Problem Description:

We will give some random job sequences (e.g,, 4,2,1,3 and here job 4 will be finished first then 2, 1 and 3) and from that using a little perturbation, it can create more job sequences randomly. These sequences need to be converted into binary so that we can use these sequences as the input of the neural network. Here we are using ART (Adaptive Resonance Theory) as neural network. ART will cluster them into groups based on their genotype and phenotype. Each group will hold similar type job sequences.

Coding part can be divided into following:

Part 1:

It will accept some random job sequences at first. Then it will create some more job sequence randomly from the given job sequences so that we can get a bunch of job sequences from the same job sequence.

Part 2:

The job sequences must be converted into binary. Binary conversion can be done into two ways which is discussed below. We need to show two ways so that we can compare which binary conversion is good.

Part 3:

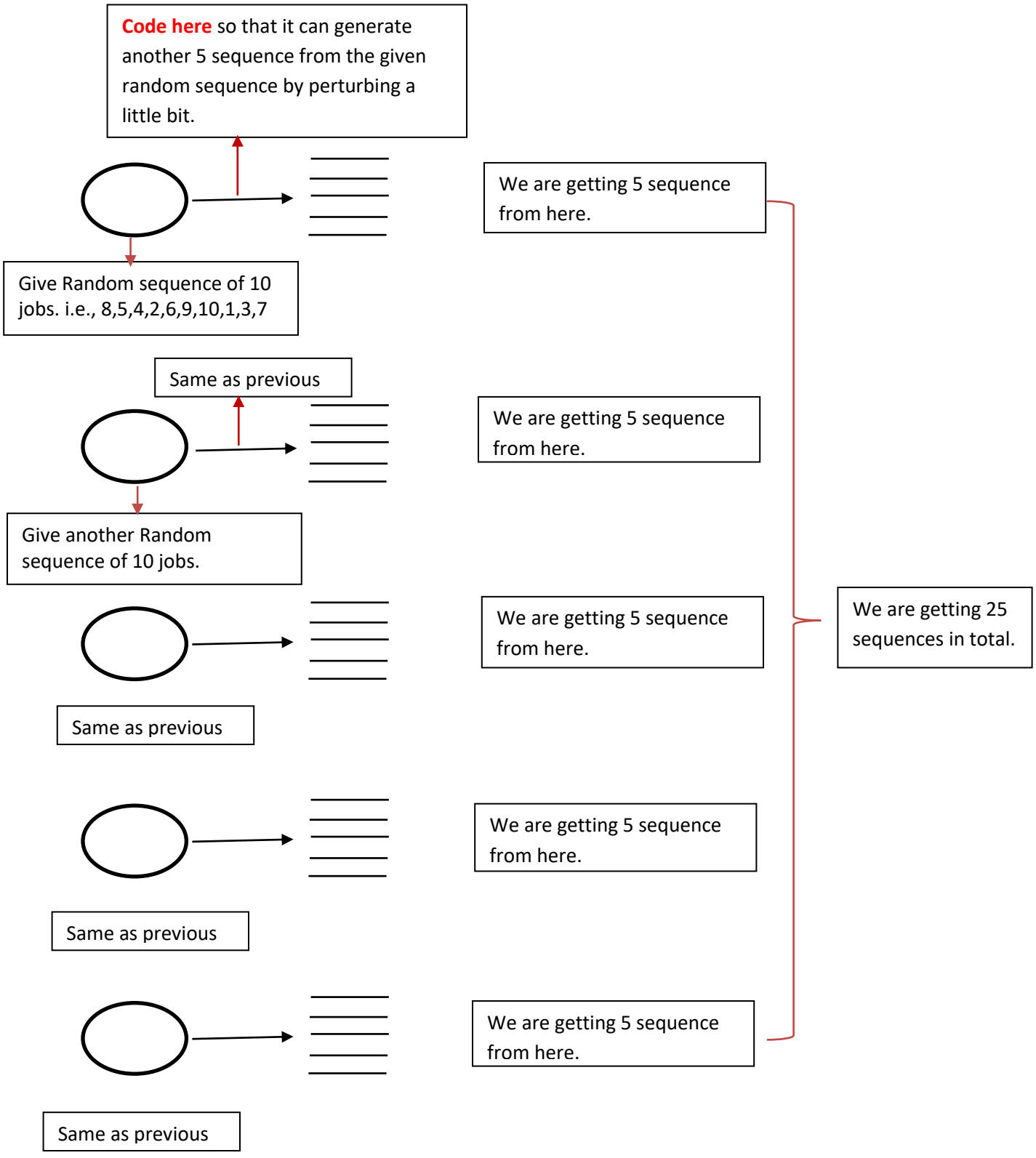
Binary converted job sequences will go directly to the neural network input zone. The neural network will classify them as group.

Part 4:

In the neural network, use centroid and Euclidian distance to avoid misclassification.

Requirements: You need to do the code for 10 & 20 (separately, at first using 10 then go for 20) jobs and use c++ language and visual studio software. You need to show binary conversion using both ways but not in the same code. One code might use one way and the other code might use other way.

Example for 10 jobs



These 25 sequences will work as the input of the neural network. Here we are using Adaptive Resonance Theory (ART) as neural network. ART can only take binary input. The 25 sequences that we got earlier need to be converted into binary so that we can use it for the input of the neural network. There are two ways to convert them into binary.

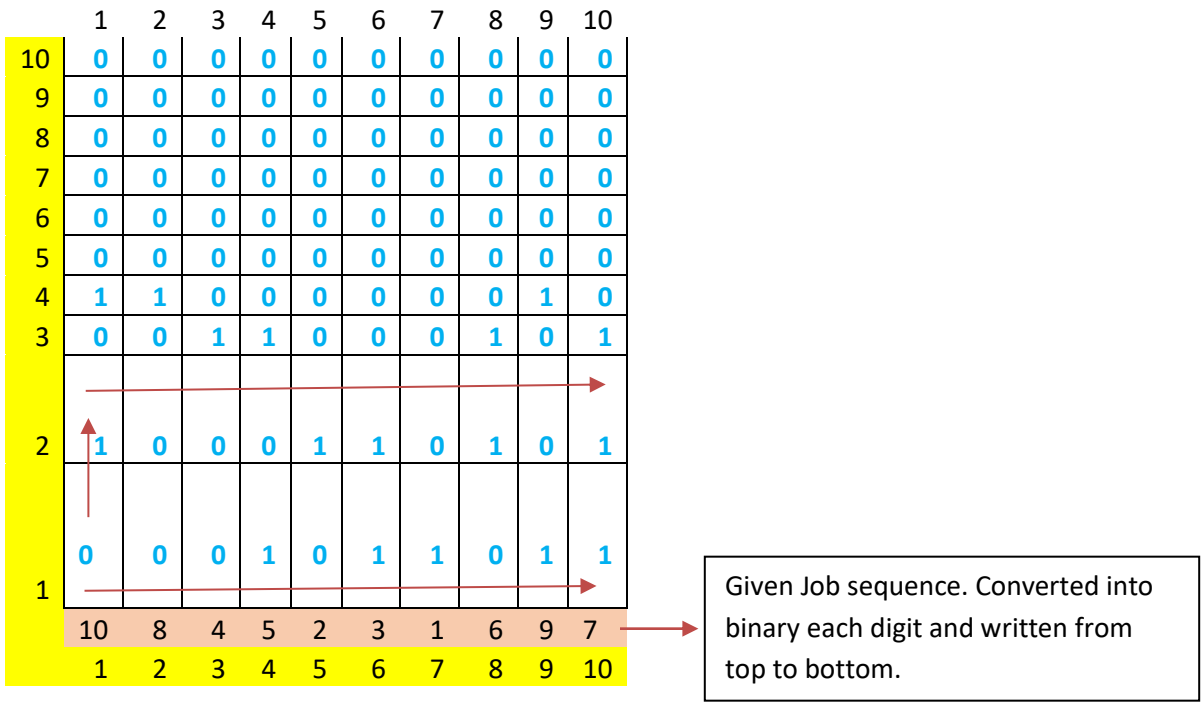
Way 1:

Suppose we get a job sequence as the following

10,8,4,5,2,3,1,6,9,7

We can simply convert each digit into binary. Then it will become:

	1	2	3	4	5	6	7	8	9	10
10	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
4	1	1	0	0	0	0	0	0	1	0
3	0	0	1	1	0	0	0	1	0	1
2	1	0	0	0	1	1	0	1	0	1
1	0	0	0	1	0	1	1	0	1	1
	10	8	4	5	2	3	1	6	9	7
	1	2	3	4	5	6	7	8	9	10



One input string for the neural network will be for this job sequence:

000101101110001101010011000101……0 (follow the arrow direction from the table to get this whole sequence from bottom left to top right corner)

Using this mechanism every sequence that we got from earlier must be converted into binary.

Way 2:

Table 1

	1	2	3	4	5	6	7	8	9	10
10	1	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	1	0
8	0	1	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0	1	0	0
5	0	0	0	1	0	0	0	0	0	0
4	0	0	1	0	0	0	0	0	0	0
3	0	0	0	0	0	1	0	0	0	0
2	0	0	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
	10	8	4	5	2	3	1	6	9	7
	1	2	3	4	5	6	7	8	9	10

Table 2

	1	2	3	4	5	6	7	8	9	10
10	1	1	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	1	1	1
8	1	1	1	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	1
6	0	0	0	0	0	0	1	1	1	0
5	0	0	1	1	1	0	0	0	0	0
4	0	1	1	1	0	0	0	0	0	0
3	0	0	0	0	1	1	1	0	0	0
2	0	0	0	1	1	1	0	0	0	0
1	0	0	0	0	0	1	1	1	0	0
	10	8	4	5	2	3	1	6	9	7
	1	2	3	4	5	6	7	8	9	10

For this method, you can see from the above that I have made the box putting number 1 (light orange color) corresponding to that job (for job 10, I have made the 10th box 1, for job 8 I have made the box 8th as 1 and so on from Table 1.

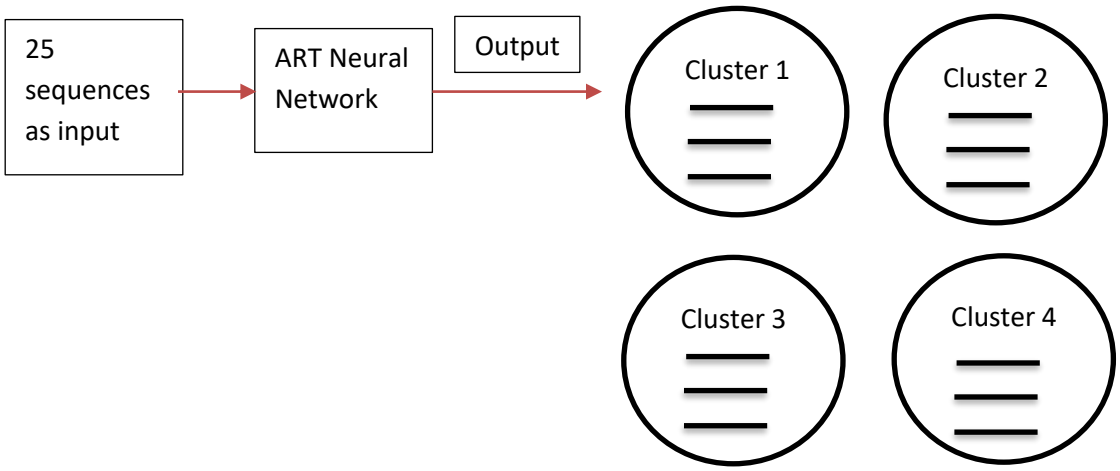
For Table 2, I have made the back and forth of corresponding 1 (orange) box as putting additional 1 (color purple). If there is no box in front of that box, then leave it as it is. If there is a box at the back then the back box will be marked as 1 (see row 10). Same for the case where there no space at the back then only put in the front box e.g., row 7.

Here, one input string for the neural network will be for this job sequence:

00000111000001110000…….0

Using this mechanism every sequence that we got from earlier has to be converted into binary.

These 25 sequences after converting them into binary will work as input of the Neural Network. This neural network will cluster them into groups according to their genotype and phenotype.



I will give you the ART Algorithm and code as well. You just need to add.

But when clustering there is a possibility to do misclassification. As we are doing perturbation again and again, it might possible that one job sequence might be classified to a different group. For that, you need to introduce centroid and Euclidian distance concept in the ART. Each cluster will have a centroid and it will calculate the distance of each member (here the job sequences) from the centroid. The shortest distance will decide its membership of that cluster.

ART Algorithm (General information, I have given you the code)

Algorithm:

- Initialize each top down weight $t_{j,i}(0) = 1$
- Initialize bottom up weight

  $b_{i,j}(0) = \frac{1}{n+1}$; *where n is the number of components in the input vector*.

While the network has not stabilized, **do**

1. Present a randomly chosen pattern $x = (x_1 + x_2 + ... + x_n)$ for learning

2. Let the active set A contain all nodes; calculate

$y_j = b_{1,j}.x_1 + ... + b_{n,j}.x_n$ *for each node* $j \in A$**;**

3. **Repeat**

a) Let j* be a node in A with largest y_j with ties being broken arbitrarily;

b) Compute $s^* = (s_1^*, ...s_n^*)$ *where,* $s_i^* = t_{j,i}^*.x_i$;

c) Compare similarity between s^* and x with the given vigilance parameter ρ:

if $\frac{\sum_{i=1}^{n} s_i^*}{\sum_{i=1}^{n} x_i} \le \rho$ **then** remove j* from set A **else** associate x with node j* and update weights:

$$b_{i,j}(new) = \frac{t_{j,i}(old).x_i}{0.5 + \sum_{i=i}^{n} t_{j,i}^*(old).x_i} \quad \text{and} \quad t_{j,i}^*(new) = t_{j,i}^*(old).x_i$$

Until A is empty or x has been associated with some node j.

4)if A is empty then create new node whose weight vector coincides with current input pattern x;

End while