# Evolutionary Comp – CS5320 Project 3 Report

**Spring - 2016**
**Kruthika Holla**

For this project, I have a data structure class: Data.java . I have selected the top 30% from the population and applied Gene Pool Recombination to get the pop-size back to N and t hen applie mutation on it.

Data.java:

```java
import java.util.Random;


public class Data {
    private int bitStringLength;
    private String individual;

    //Constructor
    public Data(){
        bitStringLength = 500; //11 bits in each individual;
total 33
        setIndividual();
    }

        //Copy constructor
        public Data (String vec){
            this.individual = vec;
            this.bitStringLength = vec.length();
        }

        //setter for individual
        public void setIndividual(){
            individual = "";
            for(int i = 0; i < bitStringLength; i++){
                //get bit
                Random randInt = new Random();
                int temp = randInt.nextInt(2) ;
                individual += String.valueOf(temp);
            }
        }

        //getter for individual
        public String getIndividual(){
            return individual;
        }

        //get length of each individual
        public int getIndividualLength(){
            return individual.length();
        }

        //replacement for individual at index i
        public void replaceIdividualAtIndex(int index, char
x){
            individual = changeCharInPosition(index, x,
individual);
        }

        public String changeCharInPosition(int position,
char ch, String str){
           char[] charArray = str.toCharArray();
           charArray[position] = ch;
           return new String(charArray);
        }

        //get gene at index i
        public char getComponentAtIndex(int index){
            char a_char = individual.charAt(index);
            return a_char;
        }

        //tostring
        public String toString(){
            String geneString = "";
            geneString = getIndividual();
            return geneString;
        }
}
```

project3.java:

```
import java.util.*;

public class Project3 {

    public static void main(String[] args) {
        int N = 100;  //pop size
        int t_max = 500; //max generations
        int bestOfRun = 0; //best of run
        Data bestOfRunVector = new Data();
        int[] fitnessDistr = new int[N]; //fitness holder

        ArrayList<Data> data = new ArrayList<Data>(N);
        int t = 0;

        //Initialize Pop(0)
        for(int i = 0; i < N; i++){
            data.add(new Data());
        }

        int stringLength = data.get(0).getIndividualLength();

        //Get fitness
        fitnessDistr = getFitnessDistr(data, N);

        int bestOfGeneration = fitnessDistr[0];
        int index = 0;
        int avgOfGen = fitnessDistr[0];


        for (int i = 1; i < N; i++){
            avgOfGen += fitnessDistr[i];
            if (fitnessDistr[i] > bestOfGeneration){
                bestOfGeneration = fitnessDistr[i];
                index = i;
            }
        }

        int[] bestOfGen = new int[(t_max/10) + 1];
        int[] AvOfGen = new int[(t_max/10) + 1];
        int j =0;

        System.out.println("Generation 0");
        System.out.println("Best of generation: "+
bestOfGeneration +" corresponding to vector: "+data.get(index));
        bestOfGen[j] = bestOfGeneration;

        System.out.println("Average of generation: "+
avgOfGen/N);
        AvOfGen[j] = avgOfGen/N;
        System.out.println("----------------------------------
----------------------");
        //System.out.println("First in main: 
"+data.toString());

        while(t < t_max){
            ArrayList<Data> afterSelection = new 
ArrayList<Data>(selection(data, N, fitnessDistr));
            ArrayList<Data> afterCrossover = new 
ArrayList<Data>(N);
            ArrayList<Data> afterMutation = new 
ArrayList<Data>(1);
            //System.out.println("After selection in main: 
"+afterSelection.toString());

            int afterSelectionSize = afterSelection.size();
            for (int i = 0; i < N; i++){
                Data tempSave = new Data();
                for(int k = 0; k < stringLength; k++){
                    int rnd = new 
Random().nextInt(afterSelectionSize);
                    tempSave.replaceIdividualAtIndex(k, 
afterSelection.get(rnd).getComponentAtIndex(k));
                }
                //System.out.println("After replacement 
:"+tempSave.getIndividual());
                afterCrossover.add(tempSave);
                int fit = fitness(tempSave);
                //best of Run
                if (fit > bestOfRun){
                    bestOfRun = fit;
                    bestOfRunVector = tempSave;
                }
                //mutation
                Data mutationVector = 
mutationApplied(tempSave);
                afterMutation.add(mutationVector);
                fit = fitness(mutationVector);
                //best of Run
                if (fit > bestOfRun){
                    bestOfRun = fit;
                    bestOfRunVector = mutationVector;
                }
            }
```

```
                data = afterMutation; //update data vector
                fitnessDistr = getFitnessDistr(data, N); //update
fitness distribution

                //Get best and average at every 10th generation
                if ((t+1) % 100 == 0){
                    bestOfGeneration = fitnessDistr[0];
                      avgOfGen = fitnessDistr[0];
                      index = 0;

                      for (int i = 1; i < N; i++){
                           avgOfGen += fitnessDistr[i];
                           if (fitnessDistr[i] >
bestOfGeneration){
                                 bestOfGeneration =
fitnessDistr[i];
                                 index = i;
                           }
                      }
                      j++;
                      avgOfGen = avgOfGen / N;
                      System.out.println("Generation "+ (t+1));
                      System.out.println("Best of generation is:
"+ bestOfGeneration+" corresponding to vector: "+data.get(index)
);
                      System.out.println("Average of generation
is: "+ avgOfGen);
                      bestOfGen[j] = bestOfGeneration;
                      AvOfGen[j] = avgOfGen;
                      System.out.println("Best so far: "+
bestOfRun + " corresponding to vector: "+bestOfRunVector);
                      System.out.println("-----------------------
---------------------------------");
                }
                t++;
           }


     }

     //function to get fitness distribution
           public static int[] getFitnessDistr(ArrayList<Data>
data, int popSize){
                int[] fitnessDistr = new int[popSize];
                for (int i = 0; i< popSize; i++){
                     int fit = fitness(data.get(i));
                     fitnessDistr[i] = fit;
                }

                return fitnessDistr;
           }

     //calculate fitness
           public static int fitness(Data data){
                int f = 0;
                int n = data.getIndividualLength();
                for (int i = 0; i< n; i++){
                     if(data.getComponentAtIndex(i) == '1')
                     f ++;
                }

                return f;
           }

           //function to select the top 30% individuals
           public static ArrayList<Data>
selection(ArrayList<Data> data, int popSize, int[]
fitnessDistr){
                ArrayList<Data> selectedData = new
ArrayList<Data>(data);
                int[] fD = new int[popSize];
                //Make a copy of fitness distr
                System.arraycopy(fitnessDistr, 0, fD, 0,
popSize);


                for (int i = 0; i < popSize - 1; i++){
                     int index = i;
                     for (int j = i + 1; j < popSize; j++){
                          if(fD[j] < fD[index]){
                               index = j;
                          }
                     }
                     int temp = fD[ index ];   //swap
                     Data tempdata = selectedData.get(index);
                   fD[ index ] = fD[ i ];
                   selectedData.set(index, selectedData.get(i));
                   fD[ i ] = temp;
```

```
              selectedData.set(i, tempdata);
            }


            int selectedSize = (30*popSize)/100;
            ArrayList<Data> finalSelectedData = new
ArrayList<Data> (selectedData.subList(popSize - selectedSize,
popSize));
            return finalSelectedData;

        }

        //Mutation
        public static Data mutationApplied(Data data){
            double p_m = 0.01; //mutation rate
            String temp = data.getIndividual();
            Data mutData = new Data(temp);
            String zero = "0";
            String one = "1";
            char z = zero.charAt(0);
            char o = one.charAt(0);

            //for each allele
            for (int i = 0; i < data.getIndividualLength();
i++){
                Random rand = new Random();
                double randNum = rand.nextDouble();
                char x = data.getComponentAtIndex(i); //get
the allele
                if(randNum < p_m){

                    if (x == z)

    mutData.replaceIdividualAtIndex(i, o);
                    else

    mutData.replaceIdividualAtIndex(i, z);
                }
            }
            return mutData;
        }

}
```

Sample of 1 run:

```
Generation 0
Best of generation: 287 corresponding to vector:
01111100001110110111010010110111100101011011101001101100010010100
11111111110000001011100111110110001011000011110011010000001110100 00
10001011101111111110010101000110000111101011101001001111011010110
11000111010001101101101011111011110100101101110110111111011001010
01001010000000001001111010101110011110011110011001100011001010110 1
01010101001110011101011101110111011111010011111101111101111111001
11100101000000110100111101010110010111110001011110001000010111110 1
000101010110100011111100110101101110011101001
Average of generation: 250
--------------------------------------------------------
Generation 10
Best of generation is: 383 corresponding to vector:
10111111110011010111010111011111111111111111110111011111111111100 11
11111011011111111111011010111100011111011011111011110101101101101
01100011111111111110110101111101111101111101101101011011111110111
10011101101111010111010011011011110111110011111011011010101111111
01110111001100111111111011001101111001111111110011111111111100011
11101111100111111111111011111110000011111101111111111111111110111 011
01010111011101011111111111111001111111111010111101100011111111111
1011111111100101011101111111111101011110110100
Average of generation is: 359
Best so far: 387 corresponding to vector:
01011000111111111111111100010001111011011111101111111111111110111 101
11101111110110111011111111111110111110111100010101111111101101100 0
11100011111110111110010111110011110111111101100111111111110100110
10111010001111110011111011101001101011111111100110111111111101011
11111101110101111111111101110011110111111111111110110111111100111 011
11111011111101110111011111111101011111101010111111111111111110011
10101011100111110010110111111111111101011100101011111101111011111
11110111100111110110111111111111111111111110
--------------------------------------------------------
Generation 20
Best of generation is: 450 corresponding to vector:
10111111111111011111011010110111111110111111101111111111101111111
11101111110111111111111110011111111111111111111111111101100111111
11111111101111111111111110111111111110111101111111111111111111111
11111111111111111111111101111111111111111111111111111111111000111
01011011101111110111111111111111011110101111111111111111101111110 1
11111110111111010111011111011111011101111111111110111111110111111
11111111111100111111111111111111111111111011111111111111111111111
```

```
110111111111111111111111111111111111010111111110
Average of generation is: 428
Best so far: 457 corresponding to vector:
11110110111111111111111111111111011101111111111111111111011011111
01111111111101111111111111101111111111111011111111011111111111111
11111111111110111011111111111111111111111111011110111111111110111
11011101111111111111111111011111111111111101110111111111010111011
11111110101111111111111111111111111111111101110011111011110111110
11111111110111111111111111111111111111111111111011011111111111111
11111111111111011111111111011111101111111111111111110111111111111
101011111111111111111111111111101101111111111
--------------------------------------------------------
Generation 30
Best of generation is: 479 corresponding to vector:
11111111111111111111101111111111111111111111111111111111111111111
11111111111111111111111110111111111111111101111111111111111111111
01111111111111111111101111001111111111111111111111111111111111111
11111111110111111111111111111111110111111111111111111111111111111
11111111111101111111111111111111111111111111111111111111111111111
11111111111111010111111111111111111111111111111111111111111111111
11110111101101111111111111111111111111110111111011111111111110111
111111111111111111111111111111101111010111111
Average of generation is: 464
Best so far: 484 corresponding to vector:
11111111110111111111101111111111111111111111111111111111111111111
11111111111111111111111111111111111110111111111111111111111011111
11111111111111111011111111111111111111111111111111111111111111111
11111111111111111111011101111111111111111111111111111111111111111
11111111111011111111111111111111111101111111111111110111111111111
11110111111111110111111111111111111111111111111111111111111111111
11111110111111111111111111111111111111111111111011111111111111111
111111111101111111111111111111111111111111110
--------------------------------------------------------
Generation 40
Best of generation is: 489 corresponding to vector:
10111111111111111111111111111111111111111111111111111111111110111
01111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
01111111111111111111111111111111111111111111111110111111111111111
11111111111111111110111111111111111111111111111111111111111011111
11111111111111111111111111111111111111111111111111011111111101111
11111111111111110111111111111111111111111111111111111111111111111
111111111111111111111111101111111111111111111
Average of generation is: 477
```

```
Best so far: 494 corresponding to vector:
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111011111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111110111111111111111111111111111111110111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111110111111111111111111111
11111111111111111111111111111111111111111111111111111111111111110
1111111111111111111111111011111111111111111111
--------------------------------------------------------
Generation 50
Best of generation is: 492 corresponding to vector:
01111111111111111111111111111111111111010111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111110111111111111111111111111111111111111111111111111111
11111111111101111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111011111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111010111111111111111
111111111111111111111111111111111111111111111
Average of generation is: 481
Best so far: 498 corresponding to vector:
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111011111111111111111111111111111111111111111111
11111111111111111111111111111011111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111
--------------------------------------------------------
Generation 60
Best of generation is: 491 corresponding to vector:
11111111111111011111111111111111111111111111111111111110111111111
11111111111111111111111111111111111111111111111101111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111110101111111111011111111111101111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111101111111111111111111111111111111111111
11111111111111101111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111
Average of generation is: 482
Best so far: 498 corresponding to vector:
11111111111111111111111111111111111111111111111111111111111111111
```

```
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111011111111111111111111111111111111111111111111
11111111111111111111111111111011111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111
--------------------------------------------------------
Generation 70
Best of generation is: 492 corresponding to vector:
11111111111111111111111111111111111111111111111111111111111111111
11110111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111110111111111111111111111111111111
11111111111111110111111111111111111111111111111111111111111111111
11110111111111111111111111111111111111111111111111111111111111111
11111111111111111111110111111111111111111110111111111111111111111
11111111111111111111111111111111111111111111111111111101111111111
111111110111111111111111111111111111111111111
Average of generation is: 481
Best so far: 498 corresponding to vector:
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111011111111111111111111111111111111111111111111
11111111111111111111111111111011111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111
--------------------------------------------------------
Generation 80
Best of generation is: 492 corresponding to vector:
11111111111111111111111111111111111111111111111111101111111111111
11111111111110111111111111111111101111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111001111111
11111111111111111111111111111111111111111111111101101111111110111
11111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111
Average of generation is: 481
Best so far: 498 corresponding to vector:
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111011111111111111111111111111111111111111111111
```

```
11111111111111111111111111111011111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111
--------------------------------------------------------
Generation 90
Best of generation is: 492 corresponding to vector:
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111011111111111111111111111111111111111
11111111111111111111111011111110111111111011111111111111111111111
11111111110111111111111111111111111111110111111111111111111111111
11111111111110111111111111111111111111111111111111111111111101111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111
Average of generation is: 481
Best so far: 498 corresponding to vector:
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111011111111111111111111111111111111111111111111
11111111111111111111111111111011111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111
--------------------------------------------------------
Generation 100
Best of generation is: 492 corresponding to vector:
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111101111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111011111111111111111111111111111111111
11111111111111011011111111111111111111111111111111111111110111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
110111111111111111111110111110111111111111111
Average of generation is: 481
Best so far: 498 corresponding to vector:
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111011111111111111111111111111111111111111111111
11111111111111111111111111111011111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
```

```
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111
--------------------------------------------------------
```

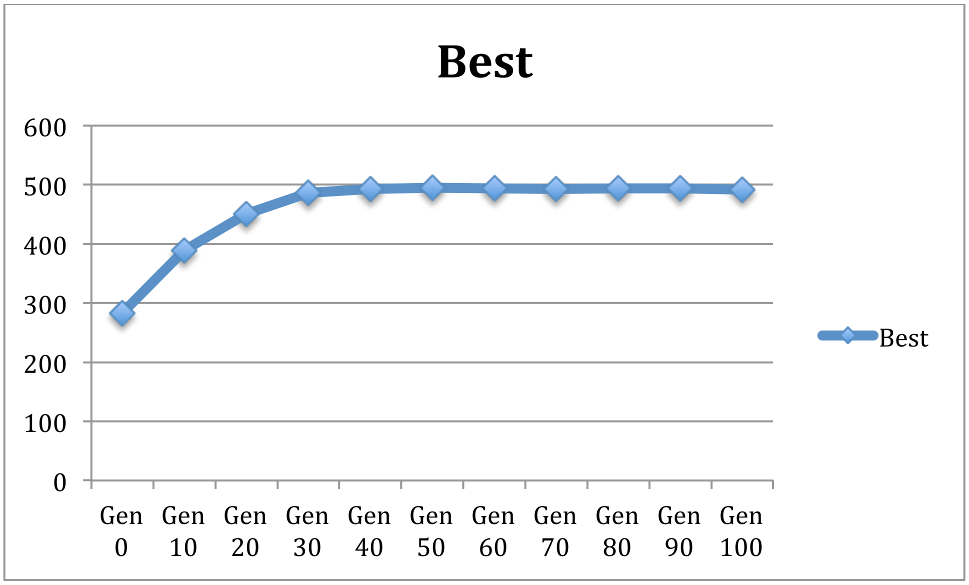
***********************************************************************************************

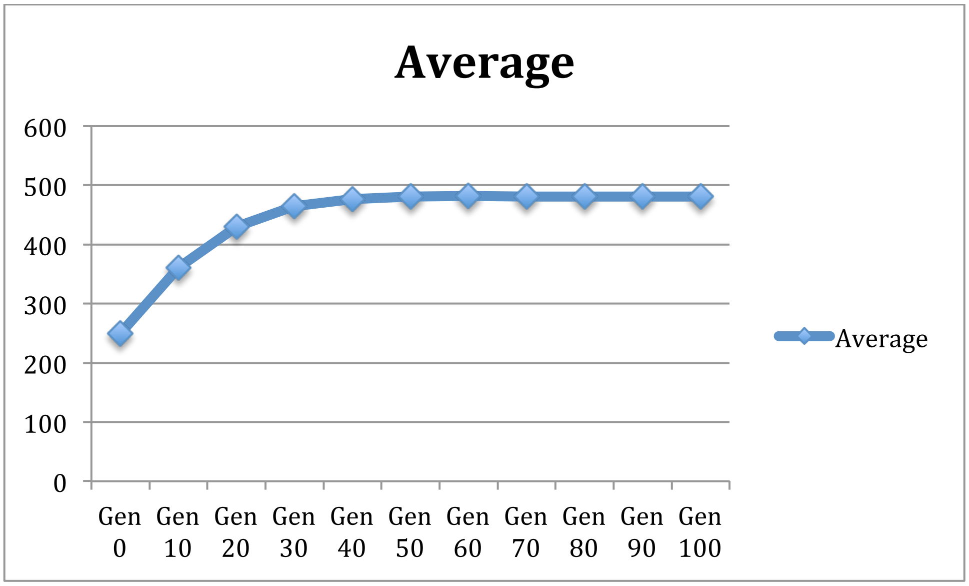
Run 1:

| Generation | Best | Average |
|---:|---:|---:|
| **0** | 283 | 249 |
| **10** | 389 | 361 |
| **20** | 451 | 430 |
| **30** | 486 | 465 |
| **40** | 493 | 477 |
| **50** | 495 | 481 |
| **60** | 494 | 482 |
| **70** | 493 | 481 |
| **80** | 494 | 481 |
| **90** | 494 | 481 |
| **100** | 492 | 481 |
| **AVERAGE** | 460.3636364 | 442.6363636 |
| **STD DEV** | 67.09735126 | 74.10030058 |

```
Best of run is: 498 corresponding to vector:
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11101111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111110111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111
N: 1000
Tmax: 100
L = 500
```

Crossover Probability: 0.8<br>
Mutation Probability: 0.02





Run 2:

```
N: 1000
Tmax: 100
L = 200
```

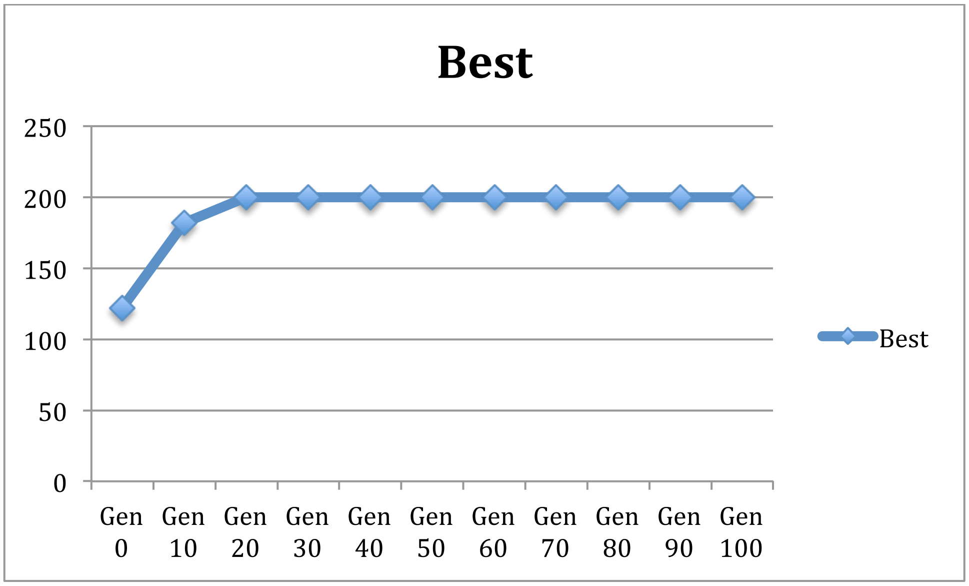
Crossover Probability: 0.8

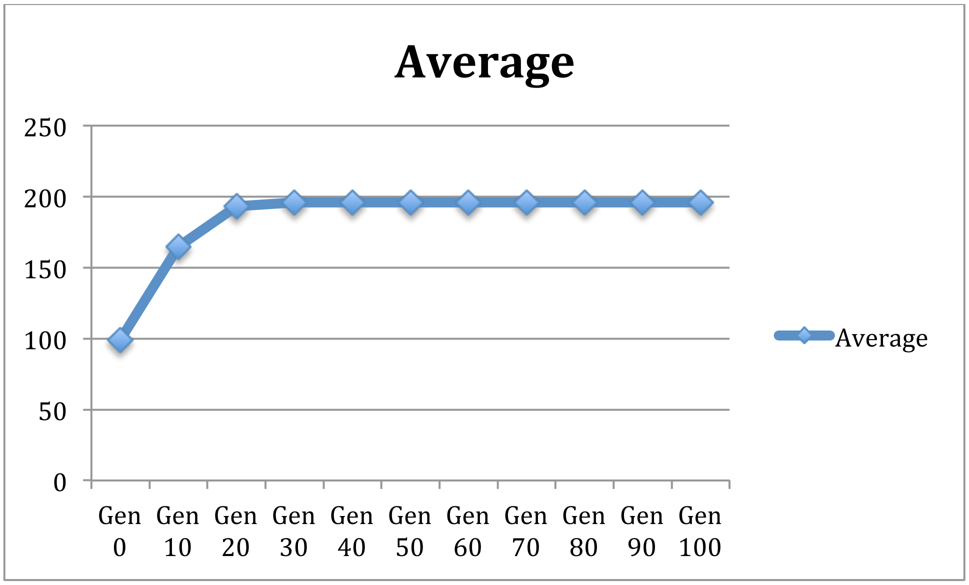
Mutation Probability: 0.02

```
Best of run is: 200 corresponding to vector:
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
```

1111111111111111111111111111111111111111111111111111111111111111
11111

| Generation | Best | Average |
|---|---|---|
| **Gen 0** | 122 | 99 |
| **Gen 10** | 182 | 165 |
| **Gen 20** | 200 | 193 |
| **Gen 30** | 200 | 196 |
| **Gen 40** | 200 | 196 |
| **Gen 50** | 200 | 196 |
| **Gen 60** | 200 | 196 |
| **Gen 70** | 200 | 196 |
| **Gen 80** | 200 | 196 |
| **Gen 90** | 200 | 196 |
| **Gen 100** | 200 | 196 |
| **AVERAGE** | 191.2727273 | 184.0909091 |
| **STD DEV** | 23.60123263 | 29.69664811 |

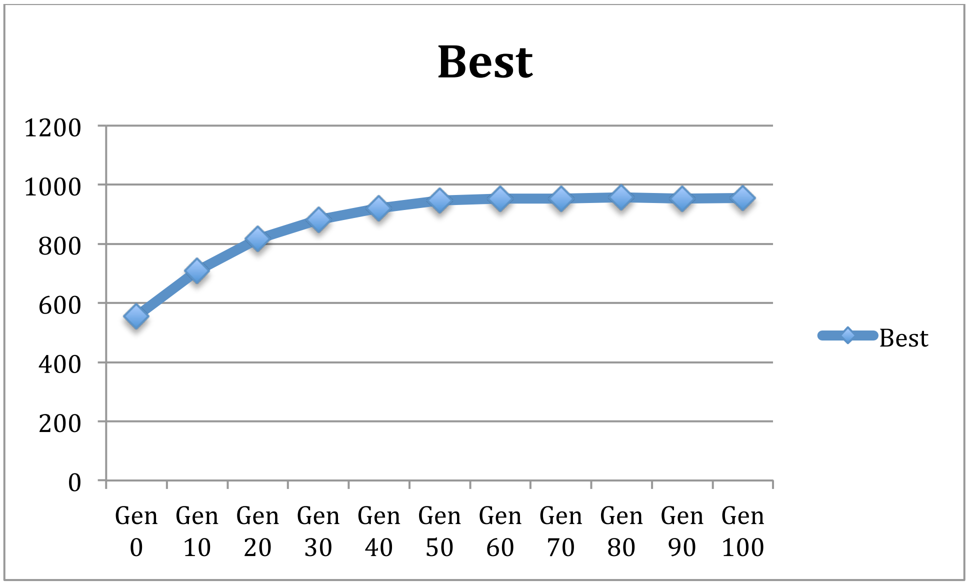


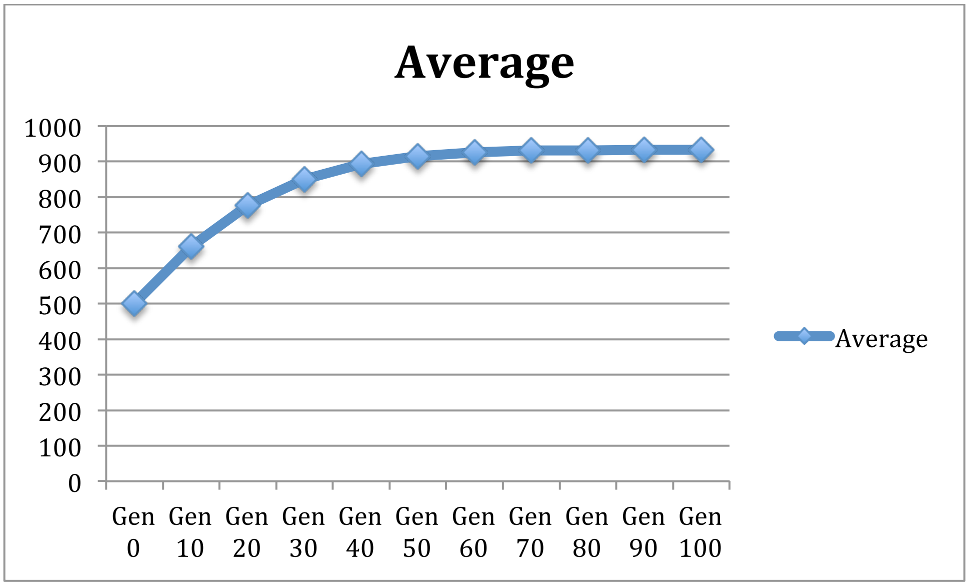


Run 3:

N: 1000
Tmax: 100
L= 1000
Crossover Probability: 0.8
Mutation Probability: 0.02
Best of run is: 971

| Generation | Best | Average |
|---|---|---|
| Gen 0 | 554 | 500 |
| Gen 10 | 709 | 661 |
| Gen 20 | 816 | 777 |
| Gen 30 | 882 | 850 |
| Gen 40 | 921 | 894 |
| Gen 50 | 946 | 916 |
| Gen 60 | 953 | 926 |
| Gen 70 | 953 | 931 |
| Gen 80 | 958 | 932 |
| Gen 90 | 953 | 933 |
| Gen 100 | 956 | 934 |
| AVERAGE | 872.8181818 | 841.2727273 |
| STD DEV | 131.5042343 | 142.1302859 |





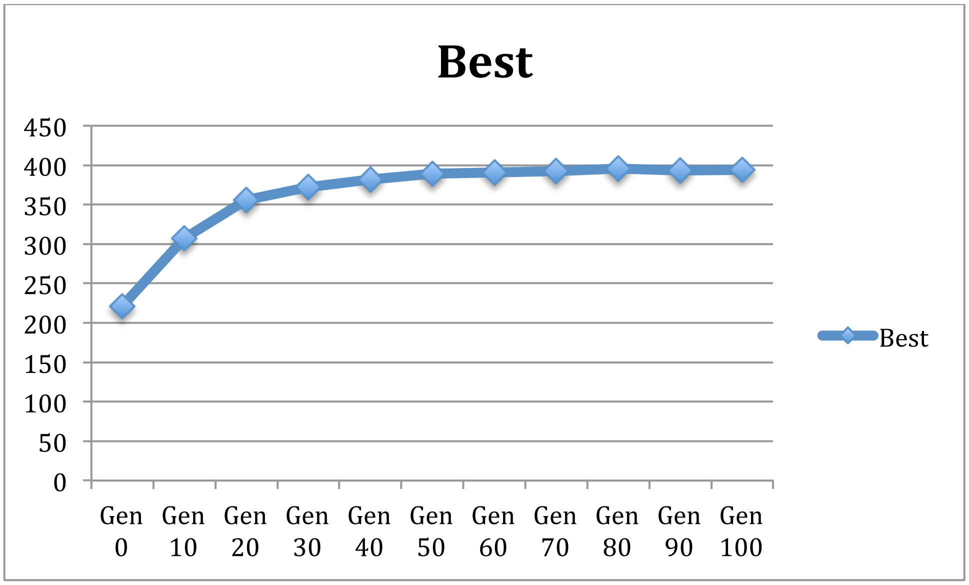
Run 4:

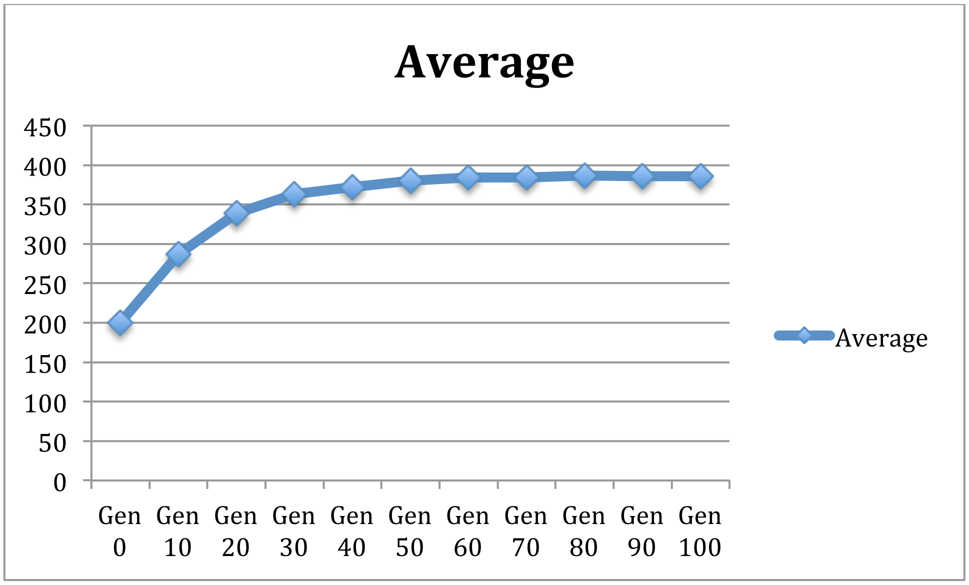
```
N: 50
Tmax: 100
L = 400
```

Crossover Probability: 0.8
Mutation Probability: 0.02

```
Best of run is: 398
```

| Generation | Best | Average |
|---|---|---|
| **Gen 0** | 221 | 200 |
| **Gen 10** | 307 | 287 |
| **Gen 20** | 356 | 339 |
| **Gen 30** | 372 | 363 |
| **Gen 40** | 382 | 372 |
| **Gen 50** | 389 | 380 |
| **Gen 60** | 391 | 384 |
| **Gen 70** | 392 | 384 |
| **Gen 80** | 396 | 387 |
| **Gen 90** | 393 | 386 |
| **Gen 100** | 394 | 386 |
| **AVERAGE** | 363 | 351.6363636 |
| **STD DEV** | 53.89062998 | 58.64004217 |




Average
450
400
350
300
250
200
150
100
50
0
Gen 0
Gen 10
Gen 20
Gen 30
Gen 40
Gen 50
Gen 60
Gen 70
Gen 80
Gen 90
Gen 100
Average