Traveling Salesman Problem by Simulated Annealing

Jiahui Li

1 Introduction

The matrix below summarizes the distances between any two of 15 cities; e.g., City B and City E are 7 units apart, while City C and City O are 4 units apart.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
A	0	1	2	4	9	8	3	2	1	5	7	1	2	9	3
B	1	0	5	3	7	2	5	1	3	4	6	6	6	1	9
C	2	5	0	6	1	4	7	7	1	6	5	9	1	3	4
D	4	3	6	0	5	2	1	6	5	4	2	1	2	1	3
E	9	7	1	5	0	9	1	1	2	1	3	6	8	2	5
F	8	2	4	2	9	0	3	5	4	7	8	3	1	2	5
G	3	5	7	1	1	3	0	2	6	1	7	9	5	1	4
H	2	1	7	6	1	5	2	0	9	4	2	1	1	7	8
I	1	3	1	5	2	4	6	9	0	3	3	5	1	6	4
J	5	4	6	4	1	7	1	4	3	0	9	1	8	5	2
K	7	6	5	2	3	8	7	2	3	9	0	2	1	8	1
L	1	6	9	1	6	3	9	1	5	1	2	0	5	4	3
M	2	6	1	2	8	1	5	1	1	8	1	5	0	9	6
N	9	1	3	1	2	2	1	7	6	5	8	4	9	0	7
O	3	9	4	3	5	5	4	8	4	2	1	3	6	7	0

We will implement a simulated annealing algorithm to solve this famous traveling salesman problem, find the optimal traveling path. We will also research how the initial temperature and the acceptance probability effect the behavior of the algorithm.

2 Method

First, we initial the path with a random order. And calculate the length of the path. Then we begin with $\alpha(\tau) = p\tau, \tau = 400, p = 0.999, \beta(m) = 100, gt(\theta|\theta_t)$ to be uniform distribution.

By above initial parameter, suppose iteration t belongs to stage j:

Step1: Select two cities with equal probability and change their position in the path, then we calculate the length of the new path.

Step2: Letδ = length of new path - length of current path:

If $\delta \leq 0$,then set the current path = new path;

If $\delta > 0$, then set current path = new path with probability $e^{-\delta/tau}$.

Step3: Repeat step1 and step 2 for 100 times (since $\beta(m) = 100$)

Step4: Updata $\alpha(\tau) = p\tau$

If the length of path does not change for 300 times, we stop the iteration.

Then, with $\tau = 400, p = 0.999$, we get the path(I, O, K, M, F, C, E, J, G, D, N, B, H, L, A, I) with length 21.

Next, we try to change the initial temperature τ_1 to see what would happen. We try $\tau_1 = 100$, 200, 300, 400, 500, 600, 700, 800, 900, 1000, other parameters remain the same. Here is the result.

τ	p	stage when stop	length	path
100	0.999	101	18	O, J, L, A, B, H, E, C, I, M, F, N, G, D, K, O
200	0.999	112	21	I, C, M, F, D, N, G, E, H, L, K, O, J, B, A, I
300	0.999	114	17	N, F, M, K, O, J, E, C, I, A, B, H, L, D, G, N
400	0.999	116	21	I, O, K, M, F, C, E, J, G, D, N, B, H, L, A, I
500	0.999	119	22	I, C, E, J, G, H, B, N, F, M, A, L, D, O, K, I
600	0.999	120	18	K, L, H, M, F, D, G, N, B, A, I, C, E, J, O, K
700	0.999	123	21	J, G, E, H, A, I, C, N, B, F, M, K, O, D, L, J
800	0.999	124	20	J, G, D, N, F, B, H, E, C, A, I, M, K, O, L, J
900	0.999	127	19	J, G, N, D, F, B, A, I, M, C, E, H, L, K, O, J
1000	0.999	124	22	N, C, I, O, K, H, M, F, D, G, E, J, L, A, B, N

We can see that when $\tau_1 = 300$, we get the shortest path (N, F, M, K, O, J, E, C, I, A, B, H, L, D, G, N) with length 17. And with different initial temperature, we will get different results, and most of them can only find the local shortest path. What's more, from the table, we can obviously find that with larger temperature, we will run in more stage to find the corresponding (local)shortest path.

Next, we try to change the value of p to see what would happen. We try p = 0.999, 0.99, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1 and other parameters remain the same. Here is the result.

With $\tau = 400$,

τ	p	stage when stop	length	path
400	0.999	116	21	I, O, K, M, F, C, E, J, G, D, N, B, H, L, A, I
400	0.990	39	25	I, A, C, M, H, L, D, G, N, B, F, O, K, E, J, I
400	0.900	18	20	D, F, B, N, G, E, J, L, A, I, C, M, H, K, O, D
400	0.800	16	20	E, J, G, N, B, H, A, L, D, F, M, K, O, I, C, E
400	0.700	11	21	B, N, E, G, A, I, C, M, F, D, L, J, O, K, H, B
400	0.600	13	19	F, N, D, G, J, O, K, L, A, I, M, C, E, H, B, F
400	0.500	12	20	K, M, I, C, E, G, H, A, B, F, N, D, L, J, O, K
400	0.400	11	19	M, C, I, A, L, H, E, J, G, N, B, F, D, O, K, M
400	0.300	7	23	J, G, E, C, N, D, F, B, H, L, A, M, I, K, O, J
400	0.200	9	22	M, K, H, E, J, G, N, B, A, I, C, O, L, D, F, M
400	0.100	7	22	M, F, N, D, L, J, E, G, A, B, H, K, O, C, I, M

We can find that, in most situation, with larger p, we are more likely to find a shorter path. What's more, it is quite obvious that with smaller p, we will run in much less stage to find the corresponding (local)shortest path, which means faster.

Since from the different try of τ, we find when $\tau = 300$, we can get a shortest path, so we try different p with $\tau = 300$ to see what would happen.

τ	p	stage when stop	length	path
300	0.999	114	17	N, F, M, K, O, J, E, C, I, A, B, H, L, D, G, N
300	0.990	38	21	G, D, F, L, A, I, K, O, J, E, C, M, H, B, N, G
300	0.900	14	23	F, D, G, J, L, A, H, K, O, I, M, C, E, N, B, F
300	0.800	12	22	M, F, D, N, B, H, E, C, O, K, L, J, G, A, I, M
300	0.700	10	21	E, C, A, B, H, L, D, F, M, I, K, O, J, G, N, E
300	0.600	9	22	K, H, L, J, O, I, A, B, N, F, M, C, E, G, D, K
300	0.500	9	24	E, G, J, L, D, F, N, B, I, O, K, M, C, A, H, E
300	0.400	7	24	D, G, E, C, A, O, K, I, J, L, H, B, N, F, M, D
300	0.300	7	25	I, A, O, J, G, H, L, D, N, B, F, C, E, K, M, I
300	0.200	10	28	A, L, O, J, G, E, K, D, F, I, M, H, B, N, C, A
300	0.100	7	22	M, C, A, I, J, E, H, L, K, O, D, G, N, B, F, M

From the result, we can find that except p =0.999, only the local shortest path can be found with other values of p. And with the larger p, the length of the local shortest path found is more likely to be shorter. But with smaller p, we will run in much less stage to find the corresponding (local)shortest path, which means faster. These two findings are consistent with the two we get with $\tau = 400$.

Above all, the shortest path we can find is (N, F, M, K, O, J, E, C, I, A, B, H, L, D, G, N), with length 17. And with larger temperature, we will spend more time to find the corresponding (local)shortest path. And with larger p, the length of the local shortest path found is more likely to be shorter. But with smaller p, we will run much faster to find the corresponding (local)shortest path.

3 Code

```
##### distance matrix
dis = matrix(c(0,1,2,4,9,8,3,2,1,5,7,1,2,9,3,
               1,0,5,3,7,2,5,1,3,4,6,6,6,1,9,
               2,5,0,6,1,4,7,7,1,6,5,9,1,3,4,
               4,3,6,0,5,2,1,6,5,4,2,1,2,1,3,
               9,7,1,5,0,9,1,1,2,1,3,6,8,2,5,
               8,2,4,2,9,0,3,5,4,7,8,3,1,2,5,
               3,5,7,1,1,3,0,2,6,1,7,9,5,1,4,
               2,1,7,6,1,5,2,0,9,4,2,1,1,7,8,
               1,3,1,5,2,4,6,9,0,3,3,5,1,6,4,
               5,4,6,4,1,7,1,4,3,0,9,1,8,5,2,
               7,6,5,2,3,8,7,2,3,9,0,2,1,8,1,
               1,6,9,1,6,3,9,1,5,1,2,0,5,4,3,
               2,6,1,2,8,1,5,1,1,8,1,5,0,9,6,
               9,1,3,1,2,2,1,7,6,5,8,4,9,0,7,
               3,9,4,3,5,5,4,8,4,2,1,3,6,7,0),ncol=15,nrow=15)
colnames(dis) = c('A','B','C','D','E','F','G','H','I','J','K','L',
                  'M','N','O')
rownames(dis) = c('A','B','C','D','E','F','G','H','I','J','K','L',
                  'M','N','O')

##### The function to calculate the length
calculate_length = function(theta){
  length = 0
  for (i in 1:14){
    length = length + dis[theta[i],theta[i+1]]
  }
  length = length +dis[theta[1],theta[15]]
  return(length)
}

##### The function to solve TSP by simulated annealing
TSP = function(mj,tao,p,Nsame){
  len = numeric(50)
  set.seed(223)
  theta = sample(1:15,15)
  delta = calculate_length(theta)
  j = 1
  count = 1
  len = 0
  set.seed(130)
  while(TRUE){
    tao  = tao * p^(j-1)
    for (i in 1:mj){
      ind = sample(theta,2)
      theta_star = theta
      theta_star[ind[1]] = theta[ind[2]]
      theta_star[ind[2]] = theta[ind[1]]
      delta = calculate_length(theta_star) - calculate_length(theta)
      if (delta <= 0){theta = theta_star}else {
        index = sample(c(0,1),1,prob = c(exp(-delta/tao), 1-exp(-delta/tao)) )
        if (index==0){theta = theta_star}}
      if (len==calculate_length(theta)){count = count+1}else{
        count =1
        len = calculate_length(theta)}
      if (count ==Nsame){break}
    }
    if (count == Nsame){break}
    j = j+1
  }
  return(c('j:',j,'length:',len,'theta:',theta))
}

##### with different tau at p = 0.999
set_tao = seq(100,1000,100)
path_tao = data.frame(tao = seq(100,1000,100), p = 0.999)
for (i in 1:10){
  tao = i*100
  TSPs = TSP(mj = 100,tao = tao,p = 0.999,Nsame = 300 )
  path_tao$stage[i] = TSPs[2]
  path_tao$length[i] = TSPs[4]
  path = as.integer(TSPs[6:20])
  path = colnames(dis)[c(path,path[1])]
  path_tao$path[i] = paste(path,collapse=", ")
}

##### with different p at tau =400
set_p = c(0.999,0.99,seq(0.9,0.1,length.out = 9))
path_p = data.frame(tao = 400, p = set_p)
for (i in 1:11){
  p = set_p[i]
  TSPs = TSP(mj = 100,tao = 400,p = p,Nsame = 300 )
  path_p$stage[i] = TSPs[2]
  path_p$length[i] = TSPs[4]
  path = as.integer(TSPs[6:20])
  path = colnames(dis)[c(path,path[1])]
  path_p$path[i] = paste(path,collapse=", ")
}

##### with different p at tau =300
set_p = c(0.999,0.99,seq(0.9,0.1,length.out = 9))
path_p = data.frame(tao = 300, p = set_p)
for (i in 1:11){
  p = set_p[i]
  TSPs = TSP(mj = 100,tao = 300,p = p,Nsame = 300 )
  path_p$stage[i] = TSPs[2]
  path_p$length[i] = TSPs[4]
  path = as.integer(TSPs[6:20])
  path = colnames(dis)[c(path,path[1])]
  path_p$path[i] = paste(path,collapse=", ")
}
```