Student: Kirill Tsibikov

# 1. All One Problem

Brest 2016

Exercise 1

1. Create 100 random binary-chromosomes each with 1000 genes.

2. Fitness is the number of “1” in one chromosome – the more the better.

3. Select 2 chromosomes at random from the better half of the population.

4. Create a child chromosome by a onr-point-crossover.

5. Give the child a mutation with a probability of 1/1000 = 0.001.

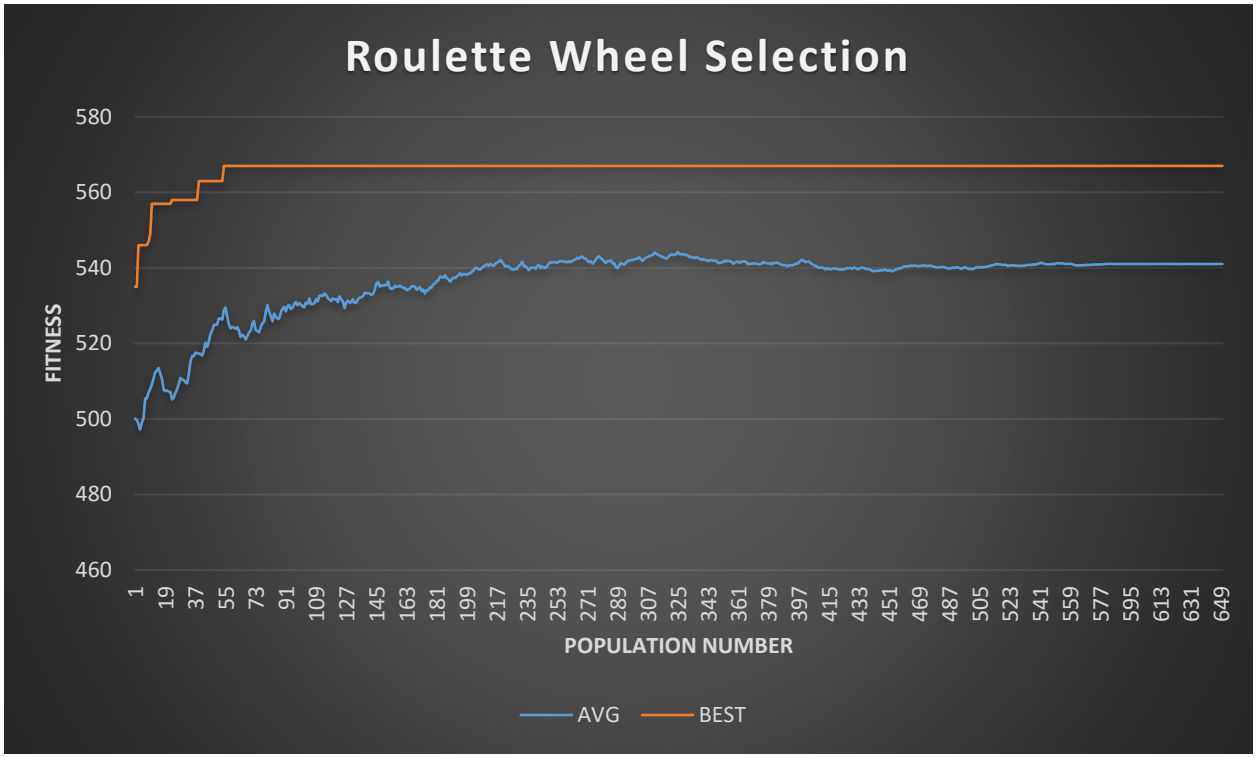
6. Repeat from 2. to 5. 100 times and create the next generation.

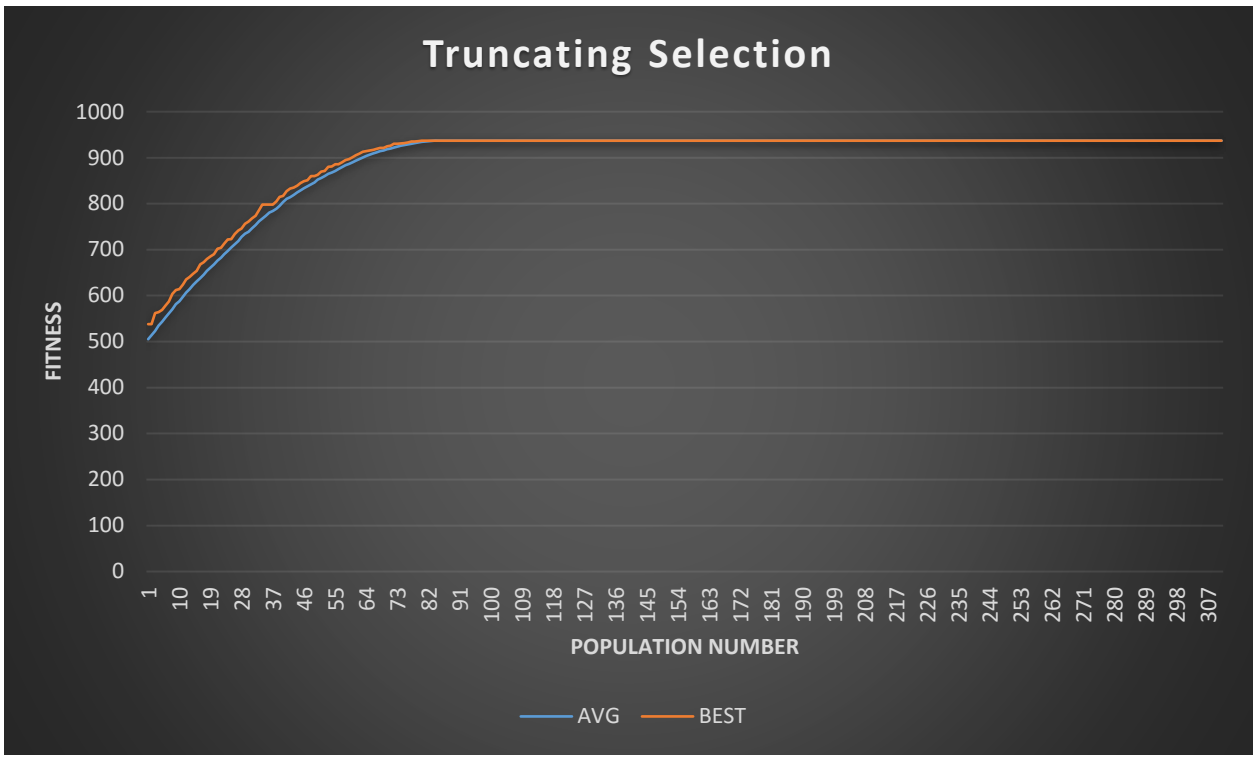
7. Repeat 6. until the fitness value does not change any more.

8. Show the result:

(1) Desplay the best chromosome in the 1st, an intermediate & final generation.

(2) Desplay the best and average fitness vs. generation.






Exercise 2

Show for the 1-st, middle and for the last generation, the number of selection for each chromosome for **Roulette Selection**.

| **The 0 population** | | | **The 500 population** | | | **The 800 population** | |
|---|---|---|---|---|---|---|---|
| **Chromosome number** | **Number of selections** | | **Chromosome number** | **Number of selections** | | **Chromosome number** | **Number of selections** |
| **1** | **2** | | **1** | **4** | | 1 | 2 |
| **2** | **1** | | **2** | **0** | | 2 | 3 |
| **3** | **2** | | **3** | **3** | | 3 | 2 |
| **4** | **1** | | **4** | **1** | | 4 | 2 |
| **5** | **1** | | **5** | **1** | | 5 | 3 |
| **6** | **1** | | **6** | **2** | | 6 | 4 |
| **7** | **3** | | **7** | **1** | | 7 | 2 |
| **8** | **0** | | **8** | **2** | | 8 | 3 |
| **9** | **3** | | **9** | **1** | | 9 | 4 |
| **10** | **1** | | **10** | **1** | | 10 | 2 |
| **11** | **0** | | **11** | **3** | | 11 | 1 |
| **12** | **2** | | **12** | **2** | | 12 | 3 |
| **13** | **4** | | **13** | **2** | | 13 | 1 |
| **14** | **2** | | **14** | **2** | | 14 | 2 |
| **15** | **4** | | **15** | **1** | | 15 | 0 |
| **16** | **2** | | **16** | **3** | | 16 | 2 |
| **17** | **0** | | **17** | **4** | | 17 | 1 |

| **18** | **2** | | **18** | **0** | | 18 | 2 |
|---|---|---|---|---|---|---|---|
| **19** | **0** | | **19** | **0** | | 19 | 1 |
| **20** | **1** | | **20** | **3** | | 20 | 2 |
| **21** | **1** | | **21** | **0** | | 21 | 3 |
| **22** | **1** | | **22** | **2** | | 22 | 2 |
| **23** | **3** | | **23** | **2** | | 23 | 1 |
| **24** | **3** | | **24** | **4** | | 24 | 2 |
| **25** | **1** | | **25** | **0** | | 25 | 2 |
| **26** | **1** | | **26** | **3** | | 26 | 1 |
| **27** | **0** | | **27** | **3** | | 27 | 0 |
| **28** | **3** | | **28** | **3** | | 28 | 1 |
| **29** | **4** | | **29** | **0** | | 29 | 2 |
| **30** | **5** | | **30** | **0** | | 30 | 3 |
| **31** | **0** | | **31** | **1** | | 31 | 3 |
| **32** | **2** | | **32** | **5** | | 32 | 2 |
| **33** | **4** | | **33** | **1** | | 33 | 1 |
| **34** | **2** | | **34** | **0** | | 34 | 0 |
| **35** | **3** | | **35** | **1** | | 35 | 1 |
| **36** | **5** | | **36** | **3** | | 36 | 1 |
| **37** | **2** | | **37** | **3** | | 37 | 0 |
| **38** | **1** | | **38** | **1** | | 38 | 4 |
| **39** | **3** | | **39** | **1** | | 39 | 1 |
| **40** | **4** | | **40** | **3** | | 40 | 1 |
| **41** | **3** | | **41** | **1** | | 41 | 0 |
| **42** | **2** | | **42** | **3** | | 42 | 2 |
| **43** | **4** | | **43** | **4** | | 43 | 1 |
| **44** | **1** | | **44** | **3** | | 44 | 1 |
| **45** | **3** | | **45** | **1** | | 45 | 0 |
| **46** | **1** | | **46** | **0** | | 46 | 1 |
| **47** | **1** | | **47** | **3** | | 47 | 0 |
| **48** | **5** | | **48** | **2** | | 48 | 1 |
| **49** | **0** | | **49** | **1** | | 49 | 2 |
| **50** | **3** | | **50** | **1** | | 50 | 4 |
| **51** | **4** | | **51** | **2** | | 51 | 1 |
| **52** | **4** | | **52** | **3** | | 52 | 2 |
| **53** | **2** | | **53** | **1** | | 53 | 5 |
| **54** | **1** | | **54** | **1** | | 54 | 2 |
| **55** | **2** | | **55** | **2** | | 55 | 3 |
| **56** | **1** | | **56** | **1** | | 56 | 4 |
| **57** | **2** | | **57** | **4** | | 57 | 1 |
| **58** | **1** | | **58** | **1** | | 58 | 2 |
| **59** | **0** | | **59** | **4** | | 59 | 1 |
| **60** | **1** | | **60** | **4** | | 60 | 3 |
| **61** | **4** | | **61** | **1** | | 61 | 2 |
| **62** | **0** | | **62** | **3** | | 62 | 3 |
| **63** | **1** | | **63** | **4** | | 63 | 1 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **64** | **1** | | **64** | **3** | | 64 | 3 |
| **65** | **2** | | **65** | **1** | | 65 | 4 |
| **66** | **3** | | **66** | **2** | | 66 | 3 |
| **67** | **3** | | **67** | **2** | | 67 | 2 |
| **68** | **3** | | **68** | **5** | | 68 | 2 |
| **69** | **0** | | **69** | **1** | | 69 | 2 |
| **70** | **1** | | **70** | **2** | | 70 | 4 |
| **71** | **2** | | **71** | **1** | | 71 | 2 |
| **72** | **2** | | **72** | **2** | | 72 | 0 |
| **73** | **2** | | **73** | **1** | | 73 | 1 |
| **74** | **5** | | **74** | **2** | | 74 | 3 |
| **75** | **0** | | **75** | **4** | | 75 | 3 |
| **76** | **4** | | **76** | **2** | | 76 | 1 |
| **77** | **1** | | **77** | **0** | | 77 | 3 |
| **78** | **1** | | **78** | **3** | | 78 | 3 |
| **79** | **4** | | **79** | **1** | | 79 | 2 |
| **80** | **3** | | **80** | **1** | | 80 | 2 |
| **81** | **3** | | **81** | **5** | | 81 | 1 |
| **82** | **2** | | **82** | **1** | | 82 | 2 |
| **83** | **3** | | **83** | **0** | | 83 | 0 |
| **84** | **1** | | **84** | **1** | | 84 | 2 |
| **85** | **1** | | **85** | **0** | | 85 | 2 |
| **86** | **0** | | **86** | **2** | | 86 | 3 |
| **87** | **3** | | **87** | **4** | | 87 | 2 |
| **88** | **2** | | **88** | **3** | | 88 | 3 |
| **89** | **3** | | **89** | **3** | | 89 | 3 |
| **90** | **1** | | **90** | **3** | | 90 | 0 |
| **91** | **3** | | **91** | **1** | | 91 | 4 |
| **92** | **3** | | **92** | **1** | | 92 | 2 |
| **93** | **2** | | **93** | **1** | | 93 | 5 |
| **94** | **1** | | **94** | **6** | | 94 | 1 |
| **95** | **2** | | **95** | **4** | | 95 | 4 |
| **96** | **0** | | **96** | **1** | | 96 | 5 |
| **97** | **2** | | **97** | **2** | | 97 | 1 |
| **98** | **1** | | **98** | **2** | | 98 | 2 |
| **99** | **1** | | **99** | **2** | | 99 | 0 |
| **100** | **3** | | **100** | **3** | | 100 | 3 |

## Listing

```
      package javaapplication1;
      import java.util.Map;
      import java.util.Random;
      class GeneticAlgo{
          boolean[][] popln;
          public enum SelectionType {
            TOURNEY, ROULETTE_WHEEL, TRUNCATING
      }
          public enum CrossingType {
            ONE_POINT_RECOMBINATION, TWO_POINT_RECOMBINATION,
ELEMENTWISE_RECOMBINATION, ONE_ELEMENT_EXCHANGE
      }
          private SelectionType slctp;
          private CrossingType crstp;
          private int genomLength; //Длина генома в битах
          private int generationCount; //Кол-во поколений
          private int individualCount; //Кол-во
Геномов(Индивидов,Особей) в поколении
          private int[] chosenchromosom;
          private SelectionType selectionType; //Тип Селекции
          private CrossingType crossingType; //Тип Скрещивания
          public GeneticAlgo(String s, String c){
              slctp=SelectionType.valueOf(s);
              crstp =CrossingType.valueOf(c);
              popln=new  boolean[100][1000];
              genomLength=1000;
              generationCount=100;
              individualCount=100;
              chosenchromosom=new int[100];
          }
          public boolean[][] run(){
              this.generateFirstGeneration();
              float ftnsmax=0;
              for(int i=0; i<100000000; i++)
              {
                  this.selection();
                  for(int j=0; j<100; j++)
                  {
                      ftnsmax=(ftnsmax<=fitnes(j))?fitnes(j):ftnsmax;
                  }
      //            System.out.print(avgfitnes());
      //            System.out.print(":");
      //            System.out.print(ftnsmax);
      //            System.out.println();
                  if(i==0||i==500||i==800)
                  {
                      System.out.println("The " + i + " population");
                      for(int j=0; j<100; j++)
                      {
                          System.out.print(j+1);
                          System.out.print(":");
                          System.out.print(this.chosenchromosom[j]);
```

```
                        System.out.println();
                    }
                }
                if(ftnsmax==1000)break;
            }

            return (new boolean[100][1000]);
        }
        private void generateFirstGeneration() {
        Random rnd=new Random();
            for(int i=0; i<100; i++)
            {
                for(int j=0; j<1000; j++)
                {
                    popln[i][j]=rnd.nextBoolean();
                }
            }
        } //генерация первого поколения
        private void selection(){
            boolean[][] genomListOffsprings=new boolean[100][1000];
            Random rndd=new Random();
            switch(this.slctp)
            {
                case ROULETTE_WHEEL:{
                    float[] wheel = new float[this.individualCount];
               wheel[0] = fitnes(0);//Значение ФитнессФункции для 1-
ого генома
                    this.chosenchromosom[0]=0;
               for (int i=1;i<this.individualCount;i++){
                         wheel[i] = wheel[i-1] + fitnes(i);//Значение
ФитнессФункции для i-ого генома
                                    this.chosenchromosom[i]=0;
                    }
               float all = wheel[this.individualCount-1];

               for (int i=0;i<this.individualCount;i++){
                         float index = Math.abs(rndd.nextFloat())*all;
                         int l = 0;
                         int r = individualCount-1;
                         int c = 0;
                         while (l < r){
                    c = (l+r) >> 1;
                    if (index <= wheel[c])
                              r = c;
                        else
                       l = c + 1;
                         }
                         int a=l;
                         index = Math.abs(rndd.nextFloat())*all;

                         l = 0;
                         r = individualCount-1;
                         c = 0;
                         while (l < r){
```

```
                    c = (l+r) >> 1;
                    if (index <= wheel[c])
                              r = c;
                         else
                       l = c + 1;
                        }
                        this.chosenchromosom[l]++;
                        this.chosenchromosom[a]++;
                        genomListOffsprings[i] = this.crossing(l,a);
                }
                    popln=genomListOffsprings;
                    break;
                 }
                 case TOURNEY:
                 {
                    for (int i=0;i<this.individualCount;i++){
                        int index1 = rndd.nextInt(individualCount);
                        int index2 = rndd.nextInt(individualCount);
                        int index3 = rndd.nextInt(individualCount);
                        int index4 = rndd.nextInt(individualCount);
                        float fr1 = fitnes(index1);
                        float fr2 = fitnes(index2);
                        index1=(fr1>fr2)?index1:index2;
                        float fr3 = fitnes(index3);
                        float fr4 = fitnes(index4);
                        index2=(fr3>fr4)?index3:index4;
                        genomListOffsprings[i] =
this.crossing(index1, index2);
                }
                    popln=genomListOffsprings;
                    break;
                 }
                 case TRUNCATING:
                 {
                    int percent=(Math.abs(rndd.nextInt())+10)%50+1;
                    this.sort();
                    for(int i=0; i<this.individualCount; i++)
                    {
                        genomListOffsprings[i] =
this.crossing((Math.abs(rndd.nextInt()))%50,(Math.abs(rndd.nextInt()))
%50);
                    }
                    popln=genomListOffsprings;
                    break;
                 }
                 default:
                    break;

          }
          } //Процедура селекци
          private boolean[] crossing(int a, int b) {
              boolean[] vec=new boolean[1000];
              switch(crstp)
              {
```

```
                case ONE_POINT_RECOMBINATION:
                {
                 for(int i=0; i<genomLength; i++)
                 {
                     Random rndd=new Random();

vec[i]=(rndd.nextBoolean())?popln[b][i]:popln[a][i];
                 }
                    break;
                }
                default:
                    break;
            }


        return vec;
        } //Процедура скрещивания
        private float fitnes(int nomber){
            float ftns=0.f;
            for(int j=0; j<genomLength; j++)
                {
                    ftns+=(this.popln[nomber][j])?1:0;
                }
            return ftns;
        } //Фитнес функция
        private float avgfitnes(){
            float ftns=0.f;
            for(int i=0; i<100; i++)
            for(int j=0; j<genomLength; j++)
                {
                    ftns+=(this.popln[i][j])?1:0;
                }
            return ftns/100;
        } //Фитнес функция
        private void sort()
        {
            for(int i=0; i<this.individualCount; i++)
            {
                boolean[] amiba;
                amiba = new boolean[1000];
                amiba=popln[i];
                double fit=fitnes(i);
                for(int j=i; j<this.individualCount; j++)
                {
                    if(fitnes(j)>fit){
                        fit=fitnes(j);
                        amiba=popln[j];
                        popln[j]=popln[i];
                        popln[i]=amiba;
                    }
                }
            }
        }
    }
```

```
public class JavaApplication1 {
    public static void main(String[] args) {
        GeneticAlgo p=new
GeneticAlgo("ROULETTE_WHEEL","ONE_POINT_RECOMBINATION" );
        p.run();
      }
}
```