EE449

Homework 2 – Evolutionary Algorithms

Due: 23:55, 22/05/2022

Berkay İPEK

2304814

Sec1

Table of Contents

1. Experimental Results

1.1 Number of Individuals

After first 1000th generation, results are:

Figure 1: Result at 1000th generation (# of individuals = 5)

Figure 2: Result at 1000th generation (# of individuals = 10)

Figure 3: Result at 1000th generation (# of individuals = 20)

Figure 4: Result at 1000th generation (# of individuals = 50)

Figure 5: Result at 1000th generation (# of individuals = 75)

In figures below, as you can go from left upper corner to right bottom corner, number of generations is increased by 1000.

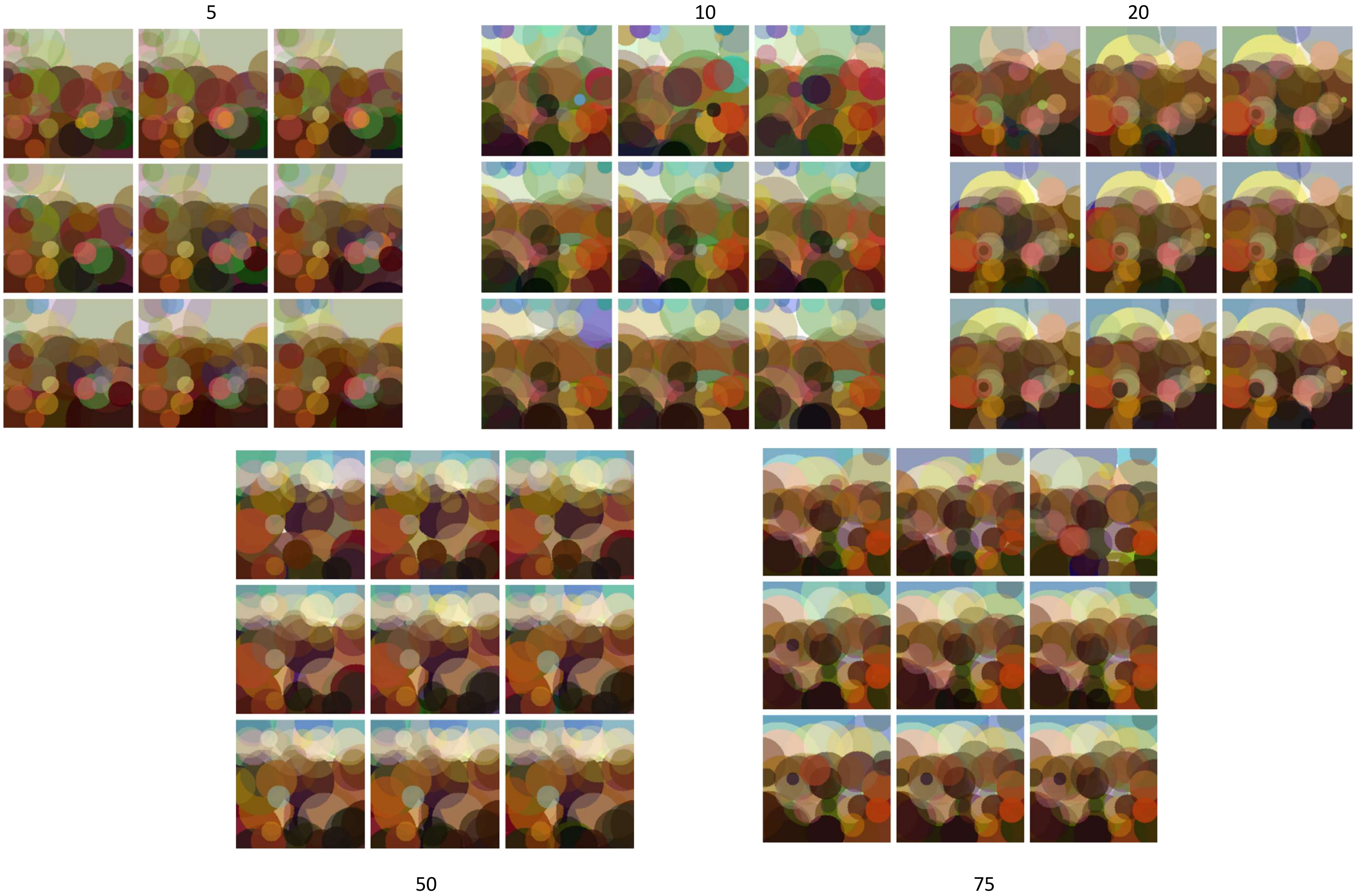


Figure 6: Result at each 1000th generation with having different individual numbers

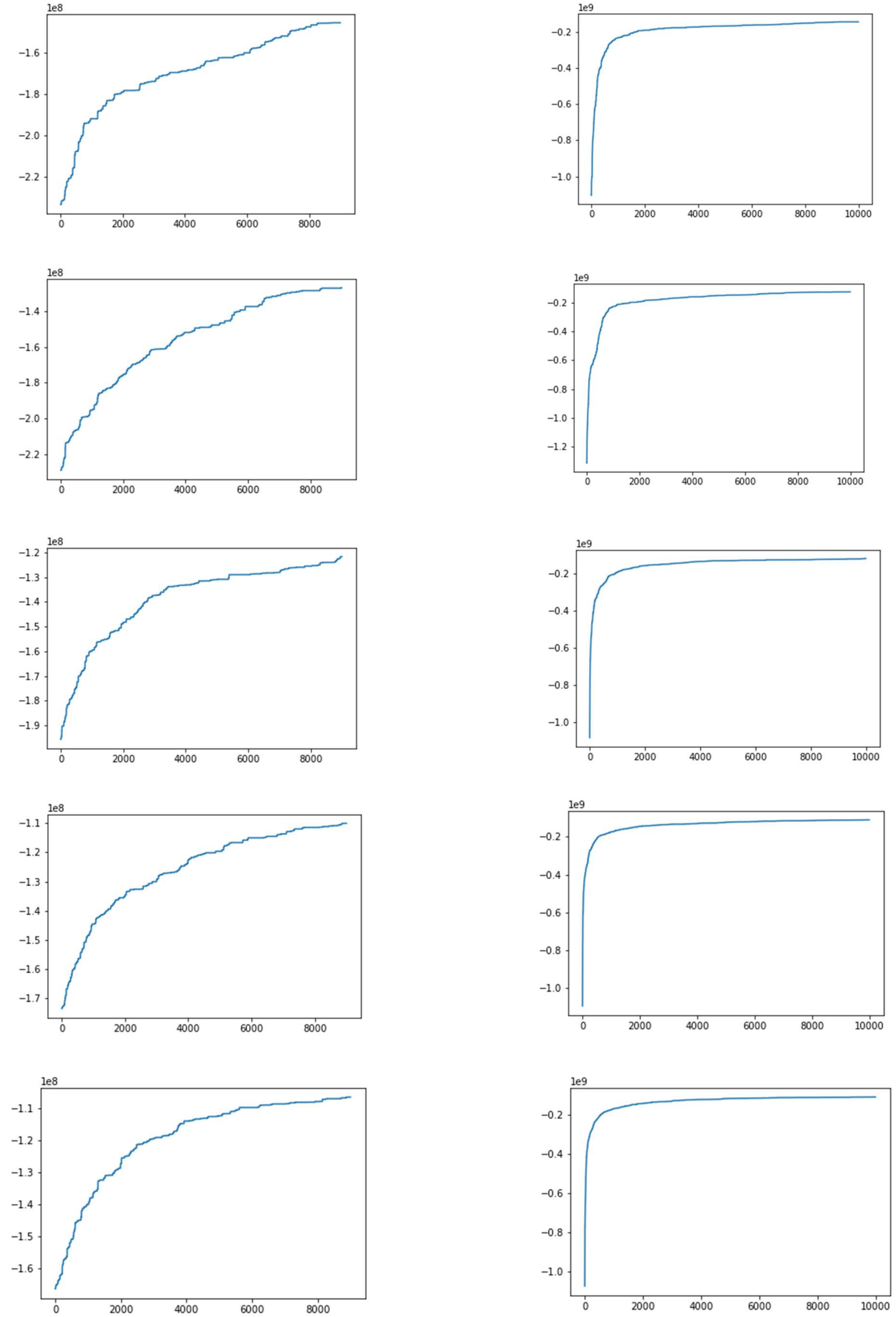


Figure 7: Fitness Values at each iteration taken for different number of individuals (left = 1000-10000, right=1-10000)

(Since there is a default case, I used same results for that case (num_of_indv=20))

(from upper to bottom, it is in descending order)

1.2 Number of Genes

After first 1000th generation, results are:

Figure 8: Result at 1000th generation (# of genes = 10)

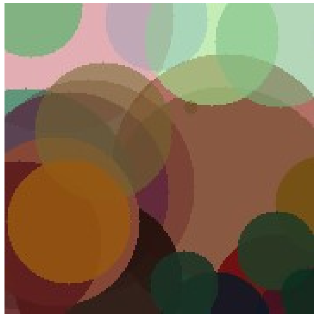


Figure 9: Result at 1000th generation (# of genes = 25)

Figure 10: Result at 1000th generation (# of genes = 50)

Figure 11: Result at 1000th generation (# of genes = 100)

Figure 12: Result at 1000th generation (# of genes = 150)

In figures below, as you can go from left upper corner to right bottom corner, number of generations is increased by 1000.

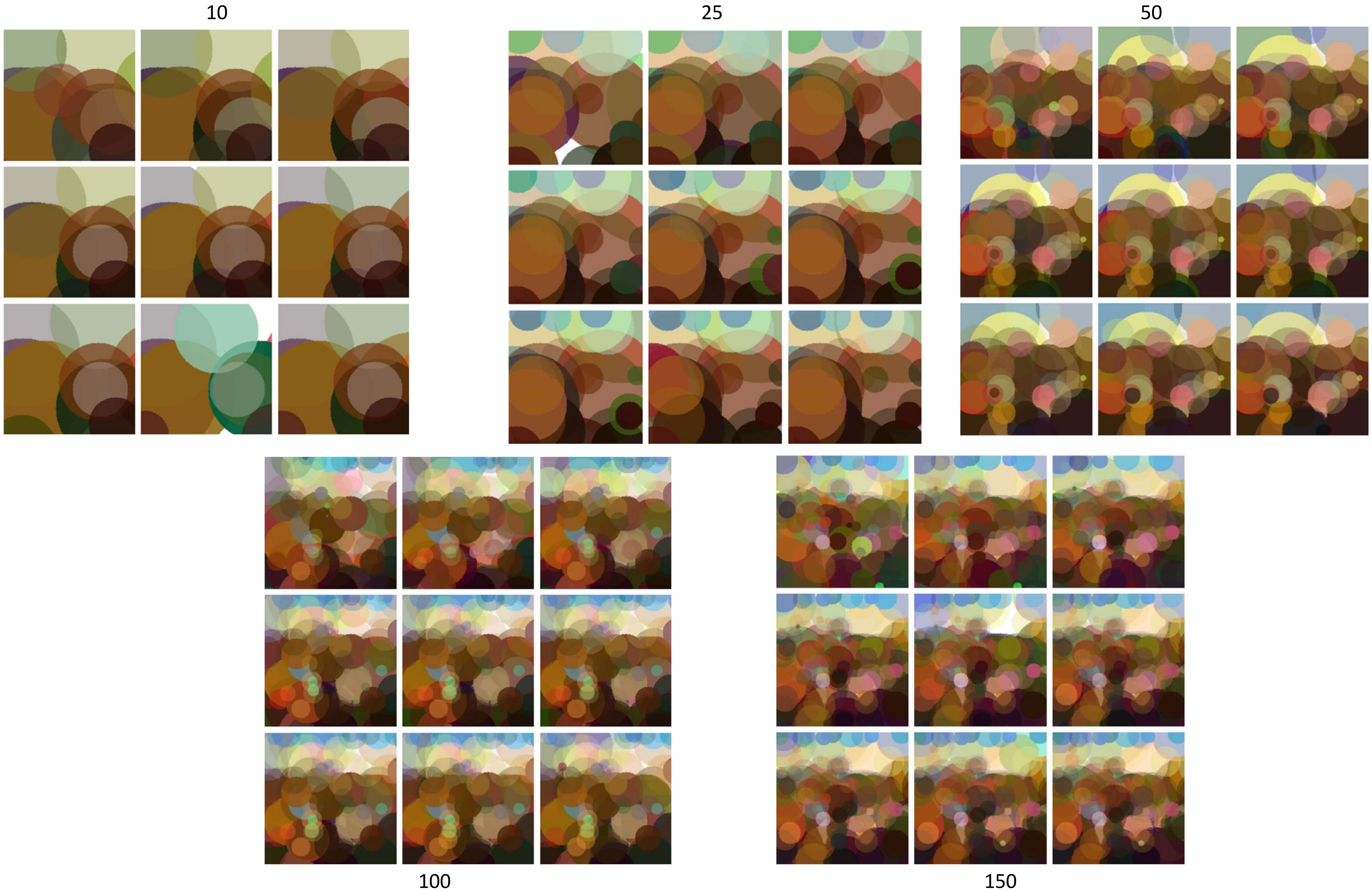


Figure 13: Result at each 1000th generation with having different number of genes

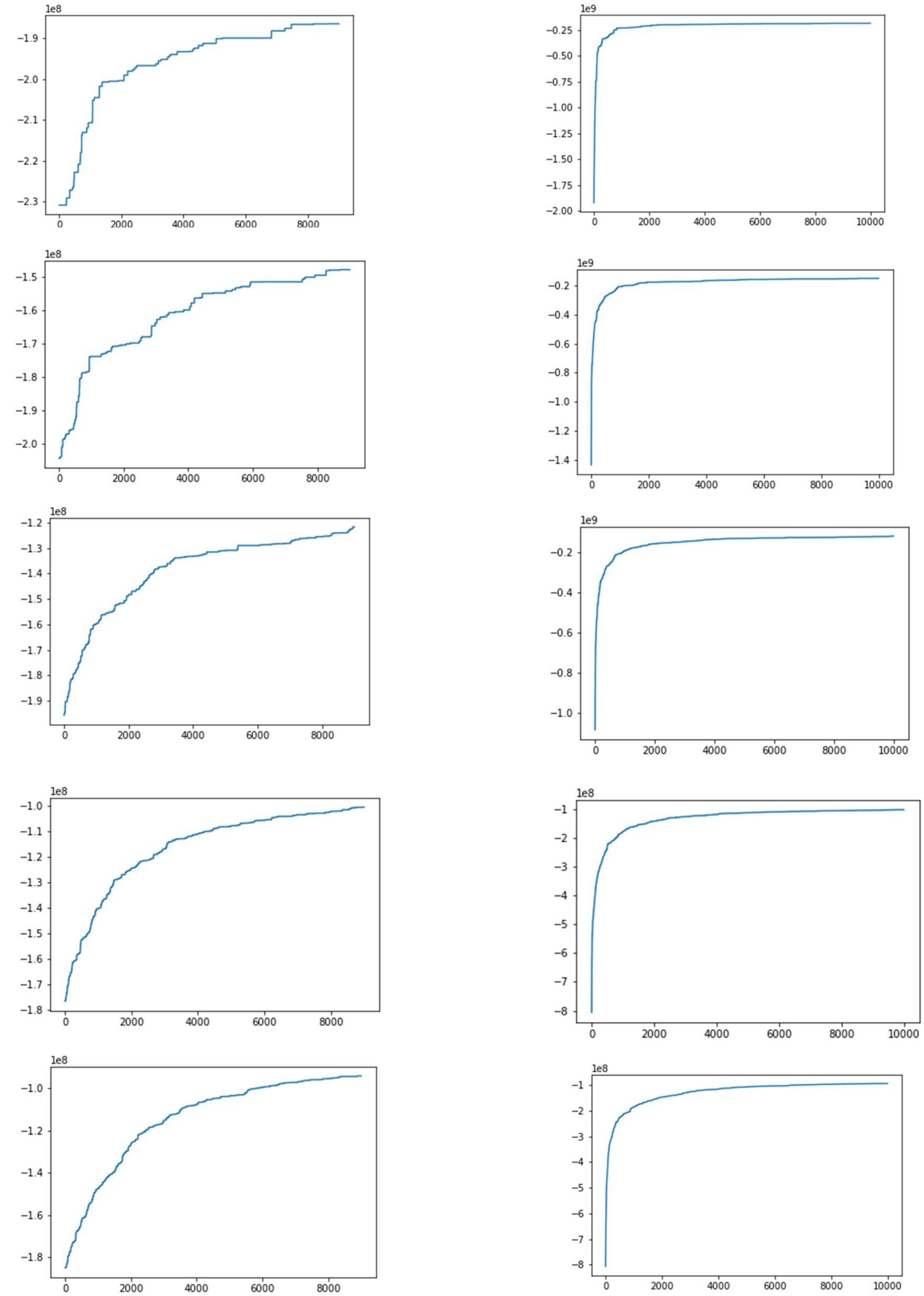


Figure 14: Fitness Values at each iteration taken for different number of genes (left = 1000-10000, right=1-10000)

(Since there is a default case, I used same results for that case (num_of_genes=50),from upper to bottom it is in descending order of the parameter)

1.3 Tournament Size

After first 1000th generation, results are:

Figure 15: Result at 1000th generation (size of tournament= 2)

Figure 16: Result at 1000th generation (size of tournament = 5)

Figure 17: Result at 1000th generation (size of tournament =10)

Figure 18: Result at 1000th generation (size of tournament = 20)

In figures below, as you can go from left upper corner to right bottom corner, number of generations is increased by 1000.

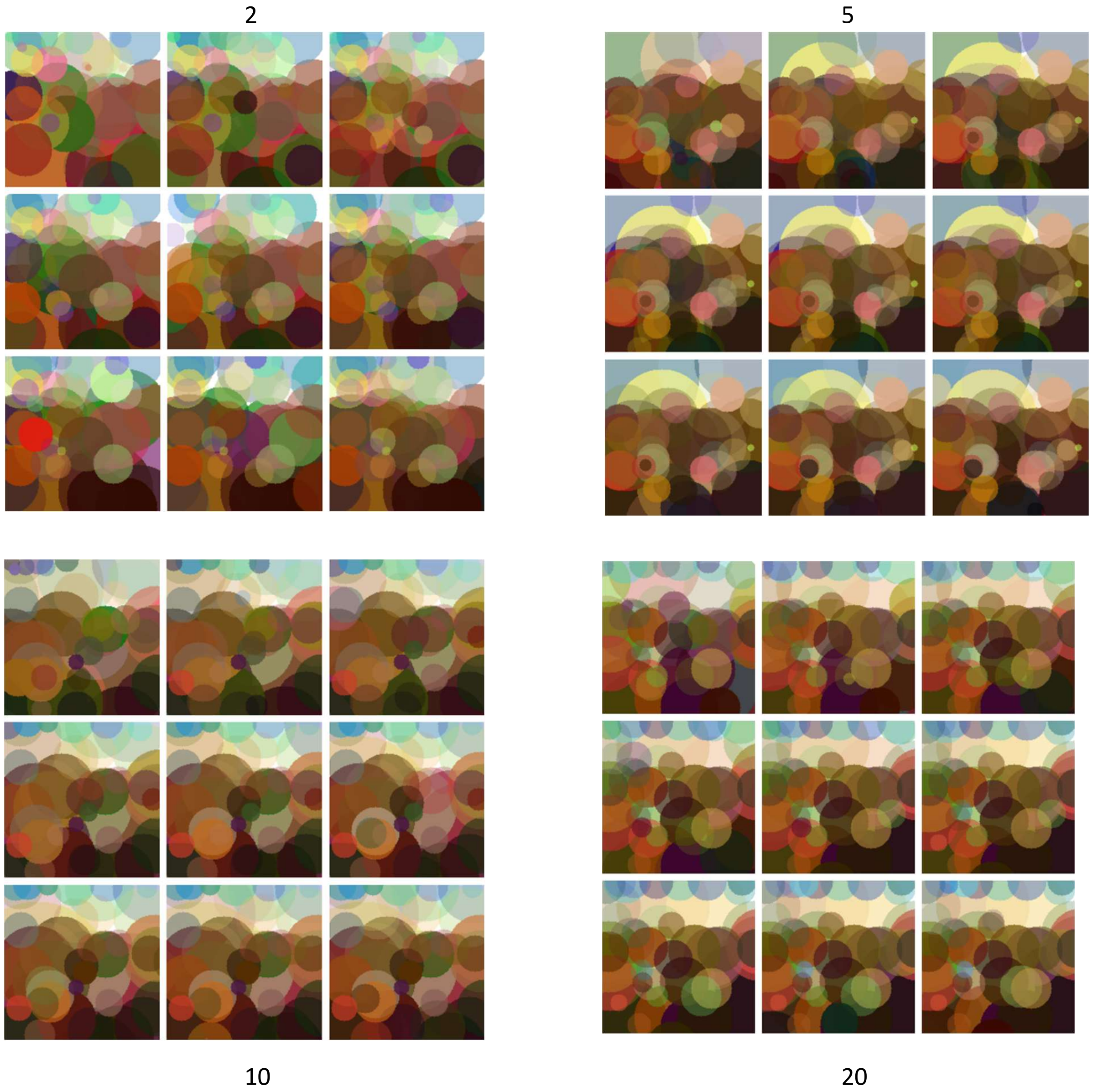


Figure 19: Result at each 1000th generation with having different number of sizes of tournament

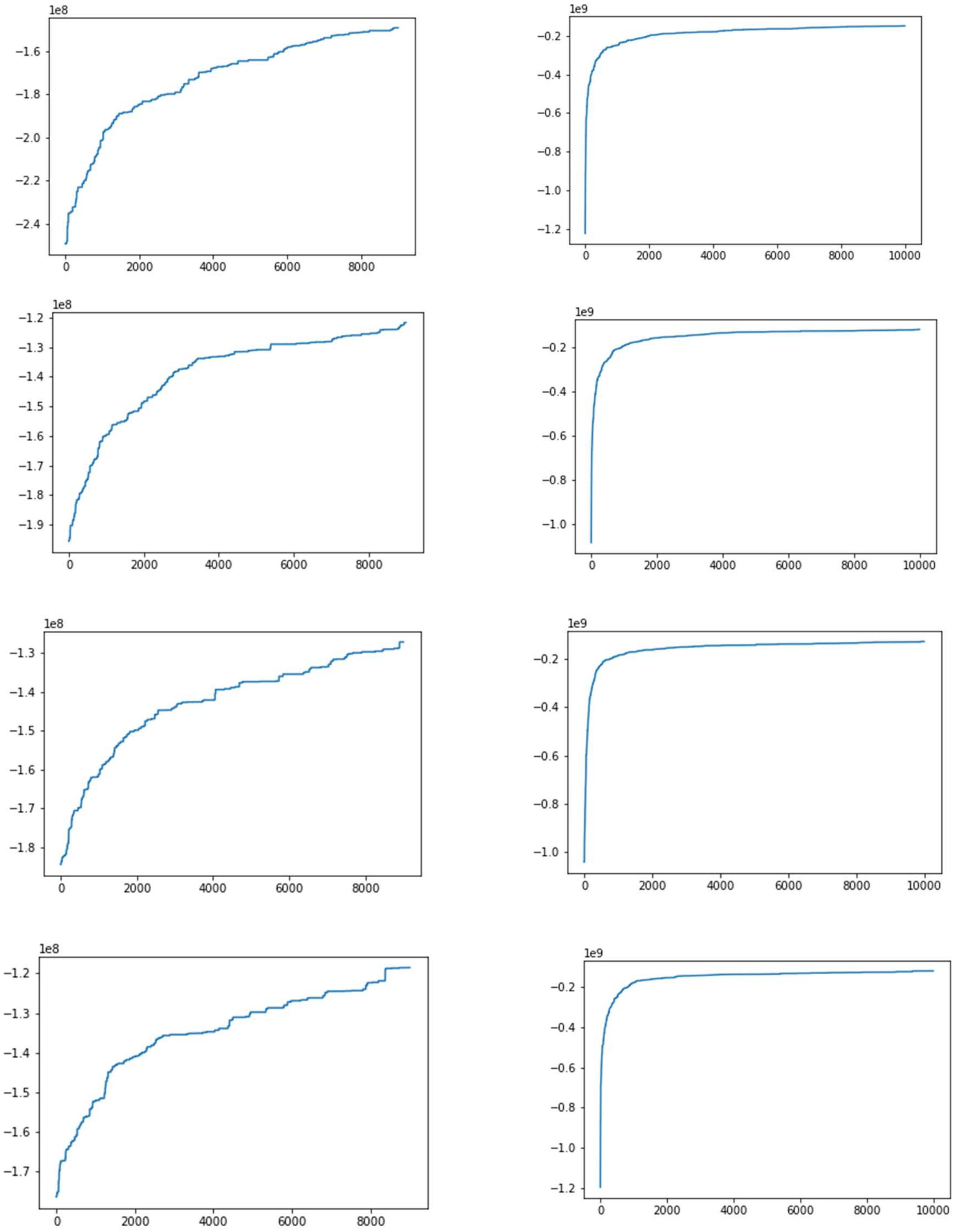


Figure 20: Fitness Values at each iteration taken for different number of sizes of tournament (left = 1000-10000, right=1-10000)

(Since there is a default case, I used same results for that case (sizes of tournament =5), (from upper to bottom, it is in descending order of the parameter)

1.4 Frac Elites

After first 1000th generation, results are:

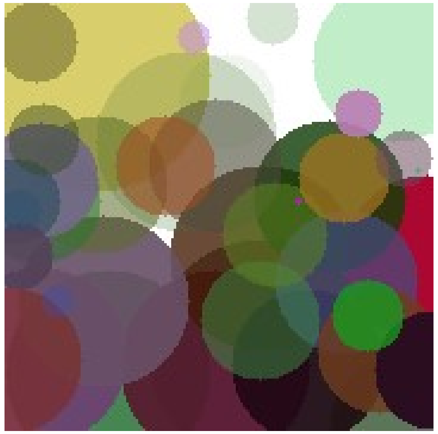


Figure 21: Result at 1000th generation (fraction of elites = 0.05)

Figure 22: Result at 1000th generation (fraction of elites = 0.2)

Figure 23: Result at 1000th generation (fraction of elites = 0.4)

In figures below, as you can go from left upper corner to right bottom corner, number of generations is increased by 1000.

0.05

0.2

0.4

Figure 24: Result at each 1000th generation with having different number of fractions of elites

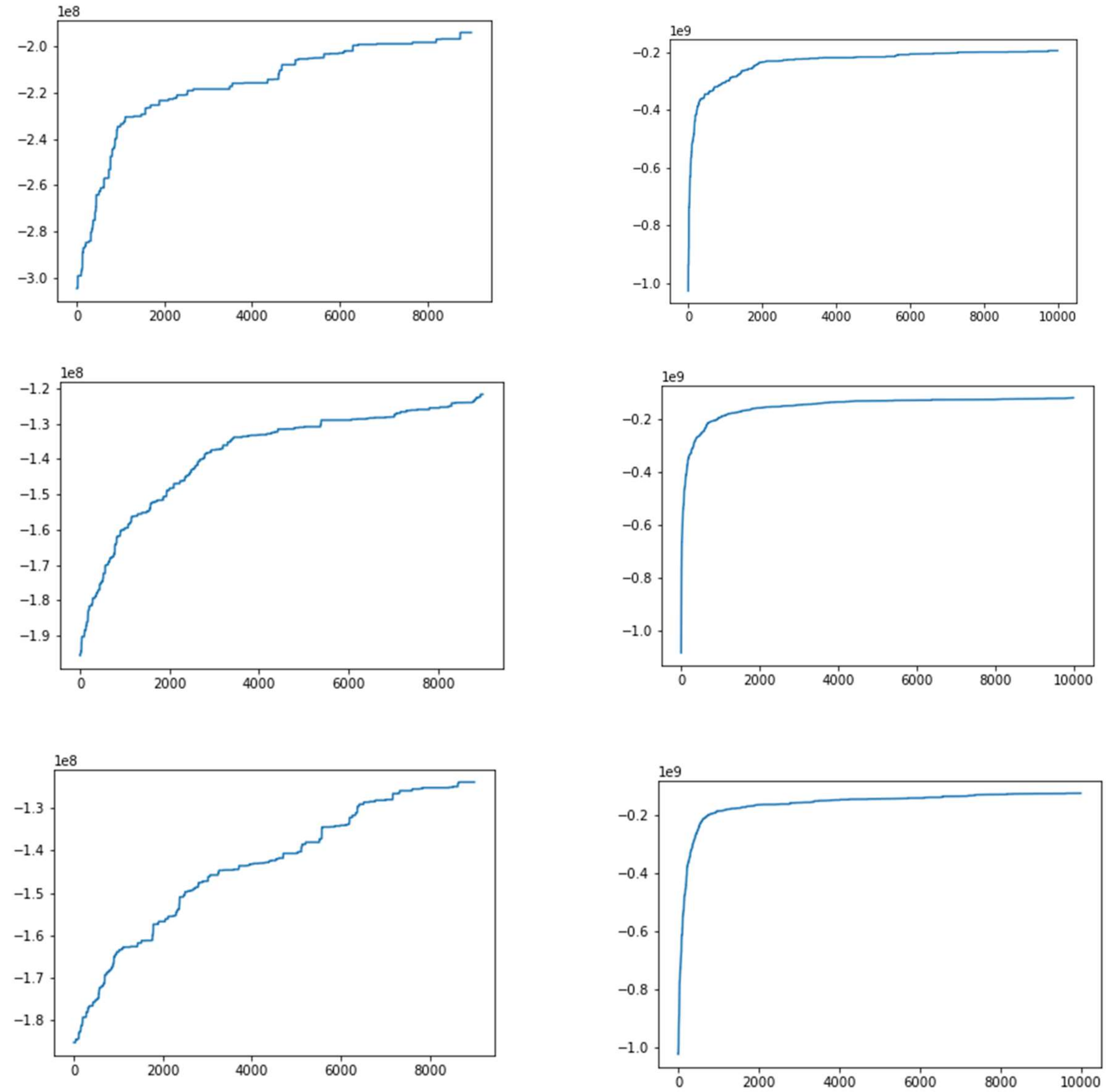


Figure 25: Fitness Values at each iteration taken for different number of fractions of elites (left = 1000-10000, right=1-10000)

(Since there is a default case, I used same results for that case (fractions of elites =0.2) (from upper to bottom, it is in descending order)

1.5 Frac Parents

After first 1000th generation, results are:

Figure 26: Result at 1000th generation (fraction of parents= 0.2)

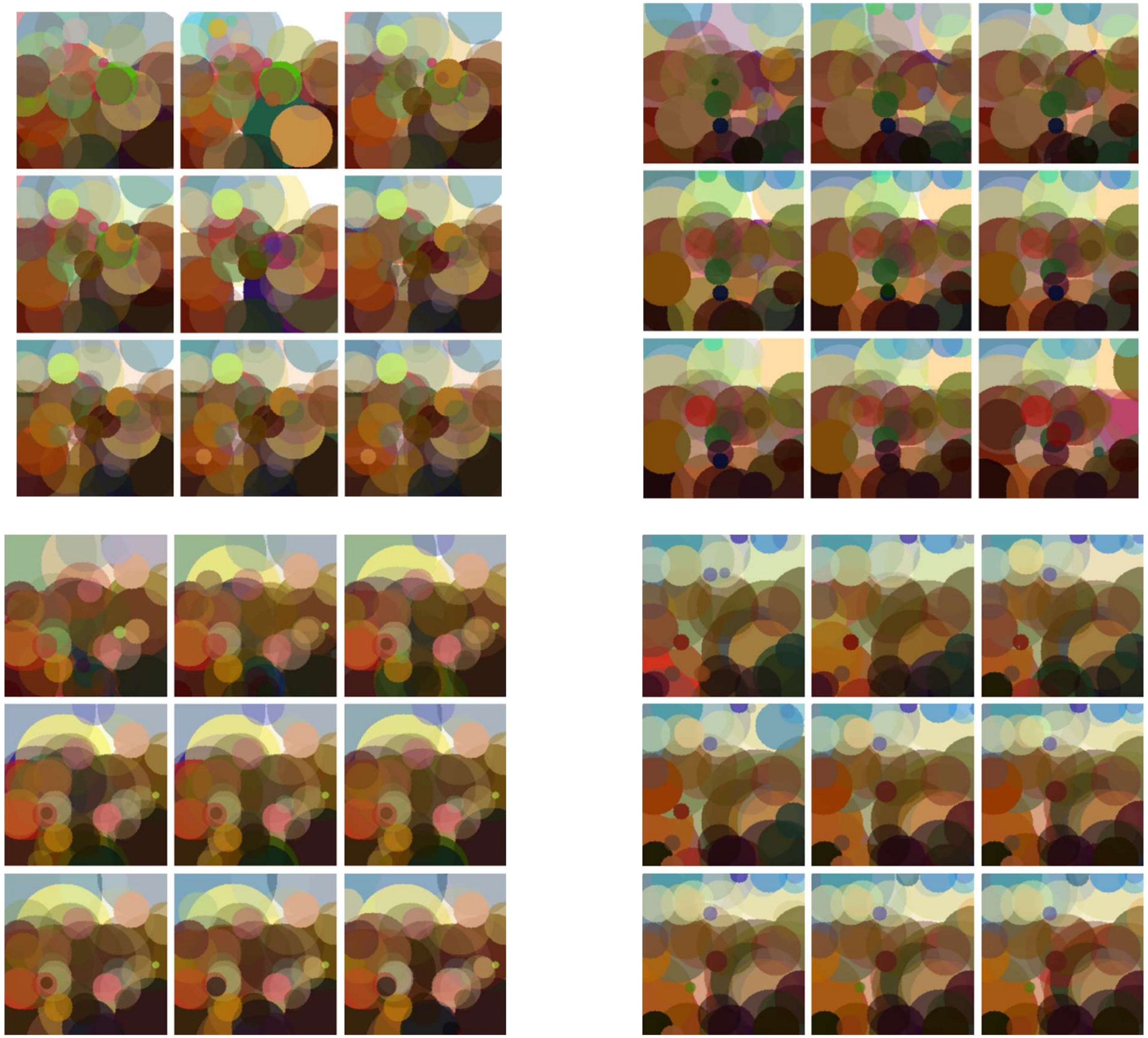
Figure 27: Result at 1000th generation (fraction of parents = 0.4)

Figure 28: Result at 1000th generation (fraction of parents =0.6)

Figure 29: Result at 1000th generation (fraction of parents = 0.8)

In figures below, as you can go from left upper corner to right bottom corner, number of generations is increased by 1000.

0.2 0.4



0.6 0.8

Figure 30: Result at each 1000th generation with having different number of fraction of parents

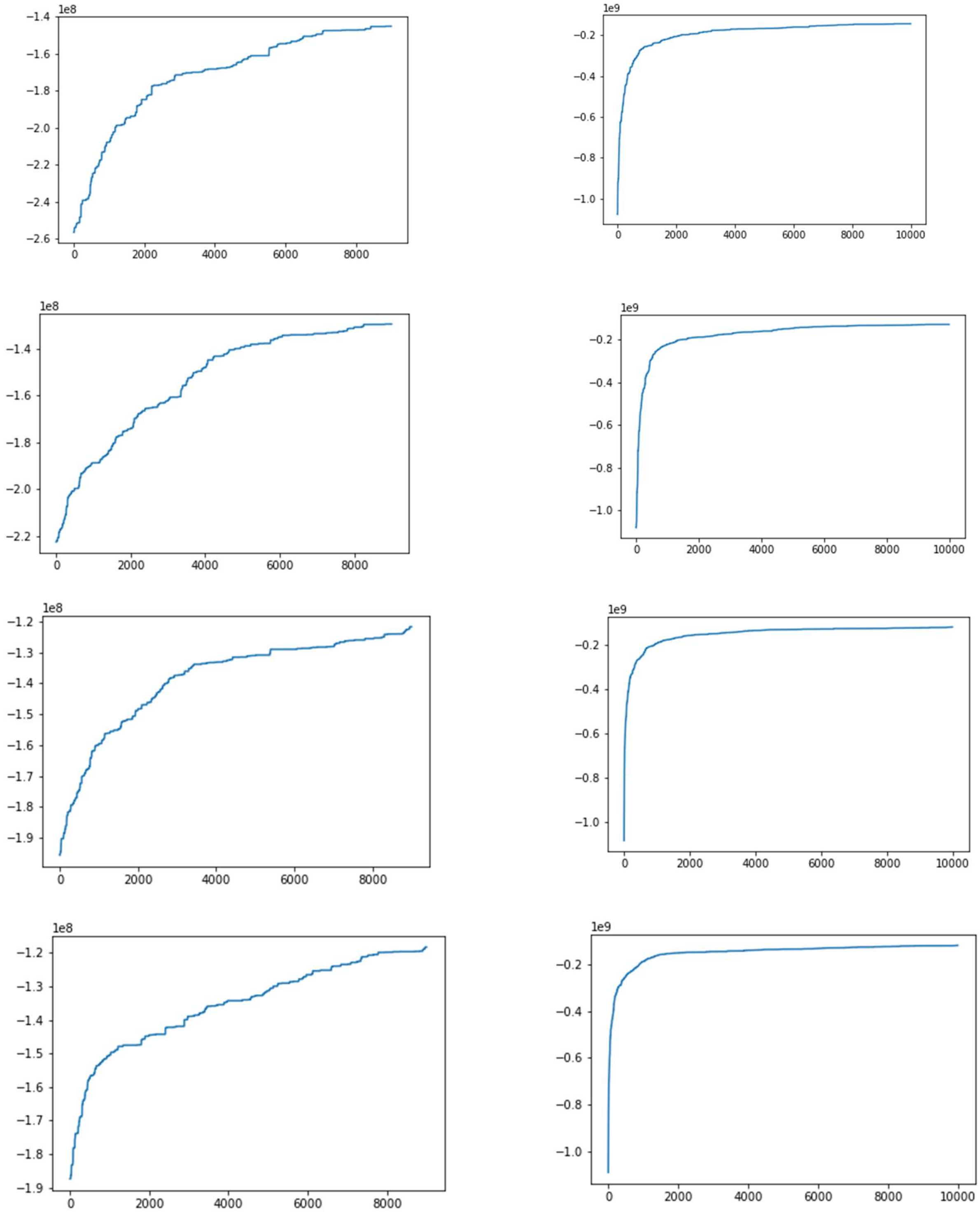


Figure 31: Fitness Values at each iteration taken for different number of fraction of parents (left = 1000-10000, right=1-10000)

(Since there is a default case, I used same results for that case (fraction of parents =0.6) (from upper to bottom, it is in descending order.)

1.6 Mutation Probability

After first 1000^{th} generation, results are:

Figure 32: Result at 1000^{th} generation (mutation probability= 0.1)

Figure 33: Result at 1000^{th} generation (mutation probability= 0.2)

Figure 34: Result at 1000^{th} generation (mutation probability =0.5)

Figure 35: Result at 1000^{th} generation (mutation probability = 0.8)

In figures below, as you can go from left upper corner to right bottom corner, number of generations is increased by 1000.

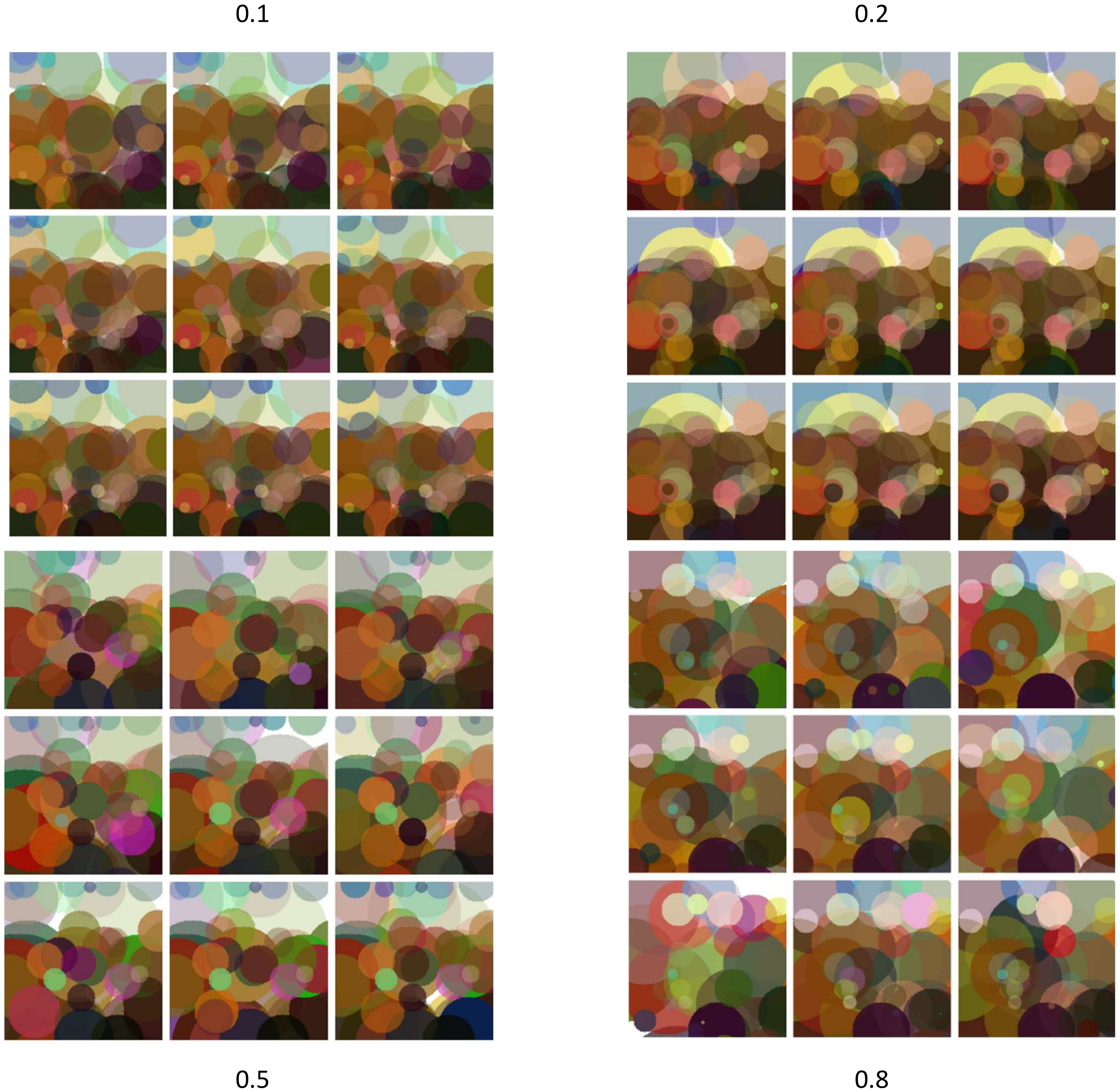


Figure 36: Result at each 1000th generation with having different number of mutation probabilities

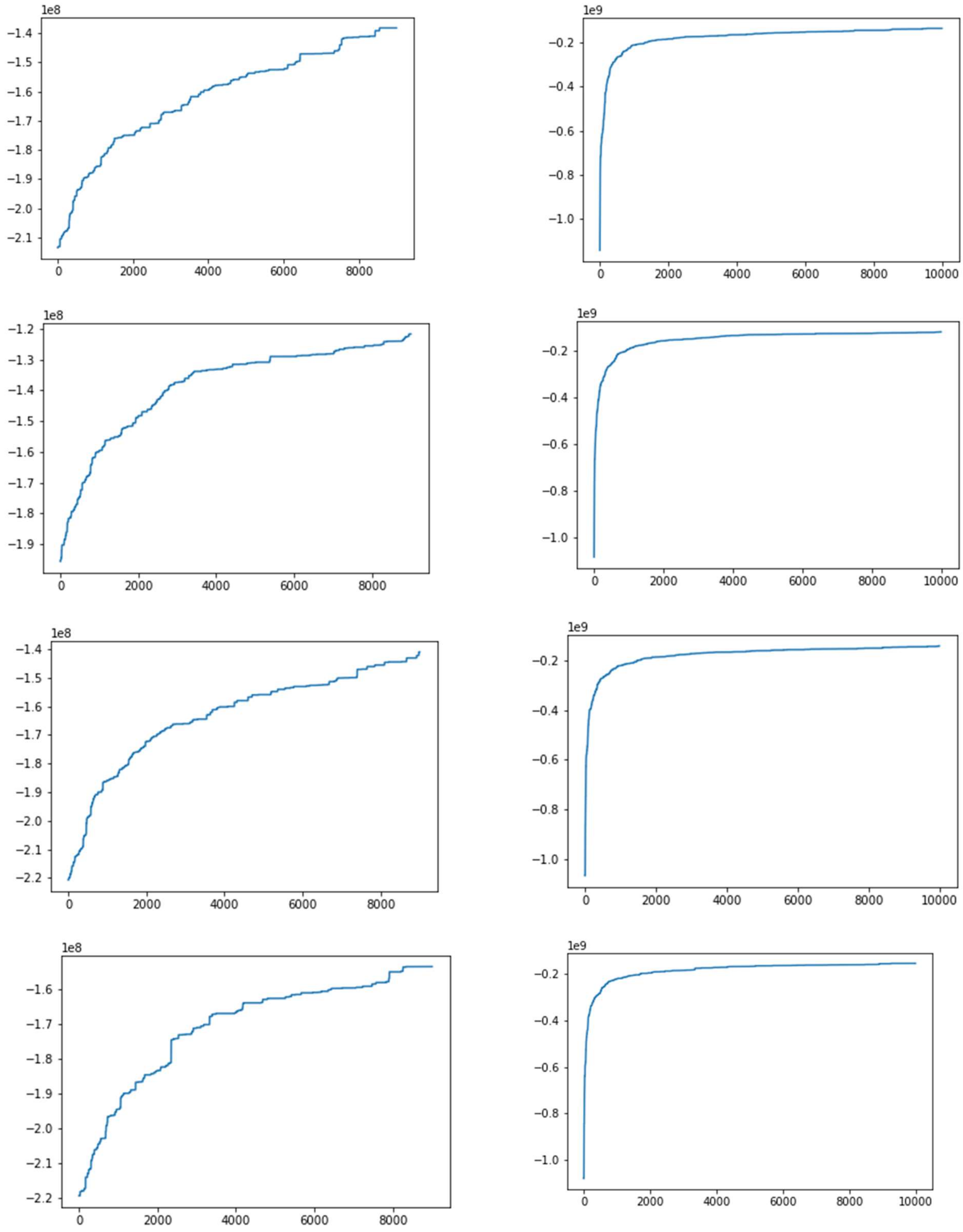


Figure 37: Fitness Values at each iteration taken for different number of mutation probabilities (left = 1000-10000, right=1-10000)

(Since there is a default case, I used same results for that case (mutation probabilities =0.2) (from upper to bottom, it is in descending order))

1.7 Mutation Guide

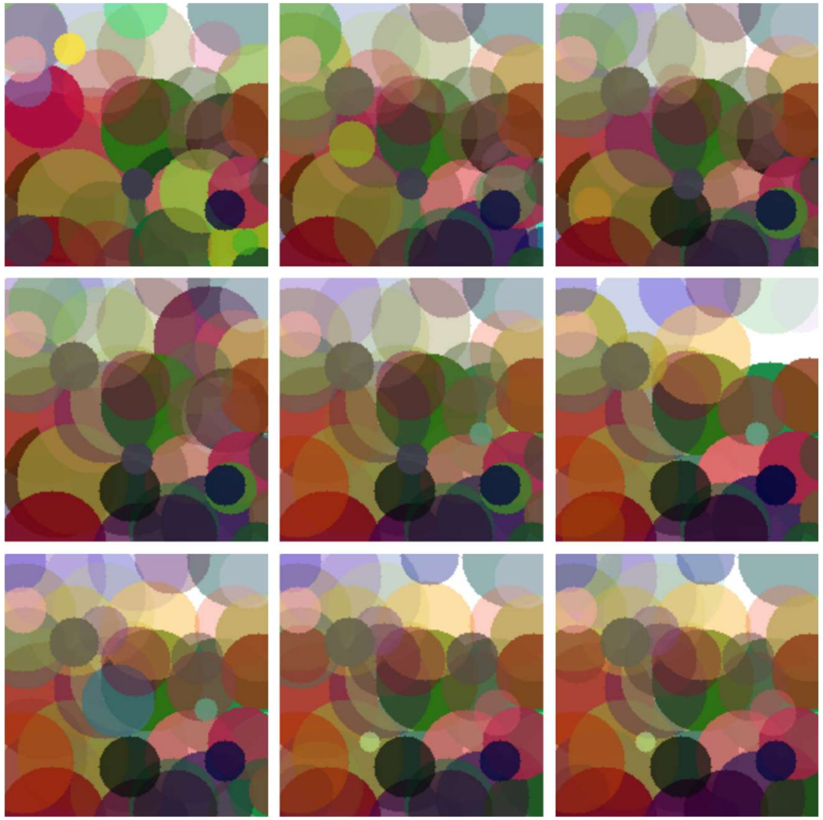
After first 1000th generation, results are:

Figure 38: Result at 1000th generation (mutation guide = “guided”)

Figure 39: Result at 1000th generation (mutation guide = “unguided”))

In figures below, as you can go from left upper corner to right bottom corner, number of generations is increased by 1000

“Guided”



“Unguided”

Figure 40: Result at each 1000th generation with having different types of mutation

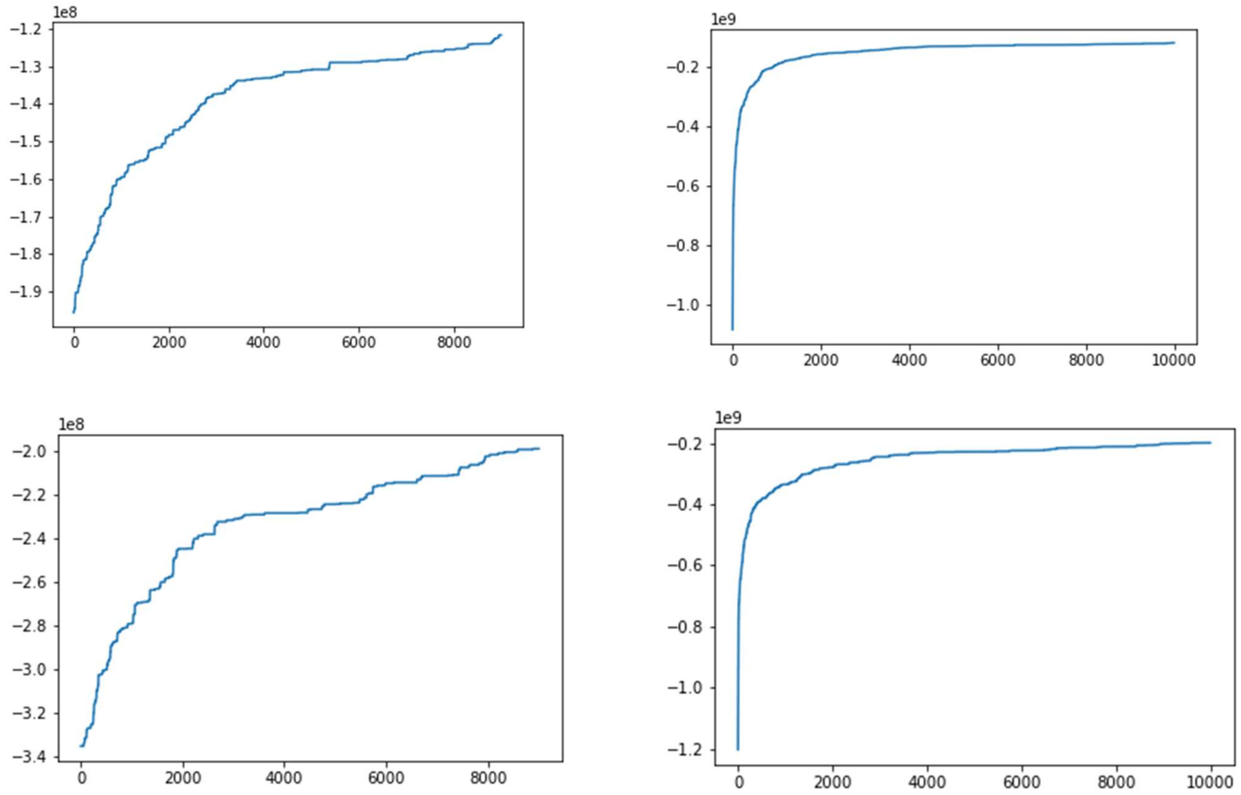


Figure 41: Fitness Values at each iteration taken for different types of mutation (left = 1000-10000, right=1-10000)

(Since there is a default case, I used same results for that case (mutation guide =”guided”), upper=guided, lower=unguided)

2. Discussions

My three suggestions can be listed as follows:

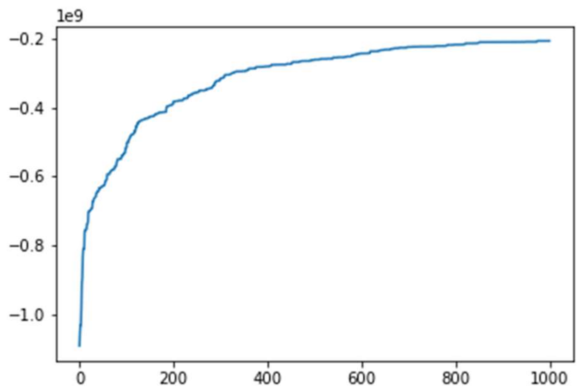
- Suggestion #1: Changeable mutation probabilities
- Suggestion #2: Changeable fraction of elites/parents
- Suggestion #3: Better Mutation Method

Before mentioning one by one, the first two ones have same point. They are all trying to do maximize power by adapting selection intensity. At the begging of training, we want to get diverse individuals. However, at the middle point of training, our aim is to decrease diversity. These diversity amount can be adjusted by mutation probabilities, fraction of elites/parents, or mutation type (which is not mentioned here).

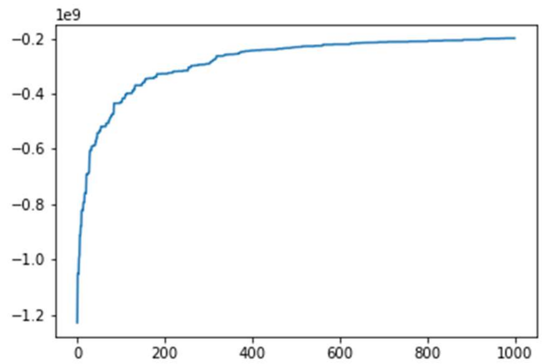
Also, since they are all independent from each other, we can apply them in one train to get perfect result of convergence time.

2.1. Changeable Mutation Probabilities

As we mentioned above, mutation probability will provide us a way to adjust selection intensity. To get better result, we are going to adjust mutation probability high as possible at the beginning so that we will have larger diversity. As the number of iteration increases, we are going to decrease this probability. This will result in smaller convergence time. The results are shown below:



Mut_prob = 0.2



Mut_prob= Changed as time passes

Figure 42: Fitness value vs iteration number with different two mutation probability method

As figure 42 states, at the iteration 600, adaptive version of mutation probability convergences; however, for constant mutation probability, it convergences at the iteration 900. Although they are both converging at the same fitness value. As you can guess, it is hard to compare from their plotting, since they converged at the iteration of 1000.

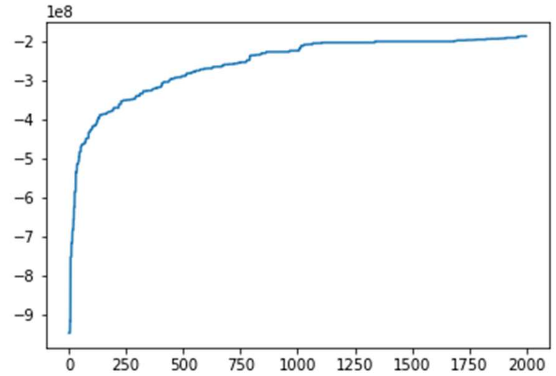
Mut_prob=0.2

Mut_prob= Changed as time passes

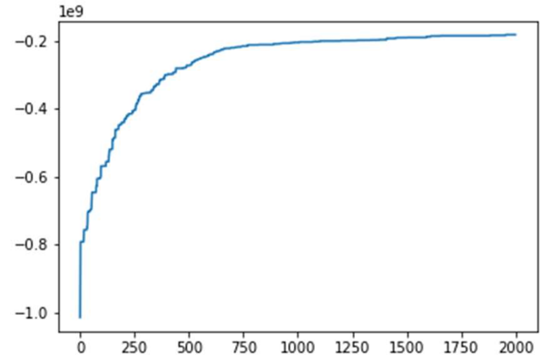
Figure 43: Results at iteration number of 1000 with different two mutation probability method

2.2. Changeable Fraction of Elites/Parents

As we mentioned above, fraction of elites(parents) will provide us a way to adjust selection intensity. To get better result, we are going to adjust fraction of elites(parents) low(high) as possible at the beginning so that we will have larger diversity. In other words, we are encouraging individuals to join tournament at the beginning rather than being a warrior in tournament. As the number of iteration increases, we are going to increase(decrease) this fraction. This will result in smaller convergence time. The results are shown below:



Fractions = Constant



Fractions = Changed as time passes

Figure 44: Fitness value vs iteration number with different two fractions method

Note: While training with my suggestion-2 converges at the iteration of 800, normal training results in convergence time of 1000th iteration although there are converges same point.

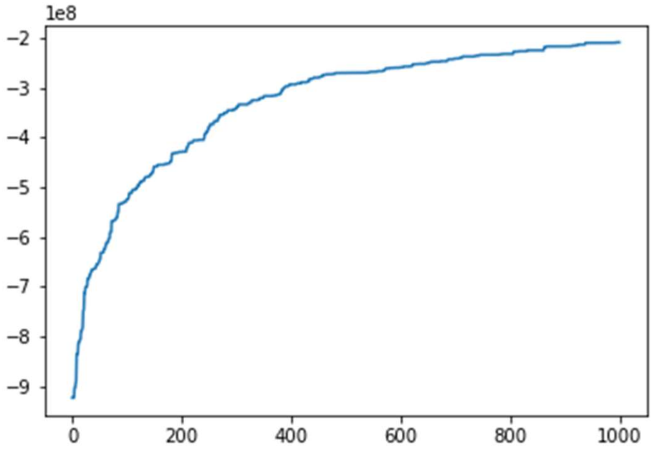
Fractions = Constant

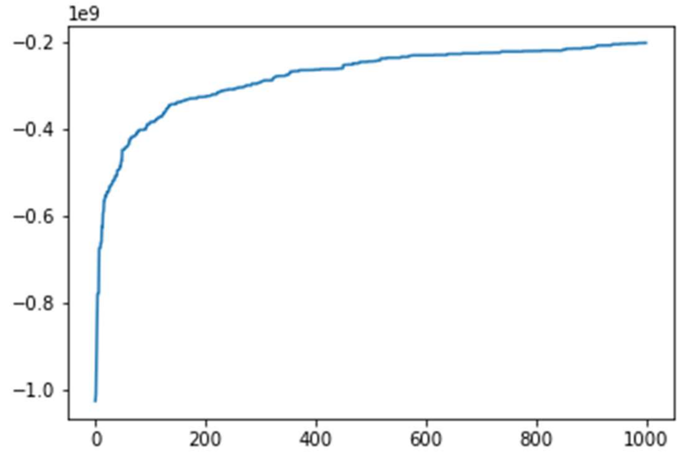
Fractions = Changed as time passes

Figure 45: Results at iteration number of 1000 with different two fractions method

2.3. Better Mutation Method

Mutation is done without considering any decrease in fitness value so we can say that do mutation until mutated one has better fitness value than previous (unmutated) one. However, this will create a slower mutation loop especially when it gets higher fitness value. Therefore, it can be done in a very small interval and at the beginning. As another application, this method can be applied with a very tiny probably (might be 0.0001).





Mutation = Normal

Mutation = At the beginning, Forced mutation

Figure 46: Fitness value vs iteration number with different two mutation method

Note: As figure 46 states, at the beginning there is a huge difference between fitness values.

Mutation = Normal

Mutation = At the beginning, Forced mutation

Figure 47: Fitness value vs iteration number with different two mutation method