\*dim+j];

}

}

maxSum+=max;

}

return maxSum;

}

void redis(double\* dis,int dim,int maxSum){

for(int i=0;i!=dim;i++)

for(int j=0;j!=dim;j++){

if(i!=j&&dis[i\*dim+j]==0)

dis[i\*dim+j]=maxSum;

}

}

void printM(double \*dis,int dim){

for(int i=0;i!=dim;i++){

for(int j=0;j!=dim;j++){

cout<<dis[i\*dim+j]<<" ";

}

cout<<endl;

}

}

void printV(int \*vec,int dim){

for(int i=0;i!=dim;i++)

cout<<vec[i]<<" ";

cout<<endl;

}

int main(int argc,char \*argv[]){

ifstream file("pr107.tsp");

double\* dis;

double\* phrom;

P\* prob;

int dim=107;

int\* tabu;

vector<int\*> solution;

dis=new double [dim\*dim];

//init(mat,dis,dim);

init(dis,dim,file);

file.close();

tabu=new int [dim];

phrom=new double [dim\*dim];

prob=new P [(dim-1)];

double initV=1./(dim\*(dim-1));

double ro=atof(argv[1]);

int cons=atoi(argv[2]);

// cout<<"ro:"<<ro<<" cons:"<<cons<<" ";

for(int i=0;i!=dim;i++){

for(int j=0;j!=dim;j++){

phrom[i\*dim+j]=initV;

}

}

double maxSum=maxColSum(dis,dim);

//cout<<maxSum<<endl;

redis(dis,dim,maxSum);

// printM(dis,dim);

bool flag=true;

double L=maxSum;

int\* staticSolve;

staticSolve=new int [dim+1];

int count=0;

struct timeval t\_end,t\_start;

// cout<<"begin"<<endl;

gettimeofday(&t\_start,NULL);

while(flag){

//cout<<"begin"<<endl;

double\* length;

length=new double [dim];

for(int i=0;i!=dim;i++)

length[i]=0.0;

for(int i=0;i!=dim-1;i++){

int\* solve;

solve=new int[dim+1];

for(int i=0;i!=dim;i++)

tabu[i]=0;

solve[0]=0;

tabu[0]=1;

for(int j=1;j!=dim;j++){

double sigmaPhrom=0.0;//

for(int t=0;t!=dim;t++){

if(tabu[t]!=1){

sigmaPhrom+=phrom[solve[j-1]\*dim+t];

}

}

int s=0;

for(int t=0;t!=dim;t++){

if(tabu[t]!=1){

if(s==0){

prob[s].possibility=phrom[solve[j-1]\*dim+t]/sigmaPhrom;

}else{

prob[s].possibility=prob[s-1].possibility+phrom[solve[j-1]\*dim+t]/sigmaPhrom;

}

prob[s].city=t;

s++;

}

}

double p=randV();

for(int k=0;k!=s;k++){//no repeat

if(p<=prob[k].possibility){

solve[j]=prob[k].city;

tabu[prob[k].city]=1;

break;

}

}

length[i]+=dis[solve[j-1]\*dim+solve[j]];

//cout<<"tabu: ";

//printV(tabu,dim);

}

solve[dim]=0;

length[i]+=dis[solve[dim-1]\*dim];

//printV(solve,dim+1);

solution.push\_back(solve);

}//generate dim-1 ant and its route and related length

double min=maxSum;

int pos=0;

for(int i=0;i!=dim-1;i++){

if(length[i]<min){

pos=i;

min=length[i];

}

}

/\* cout<<"solve:";

for(int i=0;i!=dim+1;i++){

cout<<solution[pos][i]<<" ";

}

cout<<endl;

cout<<"min"<<min<<" "<<maxSum<<endl;\*/

if(atoi(argv[3])==1){

update(phrom,solution[pos],ro,min,dim,0.6,0.000000003);

}else{

update(phrom,solution[pos],ro,min,dim);//0.6,0.000000003);

}

if(L-min==0){

count++;

}else{

count=0;

}

if(count>=cons){

flag=false;

}

L=min;

//cout<<L<<endl;

/\*gabage reuse\*/

for(vector<int\*>::iterator it=solution.begin();it!=solution.end();it++){

delete [] \*it;

}

solution.clear();

delete [] length;

}

gettimeofday(&t\_end,NULL);

int timeuse=1000000\*(t\_end.tv\_sec-t\_start.tv\_sec)+t\_end.tv\_usec-t\_start.tv\_usec;

cout<<"ro: "<<ro<<" cons:"<<cons;

cout<<" totalLength:"<<L<<" timeConsum:"<<timeuse<<"us"<<endl;

return 0;

}