

## TSP random tour generation algorithm

Random tour generation is very important part of the Cross Entropy algorithm (Reuven Y. Rubinshtein and Dirk P. Kroese “The Cross Entropy method” ) for solving combinatorial optimization problems. During the algorithm we must generate many tours, but the trouble is that it is a time consuming process that slows the algorithm considerably. The generation algorithms that are given in the book takes  $O(n^3)$  where  $n$  is a number of nodes. We will show that one can improve those algorithms to  $O(n^2)$ .

$P[n \times n]$  – probability matrix where  $p_{i,j}$  is a probability to travel from node  $i$  to node  $j$ .

$S[n]$  – visiting array ( $S[i]=1$  if we have already visited in node  $i$ ,  $S[i]=0$  otherwise ).

Norm – normalization factor.

$v$  - current node index.

$Res[n+1]$  – result array.

### The algorithm:

1.  $t=1$ ,  $S[1]=1$ , for all  $i \neq 1$   $S[i]=0$ ,  $v=1$ ,  $Res[1]=1$   $Res[n+1]=1$ .
2. generate random variable  $u \sim U(0,1)$ .
3.  $Norm = \sum_{i=1}^n (1-S[i]) * p_{v,i}$ .
4.  $u=u*Norm$ .
5.  $sum=0$ ,  $i=0$ .
  - 5.1 while  $sum < u$  do
    - 5.1.1  $i=i+1$ .
    - 5.1.2 If  $S[i]=0$   $sum=sum+ p_{v,i}$
6.  $t=t+1$ ,  $Res[t]=i$ ,  $S[i]=1$  ,  $v=i$ .
7. if  $t=n$  stop else return to 2.

The algorithm produces the tour permutation in  $Res$  array.

The complexity of steps 2,4,6,7 is  $O(1)$ .

The complexity of steps 1,3,5 is  $O(n)$ .

The  $t$  variable will run for  $n-1$  times so the total algorithm complexity is  $O(n^2)$ .

We were able to reduce the complexity because we don't update the probability matrix but use the uniform distribution between 0 and Norm.

Consider an example:

0	0.3	0.2	0.4	0.1
0	0	0.7	0.15	0.15
0	0.5	0	0.1	0.4
0	0.09	0.11	0	0.8
0	0.6	0.1	0.3	0

= P

S={1,0,0,0,0}      Res={1,x,x,x,x,1} v=1

1.  $u=0.512$  , Norm =  $0.3+0.2+0.4+0.1=1$   
 $u=0.512*1=0.512$

S={1,0,0,1,0}      Res=(1,4,x,x,x,1) v=4

2.  $u=0.82$  Norm= $0.09+0.11+0.8=1$   
 $u=0.82*1=0.82$

S={1,0,0,1,1}      Res={1,4,5,x,x,1} v=5

3.  $u = 0.993$  , Norm=  $0.6+0.1=0.7$   
 $u=0.993*0.7=0.6951$

S={1,0,1,1,1}      Res={1,4,5,3,x,1} v=3

4.  $u=0.62$       Norm=0.5  
 $u=0.2*0.5=0.31$

S={1,1,1,1,1}      Res={1,4,5,3,2,1}

The algorithm stops and in the Res array we can see the permutation, in the example the tour is as follows.

The agent starts at city 1, goes to 4... (1- > 4- > 5- > 3- > 2- > 1)

Please send your comments.

Radislav Vaisman  
[sslava@t2.technion.ac.il](mailto:sslava@t2.technion.ac.il)  
[slvaisman@hotmail.com](mailto:slvaisman@hotmail.com)

The Mat Lab code.

```
% TSP tour generation given the P - probability matrix
% and N - number of cities.
function main
% TSP trajectory generation.
% N- number of cities, P - probability matrix
N=5;
P=[0,0.3,0.2,0.4,0.1;0,0,0.7,0.15,0.15;0,0.5,0,0.1,0.4;0,0.09,0.11,0,0.8;0,0.6,0.1,0.3,0];
Res=generateTour(N,P);
Res
```

```
function Res=generateTour(N,P)
    % initialization
    t=1; % iteration counter
    S(1,1)=1; % S - is a visiting array - we are now in city number 1.
    for i=2 : 1 : N
        S(1,i)=0; % We did not visited the other citys.
    end
    v=1; % current city.
    Res(1,1)=1; Res(1,N+1)=1; % the start and end citys.
    % generation
    while(t<N)
        u=rand(1,1);
        Norm=calcNorm(N,P,v,S);
        u=u*Norm; % reparing random variable.
        j=calcNextCity(P,v,S,u);
        t=t+1;
        Res(1,t)=j;
        S(1,j)=1;
        v=j;
    end
end
```

% calculation of normalization factor.

```
function Norm=calcNorm(N,P,v,S)
    Norm=0;
    for i=1 : 1 : N
        Norm=Norm+(1-S(1,i))*P(v,i);
    end
end
```

```
% find next city in the permutation
function city=calcNextCity(P,v,S,u)
    sum=0;
    i=1;
    while(sum<u)
        i=i+1;
        if(S(1,i)==0)
            sum=sum+P(v,i);
        end
    end
    city=i;
end
```