STA 243 Assignment 1

Chen Zihao 915490404

Jichen Xu 915543694

1. Simulated Annealing

(Chen zihao:75%, Jichen Xu:25%)

Let input the distance matrix into R first by pasted from the pdf.

1.1 First, we need to define the path in a vector form. label the city with numbers instead of alphabet, e.g. 1 for A , 2 for B, etc. then we can get the path which is also the candidate in this problem:

$\theta = (i_1, ..., i_{15})$ where $i_j \neq i_k$ and $i_j \in \{1, 2, ..., 15\}$

the candidate space contains all the possible θ.

To solve the travel problem, we should also compute the total distance. Here is the function to calculate the total distances for each candidate path.

$$Distance(\theta) = d(i_{15}, i_1) + \sum_{j=1}^{14} d(i_j, i_{j+1})$$

where $d(i_j, i_k)$ is the distance from i_j city to the i_k city which is given in the matrix mentioned in the beginning $d(i_j, i_k) = D_{jk}$

1.2 Second, the Simulated Anealing algorithm part.

1.2.0 Initialization

initialize $\tau_1 = 400$ and other things mentioned in the question.

1.2.1 step 1: sample a candidate

From the first part, we have define what is the candidate space. To use a uniform distribution for the proposal density. Get the random number j and k uniformly without replacement and then switch i_j and i_k and I did it twice to get the neighbour which means in this step I will exchange at least 2 at most 4 variables.

1.2.2 step 2: calculate the distance and compare them.

a) if the new candidate is better than the previous best one, take the candidate as the best.

$\Delta = Distance(\theta^*) - Distance(\theta_k) < 0$ where θ_k is the recent best solution and θ^* is the new candidate drawn by the step 1, then take θ^* as θ_{k+1}

b) Otherwise, $\theta_{k+1} = \theta^*$ with a probability $\exp(-\frac{\Delta}{\tau_j})$, keep $\theta_{k+1} = \theta_k$ o.w.

1.2.3 step 3: repeat step 1 and 2 $m_j = 100$ times.

1.2.4 step 4: update $\tau_{j+1} = \alpha(\tau_j), m_{j+1} = \beta(m_j)$ and move to stage j+1.

1.2.5 step 5: reheat.

run the algorithm again with the initial point as the previous result.

here is the result for different parameters:

1. $p = 0.999, tau = 400$

the result path is of length:

```
## [1] 53
```

the result is bad because $exp(-30/(400*0.999^{1000})) = 0.815$, the chance to accept the worse case is still high even in the final stage. The temperture is still high in the end.

2. $p = 0.99, tau = 400$

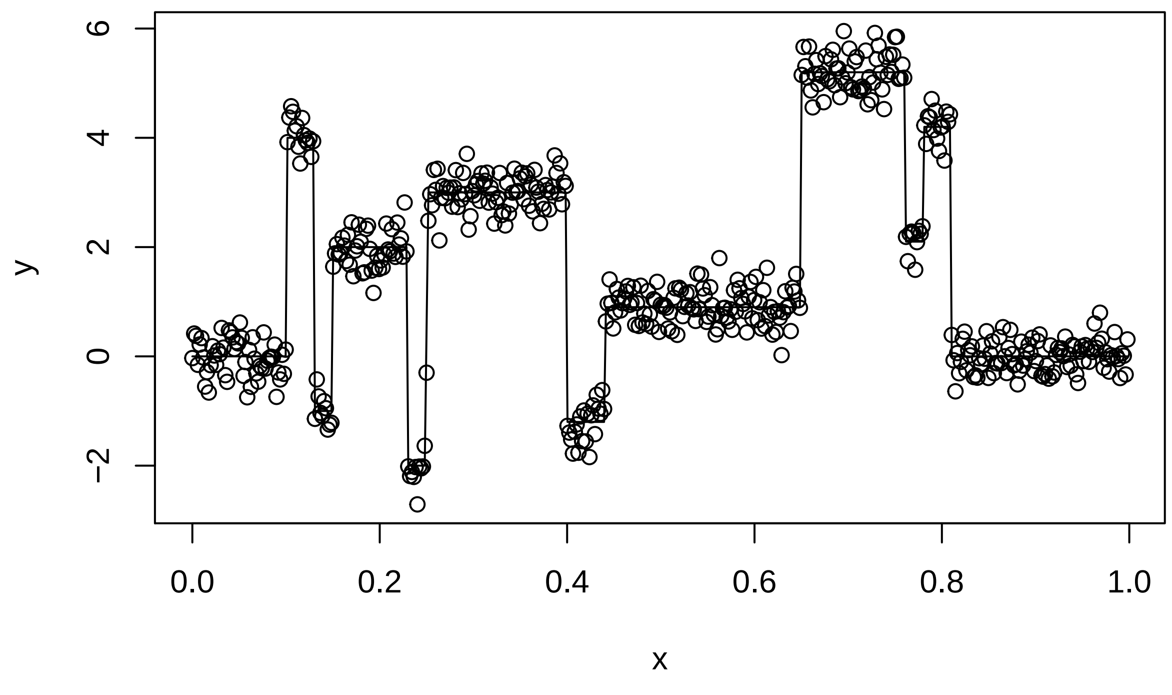
```
## [1] 19
```

I run it several times, the results are stable around 17 to 20, which are good enough. One of the best path (with distance 17) I found is (3,9,1,2,8,12,10,15,11,13,6,14,4,7,5)

2. Genetic Algorithms

(Chen zihao:75%, Jichen Xu:25%)

2.0 the graph of original data is shown as follow.



2.1 Randomly generate an initial population of chromosomes of size S.

I will use a matrix form to store the chromosomes, each row of the matrix is a chromosome. Every chromosome is a 512 demension vector and each element is 0 or 1. If the ith element is 1 which means x_i is a breakpoint.

there is a special point (x_1, y_1) it will always be a break point as the model (1) shows, for convience in coding, i will set the first elements in the chromosomes always be zero.

2.2 Compute the MDL/AIC value for each of the S chromosomes.

First we need to code the function to calculate the MDL and AIC which will be shown in the coding file. Then We can compute the value for each chromosomes.

2.3 Sort the MDL/AIC values in descending order. Assign rank 1 to the chromosome with the largest MDL/AIC value, rank 2 to the one with the second largest and so on. Denote the rank of chromosome i as r_i.

2.4 With probability P_{cross}, perform a crossover operation. Otherwise, perform a mutation operation.

for crossover, first we need to select the two parents.

for the ith chromosomes in the orderd matrix above, it has the probability of $\frac{2r_i}{(1+S)S}$ to be chosen. To make life easiler, I draw the parents with replacement.

In this case, it is equivalent as the following case:

put $(1+S)S/2$ balls in a box with the following rules: for each ball we label it with a number and print them with different colors. The first ball with the first color and then the following two balls with the second color, and then the following three balls with the third color and so on.

for the ith color we have i balls in the box so that we have i/(1+S)S probability to get a i-th color balls with 1 draw. On the other hand, the nuumber of the balls is uniform distributed. So that we can say that:

Uniformly randomly choose a constant C from $\{1, 2, ..., (1+S)S\}$ and then finding the smallest interger n such that

$$C \leq \frac{(1+n)n}{2}$$
$$n \geq \sqrt{2C+0.25} - 0.5$$

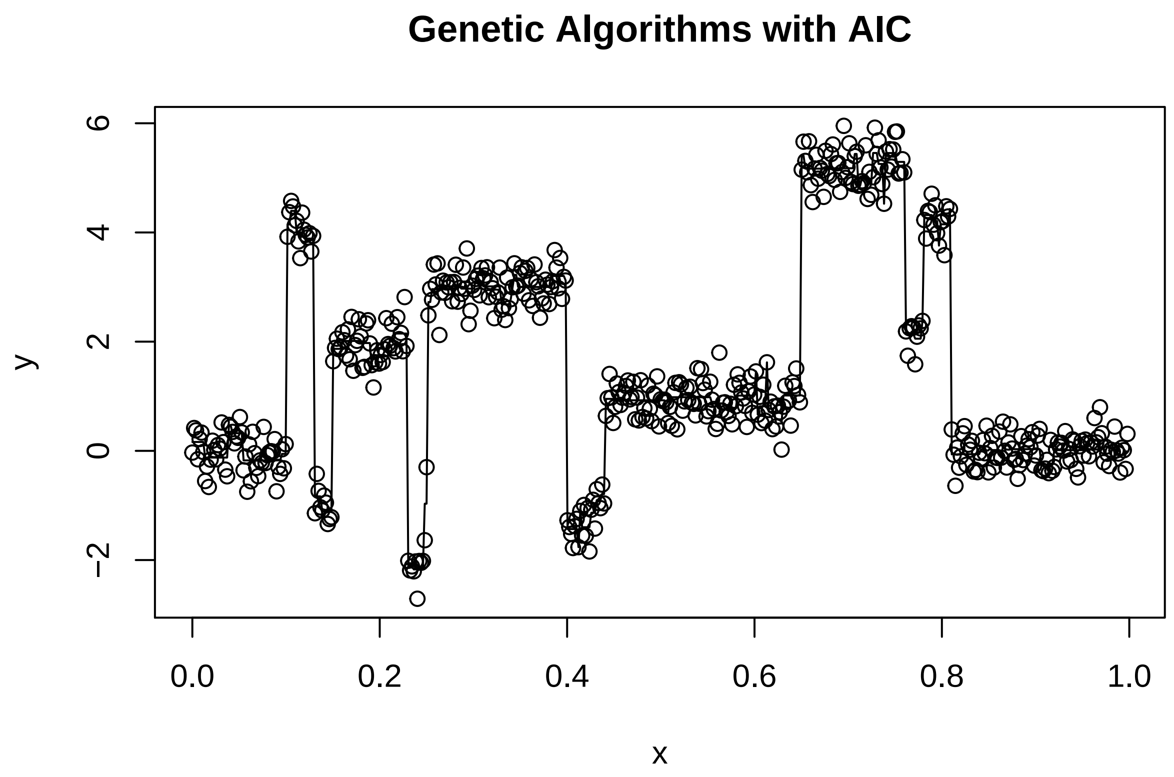
then we take the nth chromosomes out and this chromosome has the probability of $\frac{2r_i}{(1+S)S}$ to be chosen.

2.5 Repeat step 4 until S child chromosomes are produced. That is, until a whole new generation is obtained.

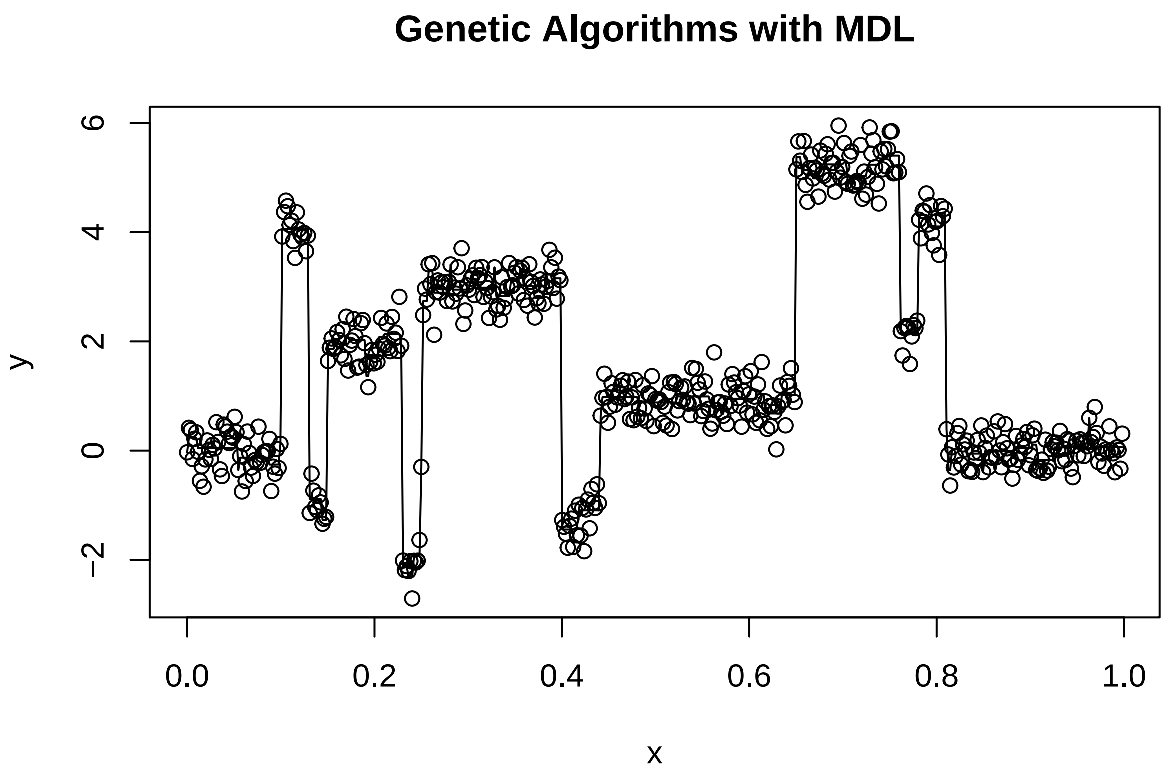
2.6 Repeat Steps 2 to 5 using the new generation as the initial population. Then repeat the whole process until the MDL/AIC value of the best chromosome does not change for N_{same} generations.

2.7 The best chromosome of the youngest generation is taken as the minimizer of MDL/AIC.

for AIC



for MDL

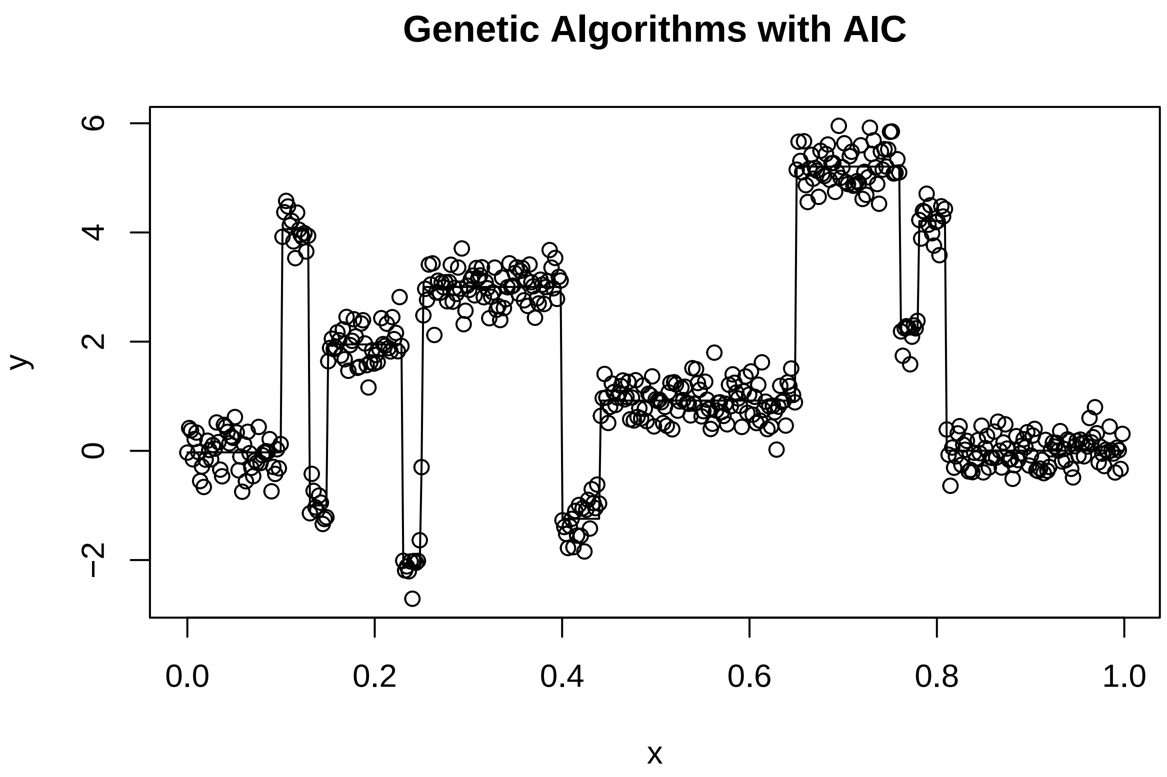


Further thinking about the genetic algorithms.

Can we just take the whole gene as a gene bank? It means that we do not do the crossover but draw the gene with a weighted frequency instead. It is the another angle to consider gene in nature. As the generation changes, the frequency of the better gene will increase. All we need is to simulate the Survival of the fittest.

In other words, change the crossover part of the algorithm. the operation above is to take two parents and exchange the genes in the progress. How about just consider each gene location sperately and draw the values with a weighted frequency?

For example, in this homework 2, we sort the MDL values in descending order. Assign rank to the chromosome as usual. But this time multiple the chromosome with its rank (weighted), sum all the chromosomes and divided it by $r_1 + ... + r_S = \frac{(1+S)S}{2}$ and we will get a vector which elements is the weighted frequency as mentioned before. Take this frequency as the p in bernoulli(p) and draw a gene.



Genetic Algorithms with AIC
y
x
6
4
2
0
−2
0.0
0.2
0.4
0.6
0.8
1.0