CHAPTER 5
IMPLEMENTATION AND TESTING

5.1 Implementation

In the previous chapter, it was explained that these project used an arraylist data structure. The way to load the data into an arraylist is to enter each data by category into each different arraylist.

```
1. for (int i=0;i<popsize ;i++ ) {
2.                 chrome = new Chromosome(4);
3.                 chrome.randomize();
4.                 populasi[i] = chrome;
5. }
```

The next step is to generate the initial population as much as popsize number. Then make 4 chromosomes for each population because the data is divided into 4 categories. Then fill the chromosome with data randomized used the randomize function like the sample of code above. The way is that each of the chromosomes sequentially stores random data according to each category. For example : the first chromosome stores random data from staple food, the second chromosome stores random data from side dishes, the third chromosome stores random data from vegetables, and the fourth chromosome stores random data from fruit. The initial population is carried out with pattern like the example above and carried out as much as the number of popsize. So the result is that the initial populaiton will appear as many as number of popsize with each chromosome consisting of 4 different food categories. After generating the initial populaiton, the next step is to calculate the calorie count of diabetics patient with the data inputted by the patient. The following is a sample code for calculating calories for men and women.

```
6. if (jk.equals("l")){
7.         totalkalorisehari = (66+((13.7*bb)+(5*tb)-(6.8*umur)));
8. } else if (jk.equals("p")){
9.         totalkalorisehari = (655+(9.6*bb)+(1.8*tb)-(4.7*umur));
10.     }
```

The parameter used to calculate the calories of diabetics are weight, height, age, and gender. For the calculation itself as explained in the previous chapter that the formula for calculating calories between men and women is different. The formula is :

Calculation calories formula for men :

$$66+(13{,}7*berat\,badan)+(5*tinggi\,badan)-(6.8*usia)$$

Calculation calories formula for women :

$$655+(9.6*berat\,badan)+(1{,}7*tinggi\,badan)-(4{,}7*usia)$$

The next step is to calculate the fitness value of each chromosome of oeach population. The calculation is to add the total of nutrient from each food category. Then check the total of each nutrient whether it meets the specified classification and after that will be given penalty value. The value of the penalty given to the chromosomes that meet the criteria will be smaller than the value of the penalty given to the chromosomes that do not meet the criteria. The reason is because the result total penalty of each chromosome will be divided by 1, so that later it will get the maximum value of fitness. After getting the value of fitness , then looking for the maximum fitness value for each chromosome are carried out as much as the populaiton number. If have got the maximum value of fitness, the chromosome that have the fitness value becomes the optimal food choice, then looking for the alternative foods. The method is the value of each fitness of each chromosome compared to the maximum fitness value that has been obtained, if the value is the same then the populaiton that has the fitness value will be used as alternative food. Then look for the average fitness value of each generation of the populaiton. The way is divided between the total fitness value of each chromosome from the each population with the number of population. Besides, it also stores the maximum fitness value of population of each generation.

The next part is the part which is the core of genetic algorithm, which is doing the crossover process, the mutation process, and the selection process. The

first is crossover process. This crossover process is carried out by crossing between 2 chromosomes. The crossover used is the one-cut of point, meaning that two chromosomes are crossed with the cut point. The chromosomes that will be crossed are obtained from random, random also done to get the cut point.

```
11.      for (int i=0;i<offspring_cutpoint ;i++ ) {
12.                      child.genes[k] = chrome1.genes[k];
13.                      k++;
14.      }
15.      for (int j=offspring_cutpoint;j<child.genes.length ;j++ )
  {
16.                      child.genes[k] = chrome2.genes[j];
17.                      k++;
18.      }
```

After obtained 2 chromosomes and 1 cut point, the next step is to choose the chromosomes that will be used as parent and which ones will be used as child. After being determined parent and child of chromosome, then directly make a cross between the two chromosomes. The crossing is done with the cut point that has obtained from the random result earlier. The way is that the chromosome that are used as parent are cut until the cut point and the rest are filled with chromosome that are made into child. Above is an example code for crossover process. If the crossover process is complete then proceed to the second part, namely the mutation process. As explained in the previous chapter, the mutation process used in this project is slightly different because the storage model is carried out by storing each chromosome consisting of 4 fodd categories. The mutation method that used is *exchange mutation*. This mutation process has the idea of exchanging contents provided that they have the same category of 2 chromosomes. The chromosomes that will be exchanged for their contents are obtained randomly. Similarly, the ctegories that will be raandomly exchanged are obtained.

```
19.      if(i == p1){
20.              chrome.genes[i] = chrome2.genes[i];
21.      } else {
22.              chrome.genes[i] = chrome1.genes[i];
23.      }
```

After obtained 2 random chromosomes and random categories as determinants of chromosome exchange, then directly exchange according to 2 chromosomes according to the categories that obtained from randomization. Then goes to the third part, namely slection process. The selection used in these project uses 2 kind of selection which later the user can choose which selection will be according to user wishes. The selections are Elistism and RouletteWheel. Elitism selection is a selection that looks for the best fitness value that will be used as the initial population for the next generation. The first thing to do in elitism selection is to calculate fitness value of each individual from the population that has passed the crossover and mutation process.

```
24.     if (largepop[i].fitness > largepop[j].fitness) {
25.         tmp = largepop[i];
26.         largepop[i] = largepop[j];
27.         largepop[j] = tmp;
28.     }
```

Then look for the best value of fitness for each individual and store it for use as the nitial population in the next generation. Whereas if using RouletteWheel selection is a little different. The RouletteWheel selection has the idea that each individual has the same opportunity to be the initial population in the next generation. First of all what is done the RouletteWheel selection is calculating the fitness value and looking for the total fitness value of the population that has passed the crossover and mutation process. Then look for the cumulative probability values of each individual.

```
29.     while(count < popsize && random > probabilityCumulative[count]){
30.         count++;
31.     }
```

From each those cumulative probabilities value compared to random value. If the value of the cumulative probability is greater than the random value, the individual who has the cumulative probability value will be used as the initial population for the next generation.

5.2 Testing

Testing is done with 2 trials. Each experiment was conducted twice with different selections. The first test was carried out in 500 generations and continued with 1000 generations.

Information about testing :

- Diabetics name : Rani
- Age : 53 y.o.
- Gender : Female
- Height : 157 cm
- Weight : 66 kg
- Number of Generation : 500 & 1000
- Selection Type : RouletteWheel & Elitism

Testing of 500 generations with RouletteWheel Selection

Illustration 5.1: Testing using RouletteWheel Selection with 500 Generation

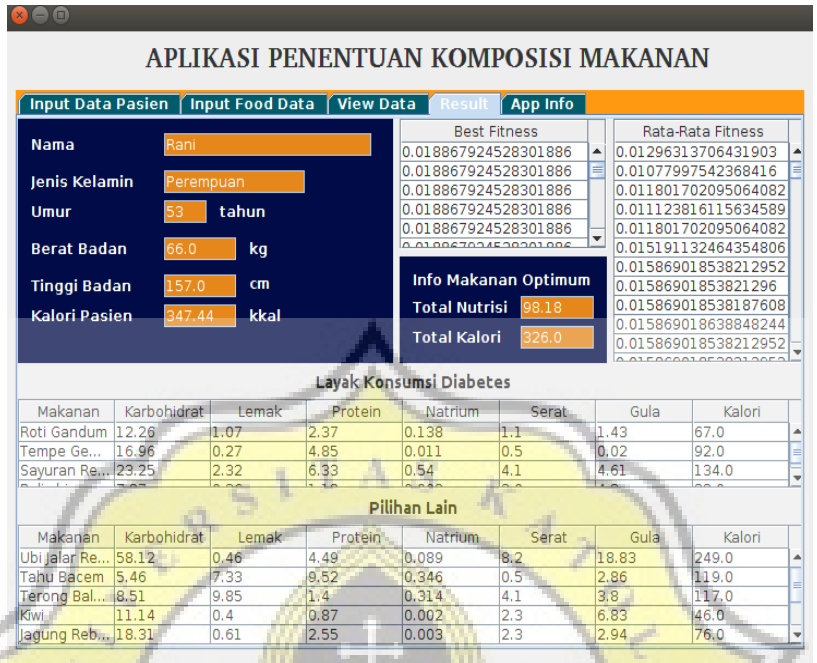

Illustration 5.2: Result of Testing using RouletteWheel Selection with 500 Generation

The above is the result of testing the data using the RouletteWheel selection and generation as many as 500. For the average value of fitness varies greatly from 0.01296313706431903, 0.01077997542368416, 0.011801702095064082, 0.011123816115634589, 0.011801702095064082, 0.015191132464354806, 0.015869018538212952 etc. While the value of the best fitness is 0.018867924528301886. The length of the process is 0.274 seconds. The calories obtained from the results of a combination of foods that have been done is 326.0 kcal. The following is a graph of the results of testing 500 generations with RouletteWheel selection.

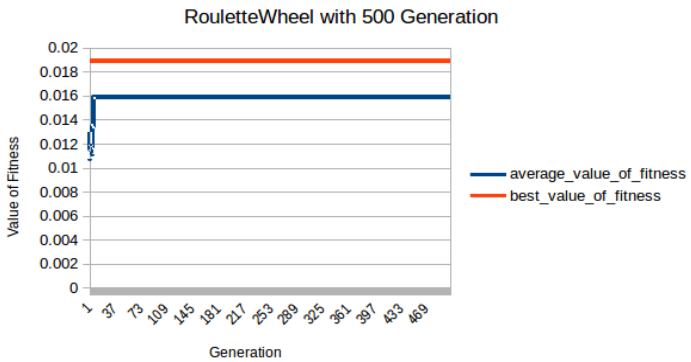

Illustration 5.3: Graph of Result from Testing using RouletteWheel Selection with 500 Generation

From the illustration 5.3 above it can be seen that average value of fitness always increase.

Testing of 500 generations with Elitism Selection

Illustration 5.4: Testing using Elitism Selection with 500 Generation

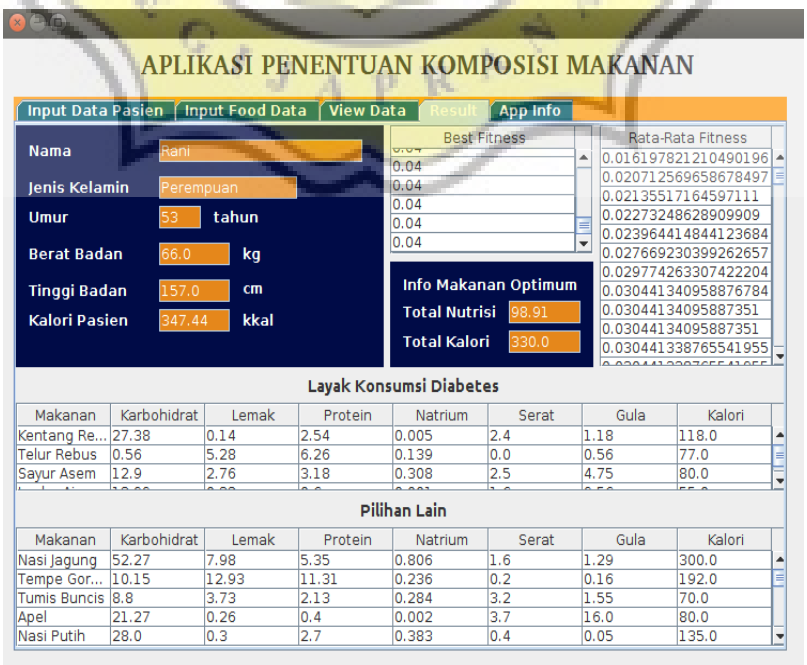

Illustration 5.5: Result of Testing using Elitism Selection with 500 Generation

The above is the result of testing the data using Elitism selection and the generation as much as 500. For the average value of fitness is not so varied, for example 0.016197821210490196, 0.023964414844123684, 0.027669230399262657, 0.029774263307422204, 0.030441340958871875, 0.03360627421999008, 0.035716244024907653, 0.03888118746090656, 0.03993615311035946, 0.039936153110307696, etc. While the value of the best fitness is 0.04. The length of the process is 0.28 second. The calories obtained from the results of a combination of foods that have been done is 330.0 kcal. The following is a graph of the results of testing 500 generations with Elitism selection.

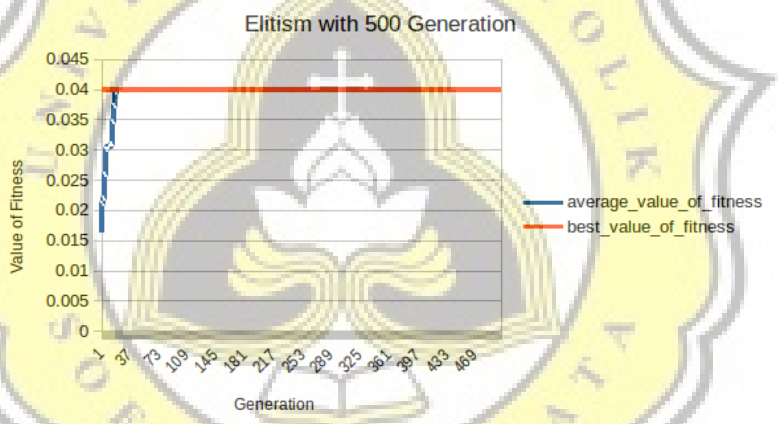

Illustration 5.6: Graph of Result from Testing using Elitism Selection with 500 Generation

From the illustration 5.6 above it can be seen that average value of fitness tends to be constant increase.

From the two types of selection above have different results for the speed, best fitness, average fitness, and of course calories results. It can be seen that Elitism selection got value of fitness average with a longer processing time of 0.28 seconds compared to RouletteWheel selection which only takes 0.274 seconds. That mean the RouletteWheel faster than Elitism, only a difference of 0.006 seconds. It is known that the total calories needed by the patient is 347.44 kcal, so that the calorific value obtained between RouletteWheel and Elitism is more optimal using Elitism.

After testing with 500 generations, it will now be tested with 1000 generations. Testing of 1000 generations with RouletteWheel Selection.

Illustration 5.7: Testing using RouletteWheel Selection with 1000 Generation

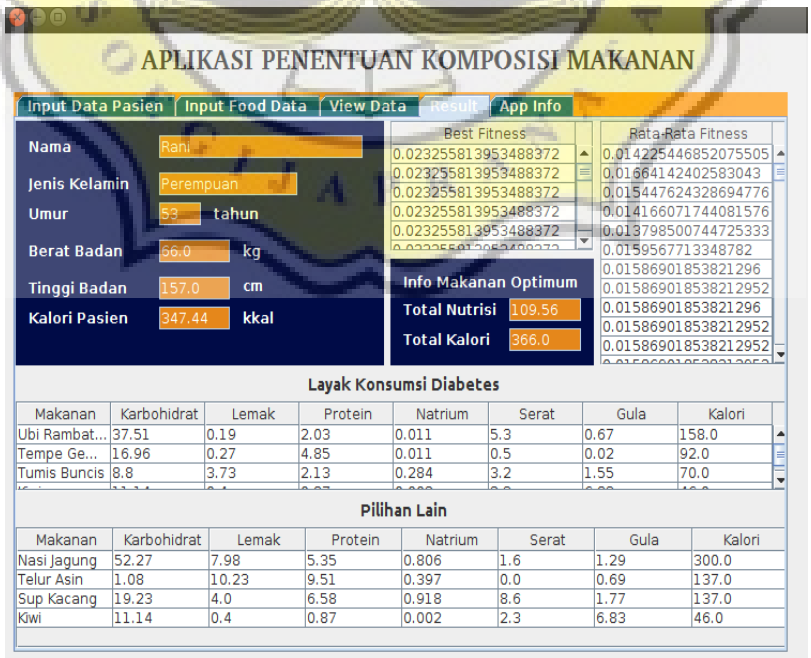

Illustration 5.8: Result of Testing using RouletteWheel Selection with 1000 Generation

The above is the result of testing data using RouletteWheel selection and 1000 generation. For the average value of fitness it varies so much, for example 0.014225446852075505, 0.01664142402583043, 0.015447624328694776, 0.014166071744081576, 0.013798500744725333, 0.0159567713348782, 0.01586901853821296, 0.015869018538212952, etc. It can also be seen that the value of the average fitness always increases but not too much. While the value of the best fitness is 0.023255813953488372. The process length is 1.118 seconds. The calorie value obtained is quite optimal, that is 366.0 kcal.

The following is a graph of the results of testing 1000 generations with RouletteWheel selection. From the following graph it can be seen that the average fitness value has decreased but then increased again.

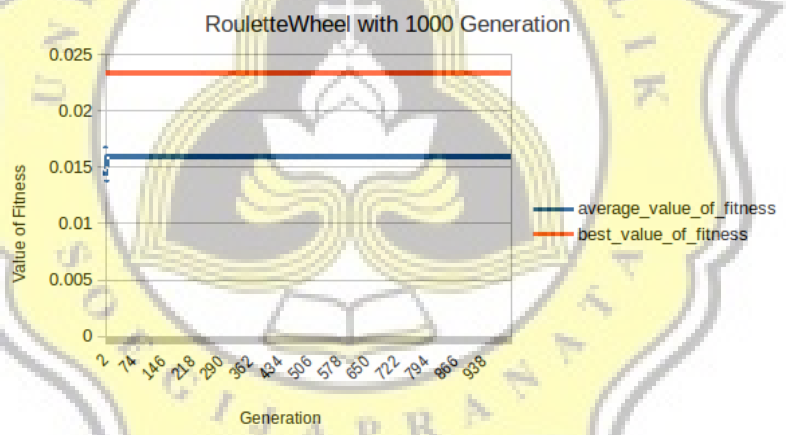

Illustration 5.9: Graph of Result from Testing using RouletteWheel Selection with 1000 Generation

Testing of 1000 generations with Elitism Selection

Illustration 5.10: Testing using Elitism Selection with 1000 Generation

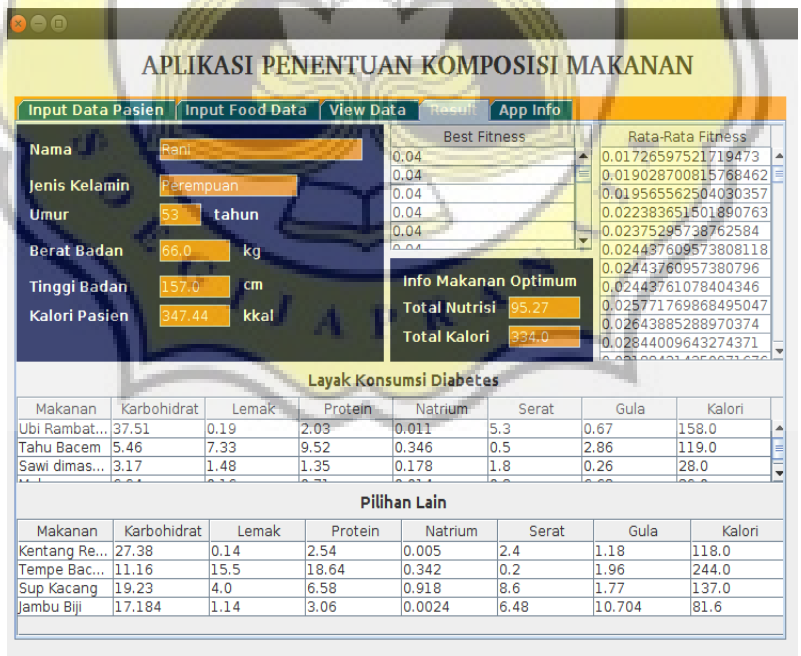

Illustration 5.11: Result of Testing using Elitism Selection with 1000 Generation

The above is the result of testing data using Elitism selection and generation as many as 1000. For the average value of fitness is quite varied and

always increased, for example 0.01726597521719473, 0.019565562504030357, 0.022383651501890763, 0.025771769868495047, 0.02643885288970374, 0.03360628658810373, 0.039936153110385335, 0.039936153110307696, etc. While the value of the best fitness is 0.04. The length of the process is 0.564 second. The calorie value obtained is only 334.0 kcal. The following is a graph of the results of testing 1000 generations with Elitism selection.

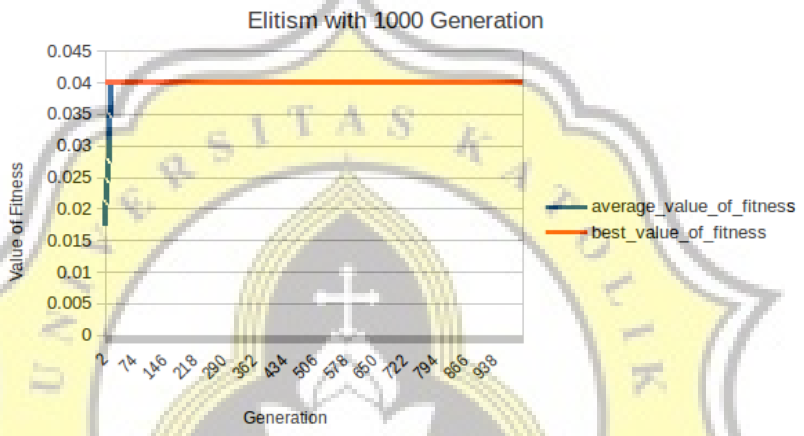

Illustration 5.12: Graph of Result from Testing using Elitism Selection with 1000 Generation

The results of the elitism selection using 1000 generations are not much different from the elitism selection that used 500 generations. The difference is only in the results of best fitness value and average fitness value.

Table 5.1: Result of Testing

	500 Generations		1000 Generations	
	RouletteWheel	Elitism	RouletteWheel	Elitism
Best Fitness	0.0188679.....	0.04	0. 02325.....	0. 04
Average Fitness	• 0.0126119... • 0.0118017... • 0.0158690...	• 0.01619... • 0.02396... • 0.03049...	• 0.01422... • 0.01664... • 0.01379...	• 0.01726... • 0.02577... • 0.03360...
Calories	326 kcal	330 kcal	366.0 kcal	334.0 kcal
Time	0.274 second	0.28 second	1.118 second	0.564 second

From the testing that has been done which uses 2 different types of selection, each of which has two different generations.

To make sure result of the testing, so these project conducted an experiment 100 times in each selection with 1000 generations.

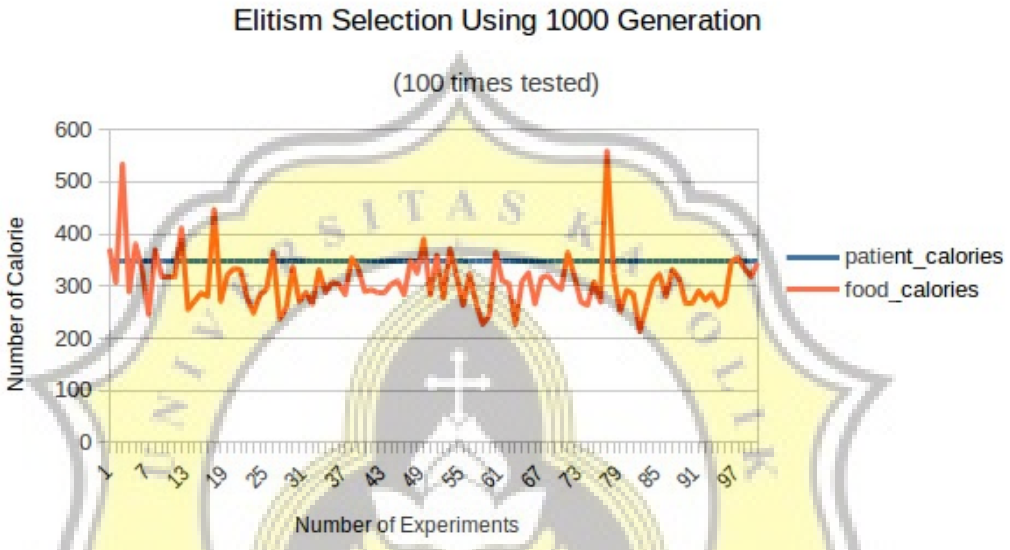

Illustration 5.13: Graph Testing Elitism Selection with 1000 Generation (100 times)

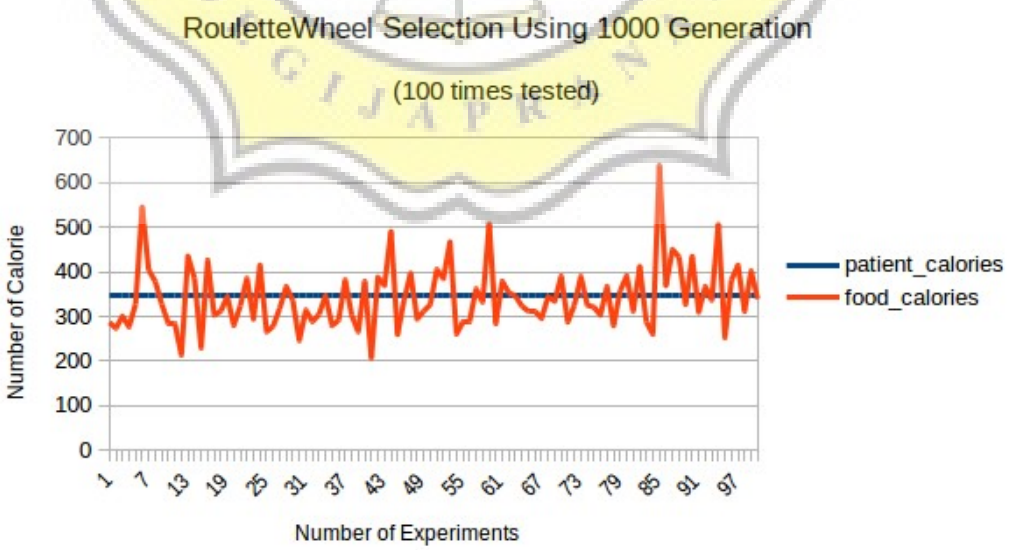

Illustration 5.14: Graph Testing RouletteWheel Selection with 1000 Generation (100 times)

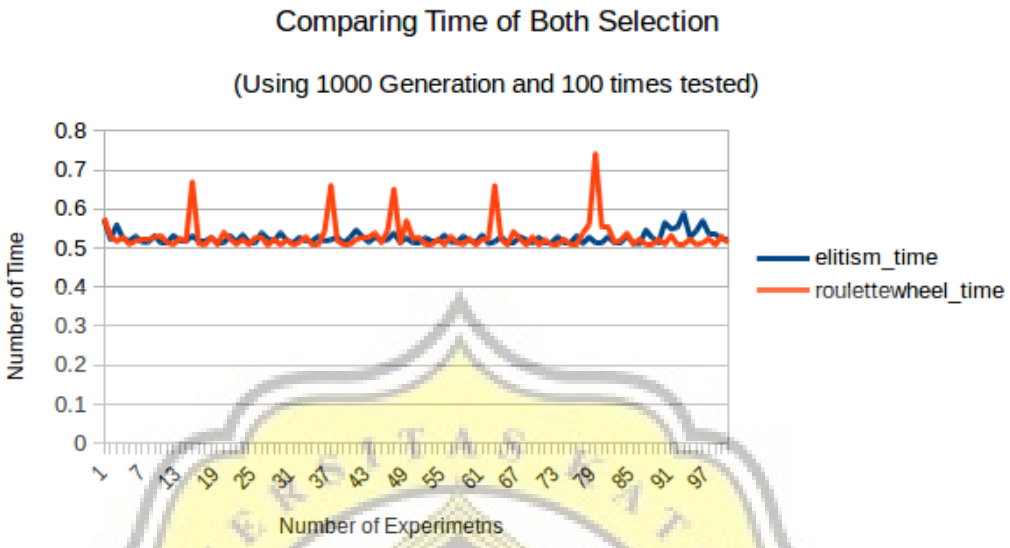

Illustration 5.15: Graph of Result Comparing Time of Both Selection in 1000 Generations

If viewed from the speed, it will be faster to use Elitism selection with a percentage difference of 0.00374%, besides that if viewed from the side of optimization, both are quite optimal but will be more optimal if also using Elitism selection with a percentage 36%.