METU EE 449
Computational Intelligence

Evolutionary Art

Ozgur Gulsuna

[a]*Middle East Technical University, Electrical and Electronics Engineering, Ankara, Turkey*

Introduction

This report explores evolutionary algorithms for image generation problem given source image using filled circles. Individuals represent the population, with each gene corresponding to circle properties. Evaluation compares images using a fitness function. The process involves selection, crossover, and mutation. Experiments vary hyperparameters, analyzing fitness plots. Two modifications are proposed for improved convergence speed and quality.

Keywords: NumPy; Evolutionary Algorithms; Image Generation; Vectorization

1. Experiments with Hyperparameters

The first part of the report is about the hyperparameters. These are the parameters that are set by the designer and the values effect the output quality. The hyperparameters are namely, population size, number of generations, mutation rate, crossover rate, and the number of circles. Here an experiment is conducted to see the effect of these parameters on the output quality. The default parameters are set as follows:

- Pop. Size: **20**
- Number of Genes: **50**
- Tournament Size: **5**
- Fraction of Elites: **0.2**
- Fraction of Parents: **0.6**
- Mutation Prob.: **0.2**
- Mutation Type: **guided**
- Generation Size: **10000**

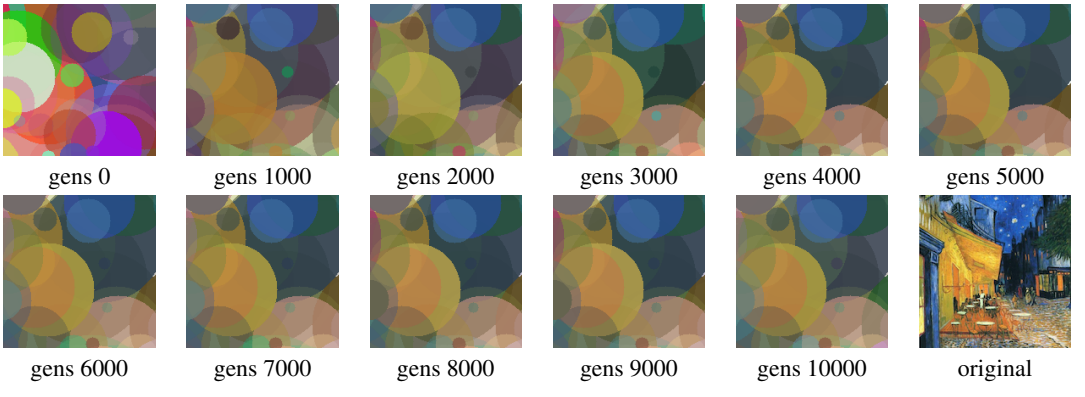
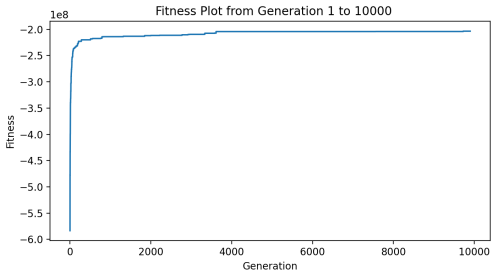


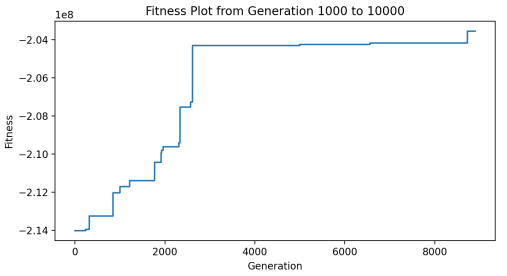
Fig. 1: Best Images for Default Parameters

E-mail address: ozgur.gulsuna@metu.edu.tr

© 2023 Creative Commons Attribution 4.0 International License.



gens 1 to 10000



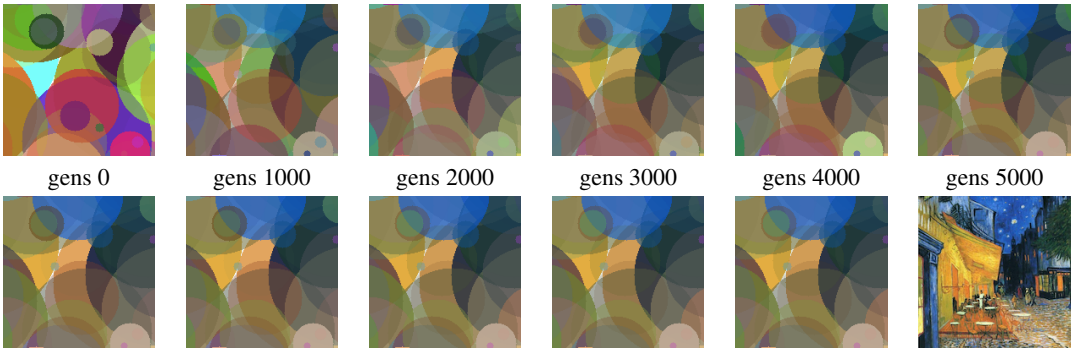
gens 1000 to 10000

Fig. 2: Fitness Plots for Default Parameters

1.1. Number of Individuals < *num_inds* >

The number of individuals is the size of the population. The individuals are used to generate next generation. Higher the number of individuals, the more diverse the starting population is.

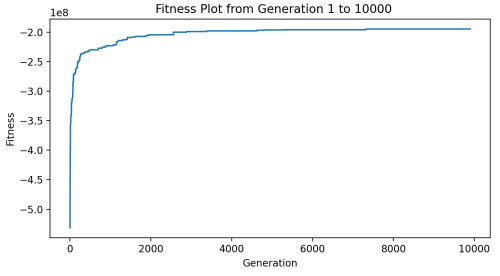
1.1.1. Number of Individuals = 5



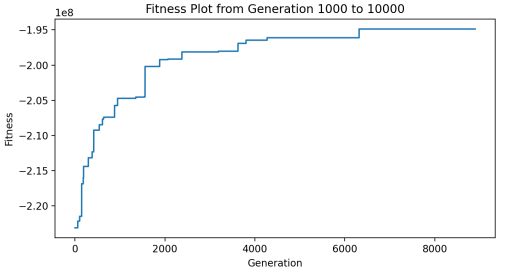
gens 0 | gens 1000 | gens 2000 | gens 3000 | gens 4000 | gens 5000

gens 6000 | gens 7000 | gens 8000 | gens 9000 | gens 10000 | original

Fig. 3: Best Images for < *num_inds* > = 5



gens 1 to 10000



gens 1000 to 10000

Fig. 4: Fitness Plots for < *num_inds* > = 5

1.1.2. Number of Individuals = 10

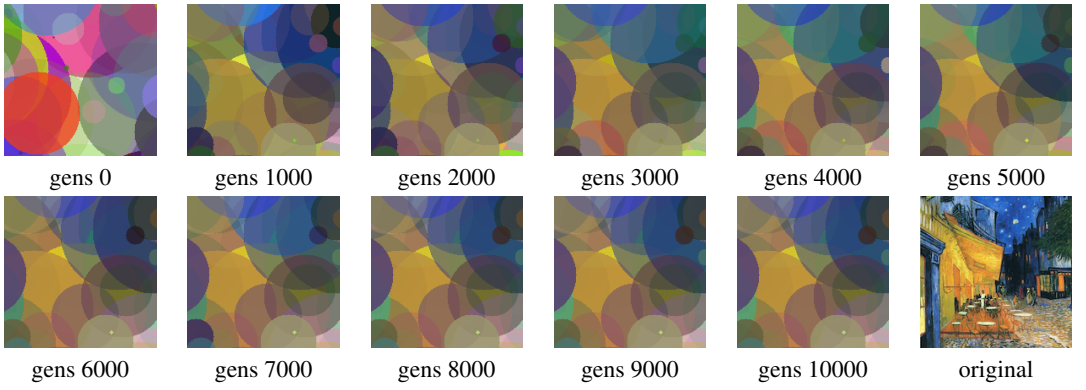
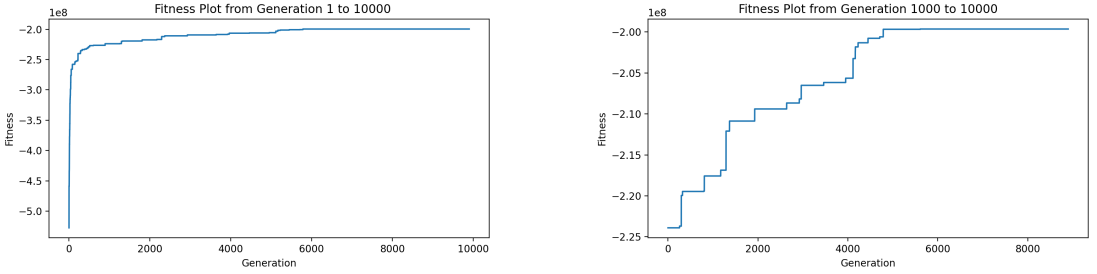


Fig. 5: Best Images for < *num_inds* > = 10



gens 1 to 10000 gens 1000 to 10000

Fig. 6: Fitness Plots for < *num_inds* > = 10

1.1.3. Number of Individuals = 40

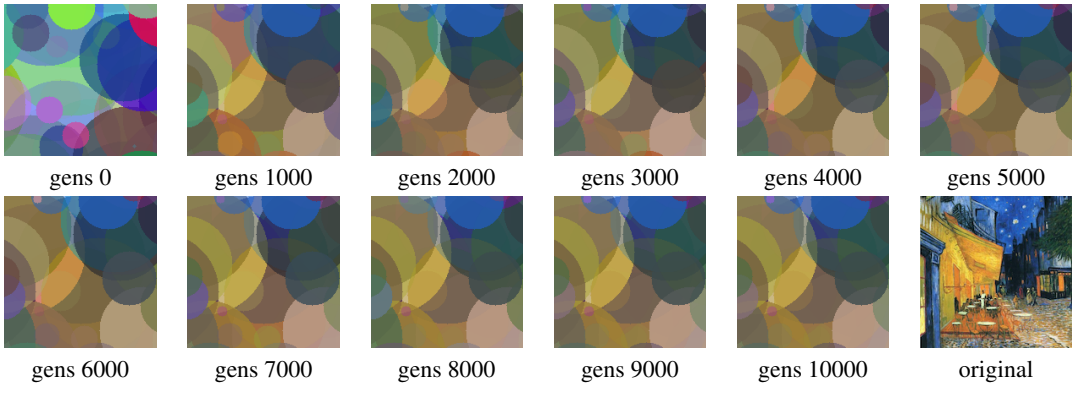
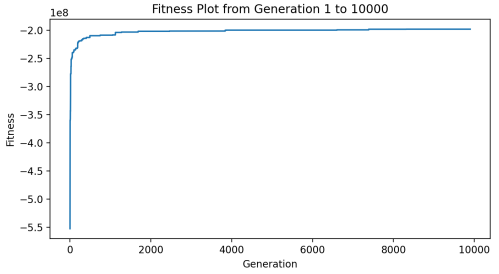
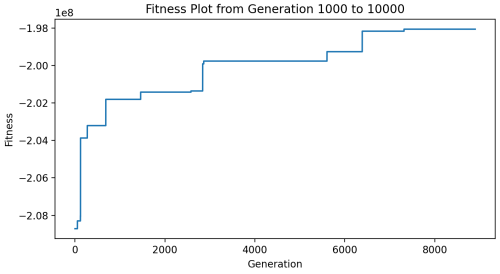


Fig. 7: Best Images for < *num_inds* > = 40



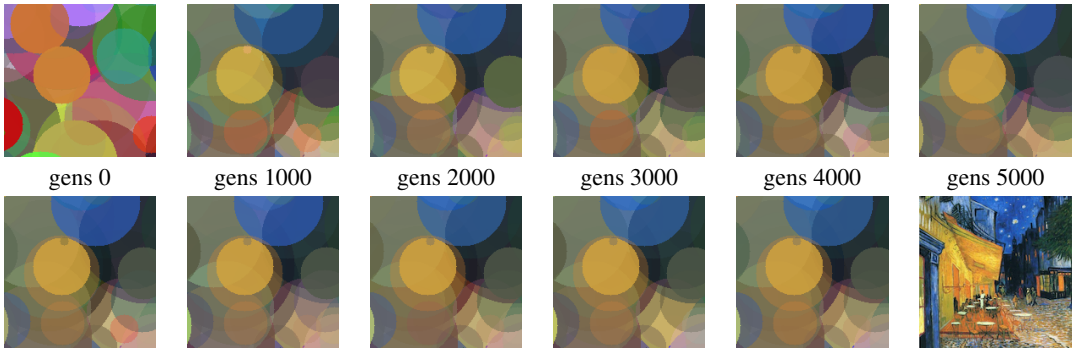
gens 1 to 10000



gens 1000 to 10000

Fig. 8: Fitness Plots for < *num_inds* > = 40

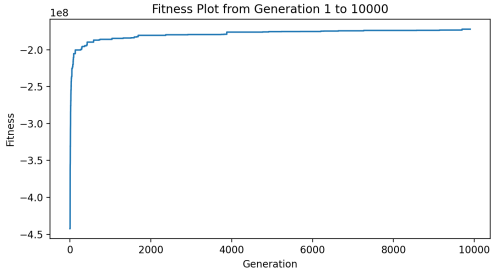
1.1.4. Number of Individuals = 60



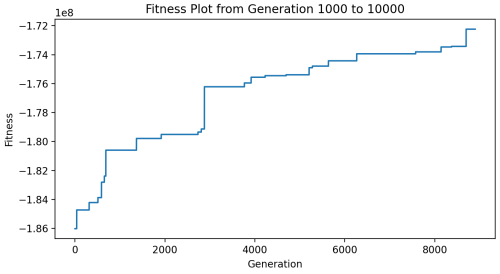
gens 0 | gens 1000 | gens 2000 | gens 3000 | gens 4000 | gens 5000

gens 6000 | gens 7000 | gens 8000 | gens 9000 | gens 10000 | original

Fig. 9: Best Images for < *num_inds* > = 60



gens 1 to 10000



gens 1000 to 10000

Fig. 10: Fitness Plots for < *num_inds* > = 60

A larger population size has better chance in exploring the solution space and finding the global optimum. However, it also means that the algorithm will take more time to converge. The smaller population size has better advantage in exploitation, that is searching around the promising regions of the solution space.

A larger population size helps a faster convergence, that means that the algorithm is able to find the optimal solution in less number of generations, eventhougn the algorithm takes more processing power.

Solution quality is also affected by the population size, we can see from the examples that the fitness plots of the larger population size is more smooth and predictable.

1.2. Number of Genes < *num_genes* >

Gene size is the number of circles that are used to generate the image. The more circles there are, the more detailed the image is. The higher computational cost and harder the problem becomes when the number of genes increases.

1.2.1. Number of Genes = 15

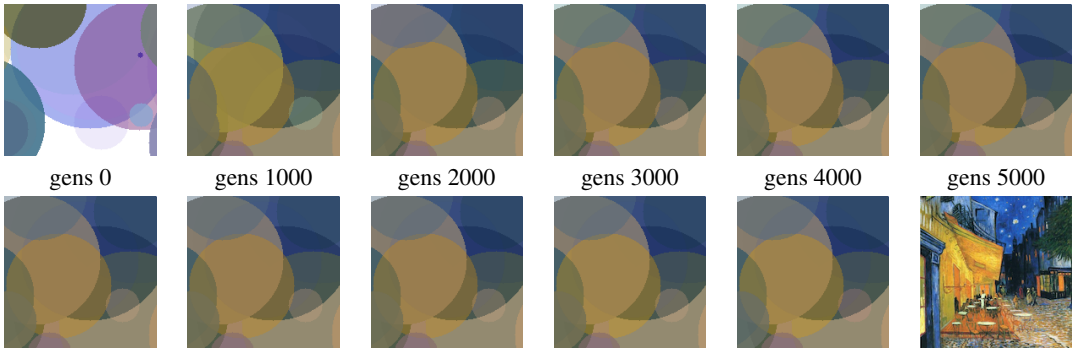
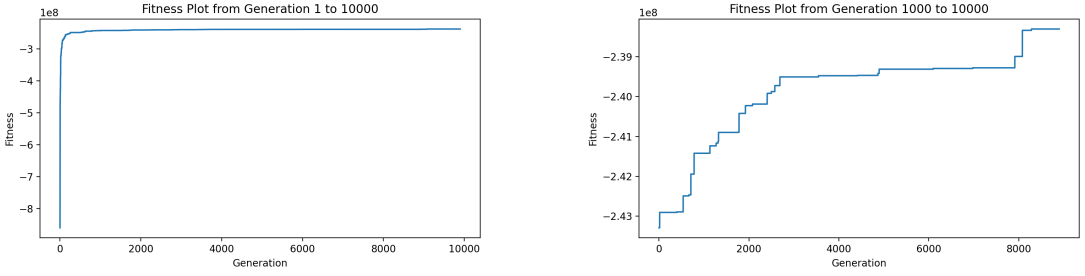


Fig. 11: Best Images for < *num_genes* > = 15



gens 1 to 10000 gens 1000 to 10000

Fig. 12: Fitness Plots for < *num_genes* > = 15

1.2.2. Number of Genes = 30

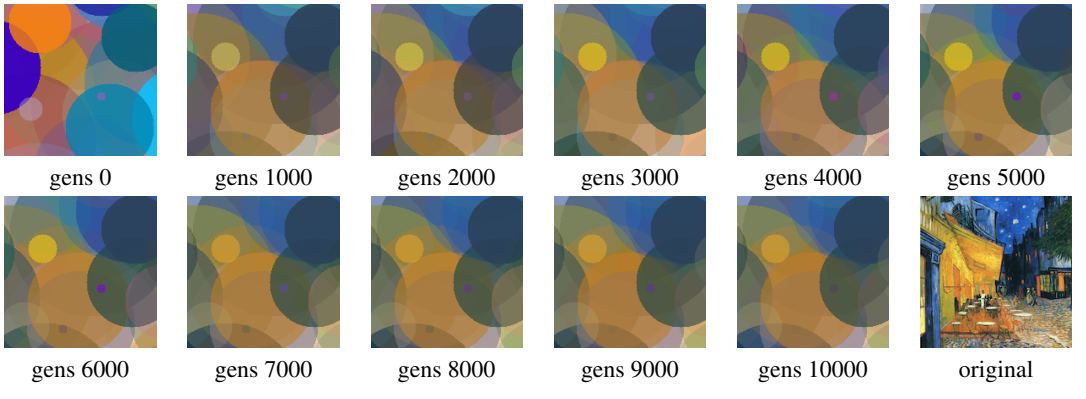
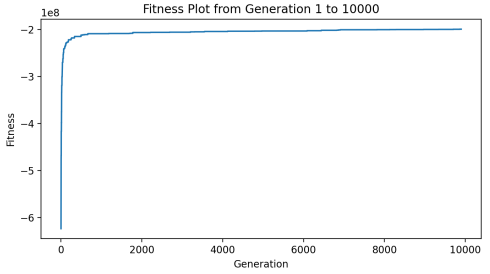
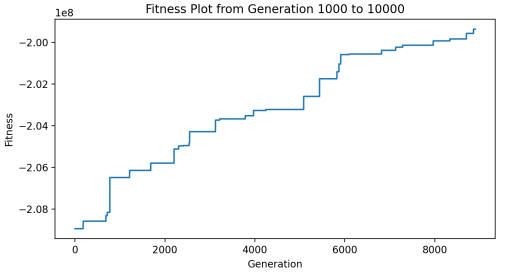


Fig. 13: Best Images for < *num_genes* > = 30



gens 1 to 10000



gens 1000 to 10000

Fig. 14: Fitness Plots for < *num_genes* > = 30

1.2.3. Number of Genes = 80

gens 0

gens 1000

gens 2000

gens 3000

gens 4000

gens 5000

gens 6000

gens 7000

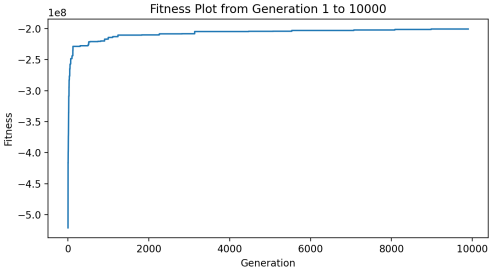
gens 8000

gens 9000

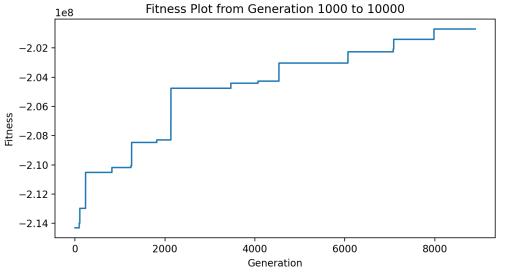
gens 10000

original

Fig. 15: Best Images for < *num_genes* > = 80



gens 1 to 10000



gens 1000 to 10000

Fig. 16: Fitness Plots for < *num_genes* > = 80

1.2.4. Number of Genes = 120

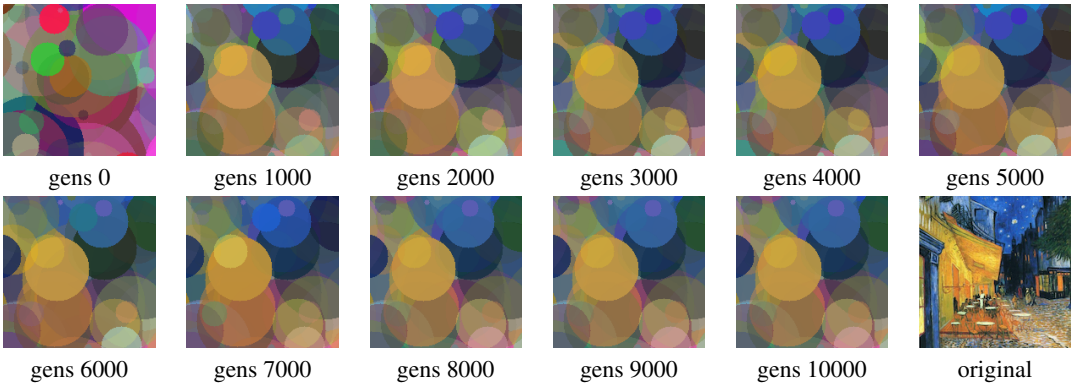
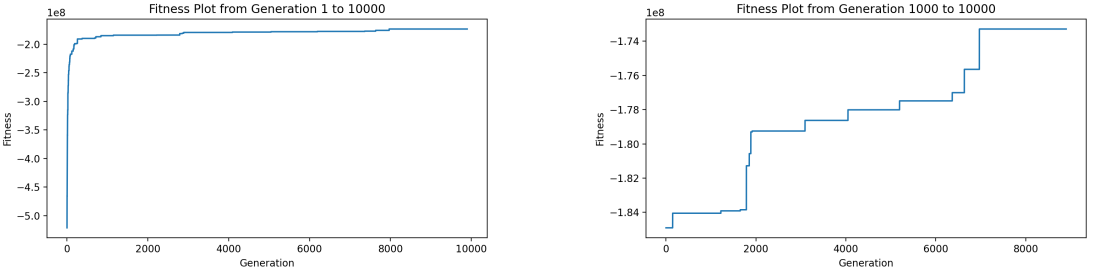


Fig. 17: Best Images for < *num_genes* > = 120



gens 1 to 10000 gens 1000 to 10000

Fig. 18: Fitness Plots for < *num_genes* > = 120

For very low number of genes, the algorithm is not able to generate the image properly. As the number of genes increases, with increased computational cost, the algorithm is able to generate the image with better fitness. The best fitness is achieved with 120 genes.

1.3. Tournament Size < *tm_size* >

Tournament size is the number of individuals that are selected for the tournament. The higher the tournament size, the higher the probability of selecting the best individual for the tournament.

1.3.1. Tournament Size = 2

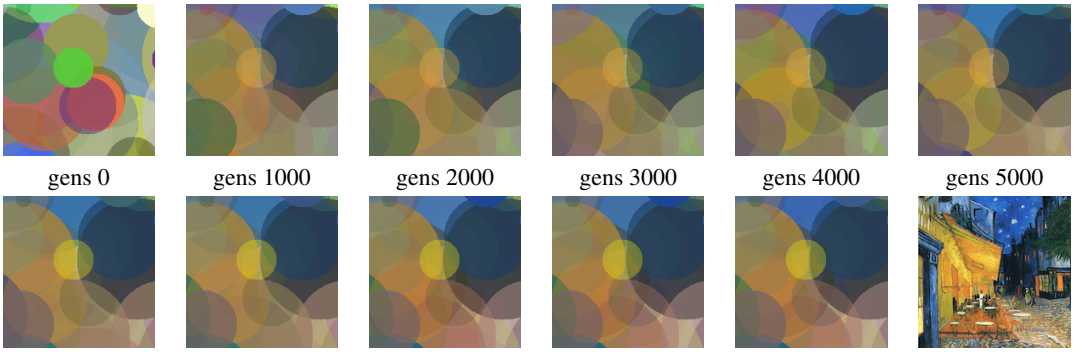
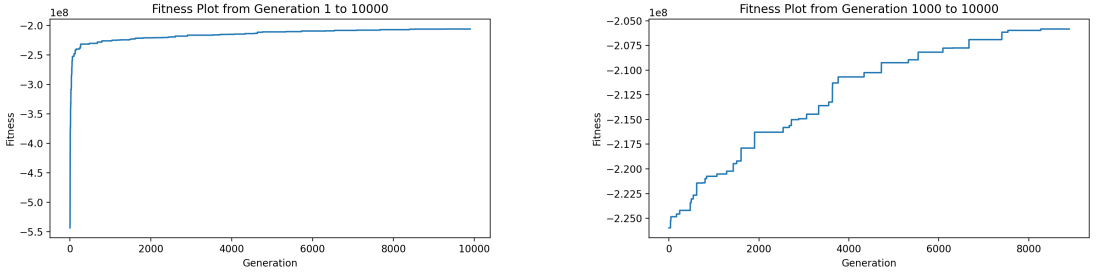


Fig. 19: Best Images for < *tm_size* > = 2



gens 1 to 10000 gens 1000 to 10000

Fig. 20: Fitness Plots for < *tm_size* > = 2

1.3.2. Tournament Size = 8

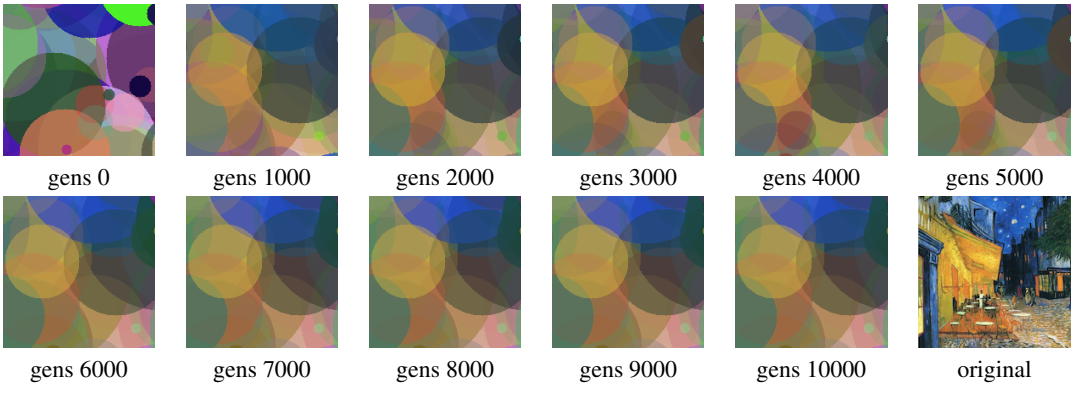
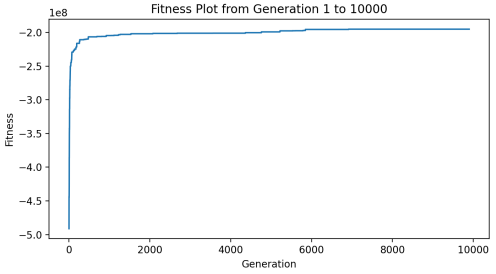
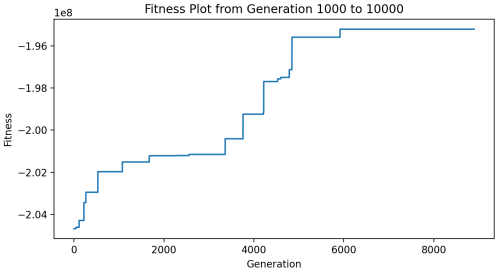


Fig. 21: Best Images for < *tm_size* > = 8



gens 1 to 10000



gens 1000 to 10000

Fig. 22: Fitness Plots for < *tm_size* > = 8

1.3.3. Tournament Size = 16

gens 0

gens 1000

gens 2000

gens 3000

gens 4000

gens 5000

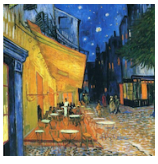
gens 6000

gens 7000

gens 8000

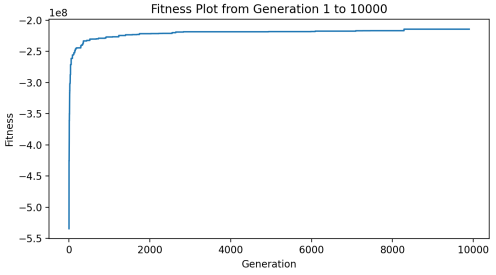
gens 9000

gens 10000

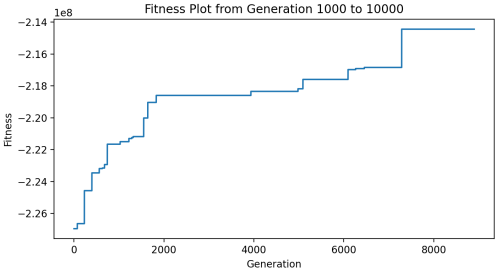


original

Fig. 23: Best Images for < *tm_size* > = 16



gens 1 to 10000



gens 1000 to 10000

Fig. 24: Fitness Plots for < *tm_size* > = 16

The results seem similar, and the results are highly dependent on the random seed.

What can be deducted is the tournament size 2 is favored by the exploitation and finds a good enough solution around a minimum. The tournament size 16 is favored by the exploration and finds a better solution around the global minimum but it takes a lot more effort. The inbetween tournament size 8, which performs very similar to size 5, is a good compromise between the two.

1.4. Fraction of Elites $< frac\_elites >$

Elites are very effective in keeping the population fit, since they are directly copied to the next generation. The fraction of elites is the percentage of the population that is copied to the next generation. The elites are also included in the reproduction process, so they can be mutated and crossed over. This approach gives more chance to the elites to be replaced by better individuals.

1.4.1. Fraction of Elites = 0.04

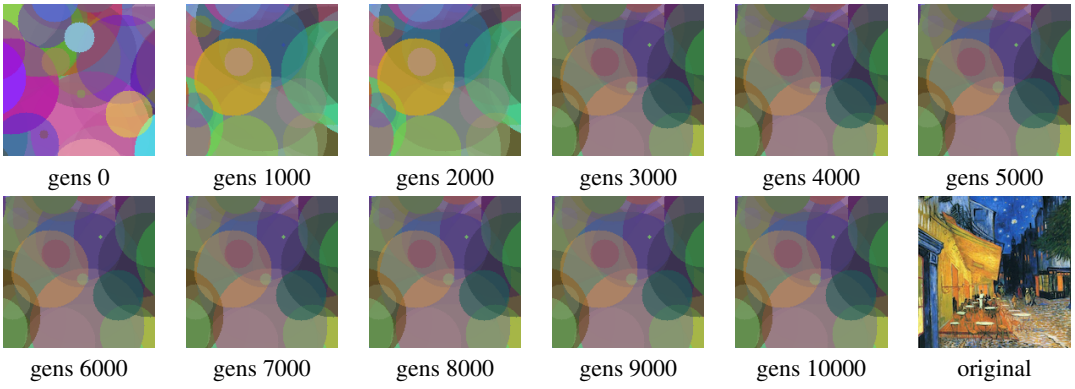
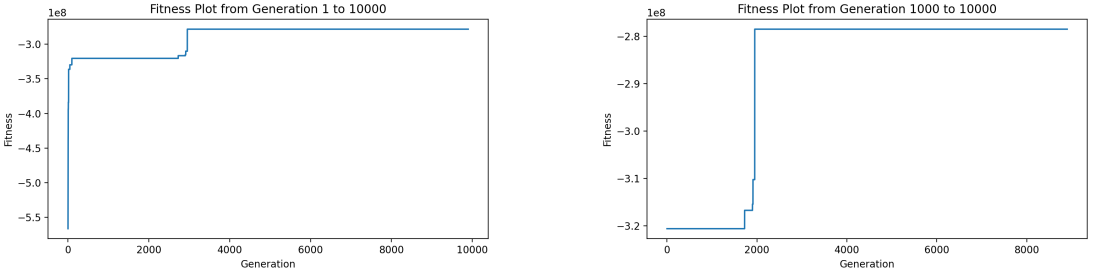


Fig. 25: Best Images for $< frac\_elites >$ = 0.04



gens 1 to 10000 gens 1000 to 10000

Fig. 26: Fitness Plots for $< frac\_elites >$ = 0.04

1.4.2. Fraction of Elites = 0.35

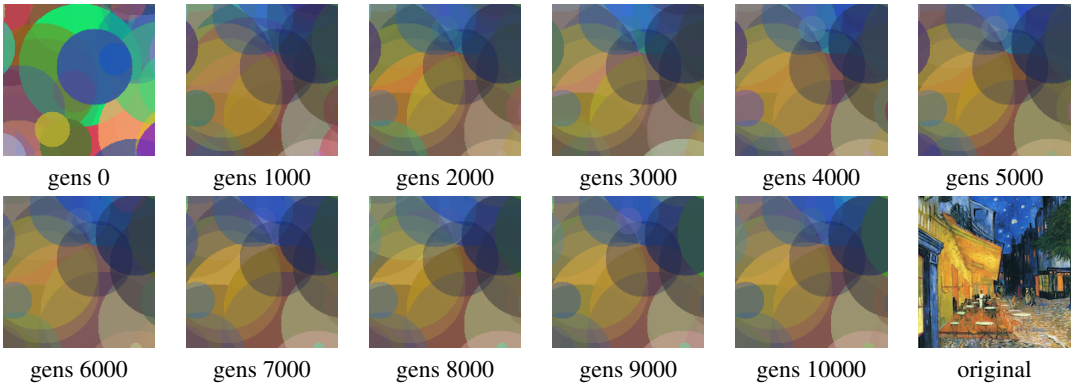
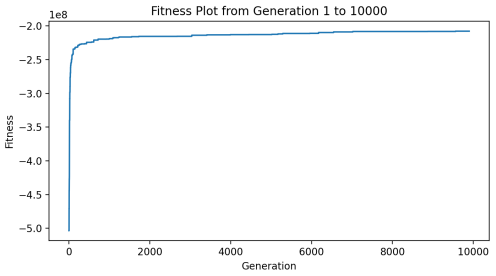
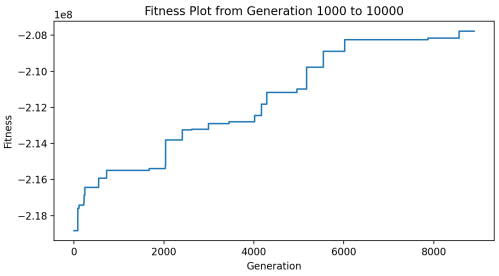


Fig. 27: Best Images for $< frac\_elites >$ = 0.35



gens 1 to 10000



gens 1000 to 10000

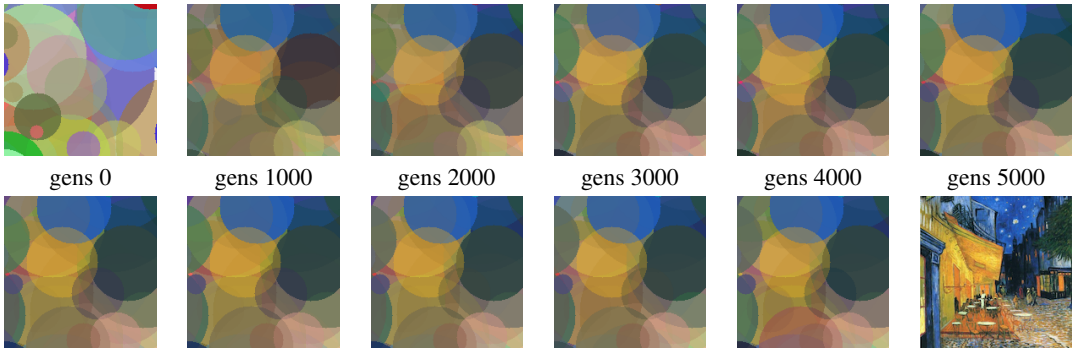
Fig. 28: Fitness Plots for $< frac\_elites > = 0.35$

The most important aspect of the elites are the preservation of the best individuals. The first case is where elite number is zero. It can be seen that the random chance factor is high and the fitness plot is blocky. The second case is the default case, where the elites are 20% of the population. It is better than the first case, but the fitness plot is still discrete. The third case is where the every 1/3rd individual is an elite. The fitness plot is smooth and the best individual is better than the previous cases.

1.5. Fraction of Parents $< frac\_parents >$

The number of parents is the number of individuals that are selected for reproduction. The genetic reproduction is done by crossing over the genes of the parents. The elites are also included in the reproduction process.

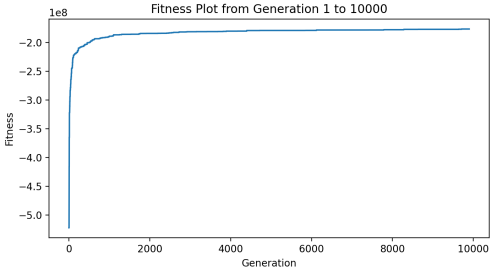
1.5.1. Fraction of Parents = 0.15



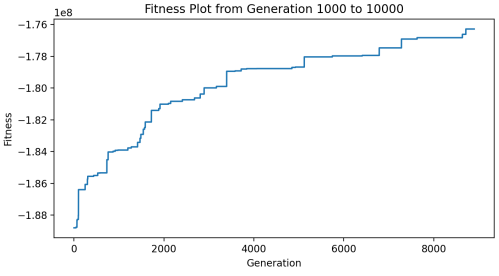
gens 0 | gens 1000 | gens 2000 | gens 3000 | gens 4000 | gens 5000

gens 6000 | gens 7000 | gens 8000 | gens 9000 | gens 10000 | original

Fig. 29: Best Images for $< frac\_parents > = 0.15$



gens 1 to 10000



gens 1000 to 10000

Fig. 30: Fitness Plots for $< frac\_parents > = 0.15$

1.5.2. Fraction of Parents = 0.3

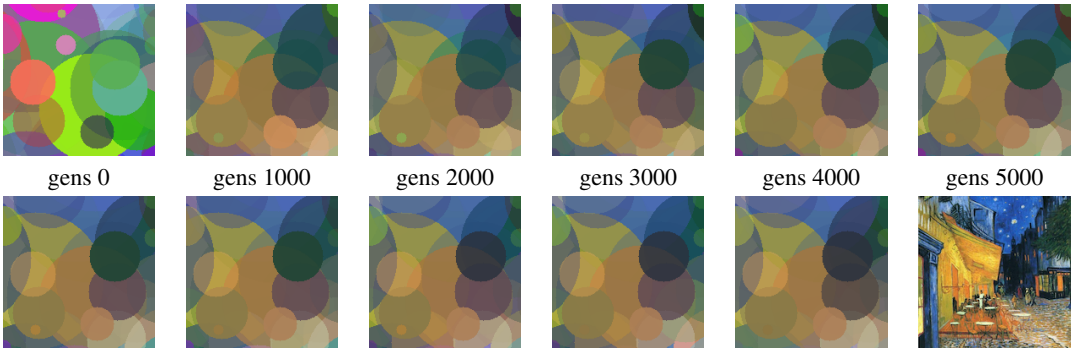
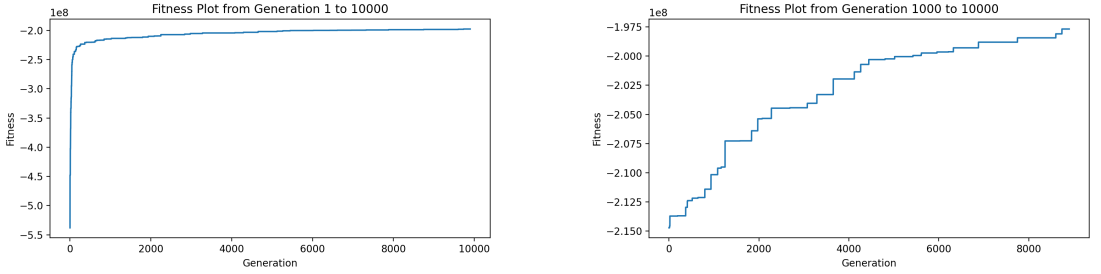


Fig. 31: Best Images for $< frac\_parents > = 0.3$



gens 1 to 10000 gens 1000 to 10000

Fig. 32: Fitness Plots for $< frac\_parents > = 0.3$

1.5.3. Fraction of Parents = 0.75

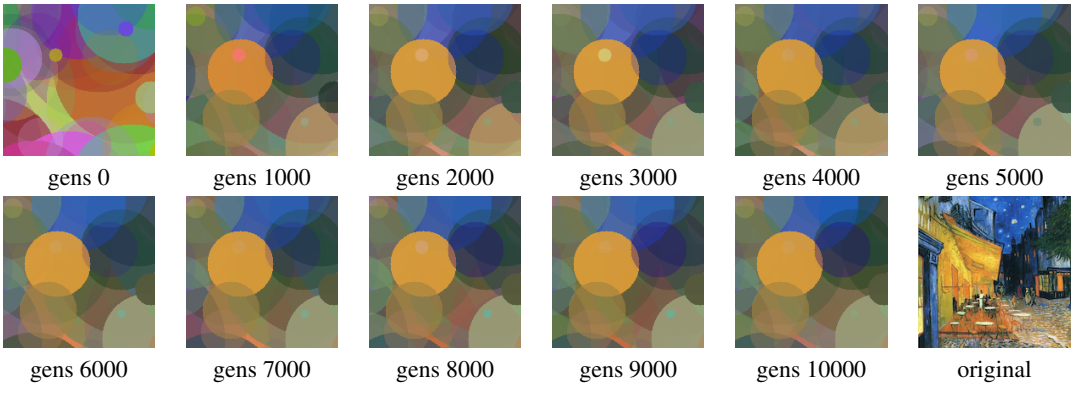
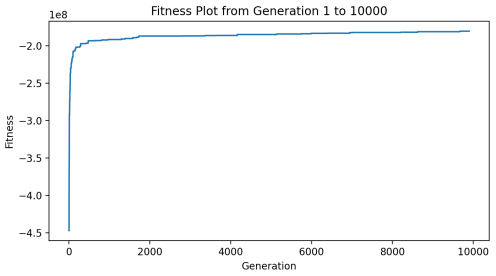
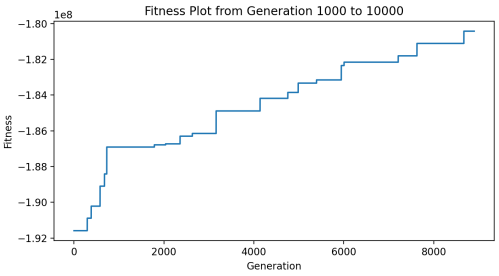


Fig. 33: Best Images for $< frac\_parents > = 0.75$



gens 1 to 10000



gens 1000 to 10000

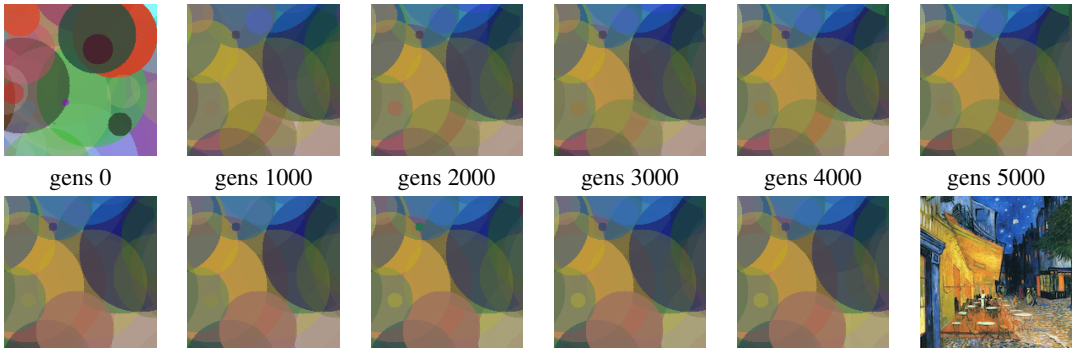
Fig. 34: Fitness Plots for < $frac\_parents$ > = 0.75

The first observations are that the fitness plot is smoother for middle range of parents. Smaller number of parents are not able to explore the search space well and the process depends on the random mutation. Higher number of parents are able to explore more but exploit less. This means that the process is slower. The middle range of parents are able to explore and exploit well. The best is the case where 30% of the population is selected as parents.

1.6. Mutation Probability < *mutation_prob* >

The mutation probability is the probability that a gene is mutated. The mutation is done by changing the value of the gene to a random value. It is required for two reasons. The first is to introduce new genes to the population and overcome the population converging to a thoroughbred form. The second is to have a chance to generate a better individual from a worse one.

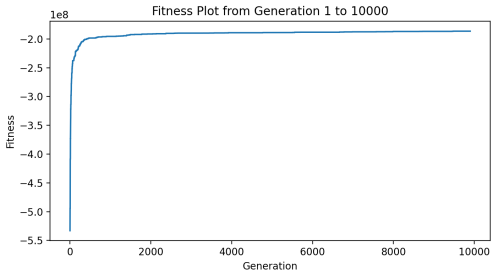
1.6.1. Mutation Probability = 0.1



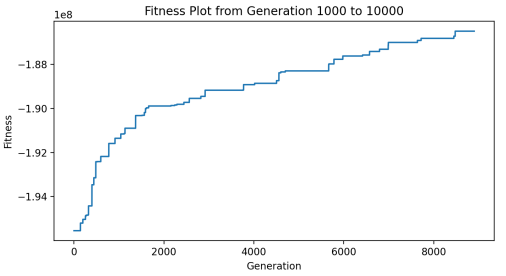
gens 0 | gens 1000 | gens 2000 | gens 3000 | gens 4000 | gens 5000

gens 6000 | gens 7000 | gens 8000 | gens 9000 | gens 10000 | original

Fig. 35: Best Images for < *mutation_prob* > = 0.1



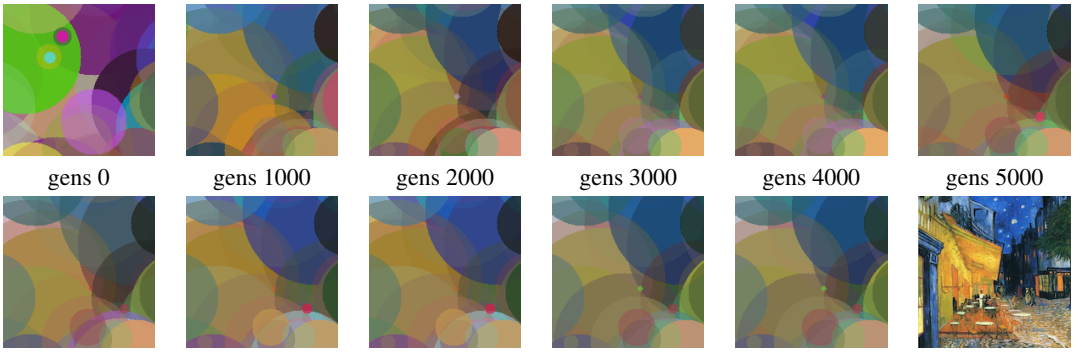
gens 1 to 10000



gens 1000 to 10000

Fig. 36: Fitness Plots for < *mutation_prob* > = 0.1

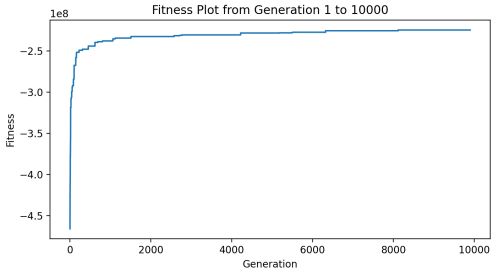
1.6.2. Mutation Probability = 0.4



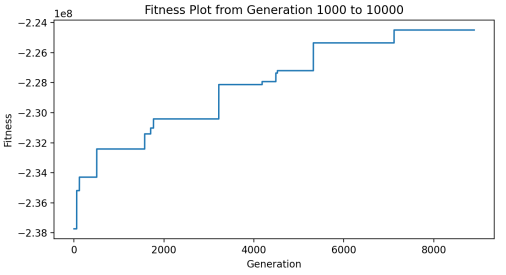
gens 0 | gens 1000 | gens 2000 | gens 3000 | gens 4000 | gens 5000

gens 6000 | gens 7000 | gens 8000 | gens 9000 | gens 10000 | original

Fig. 37: Best Images for < *mutation_prob* > = 0.4



gens 1 to 10000



gens 1000 to 10000

Fig. 38: Fitness Plots for < *mutation_prob* > = 0.4

1.6.3. Mutation Probability = 0.75

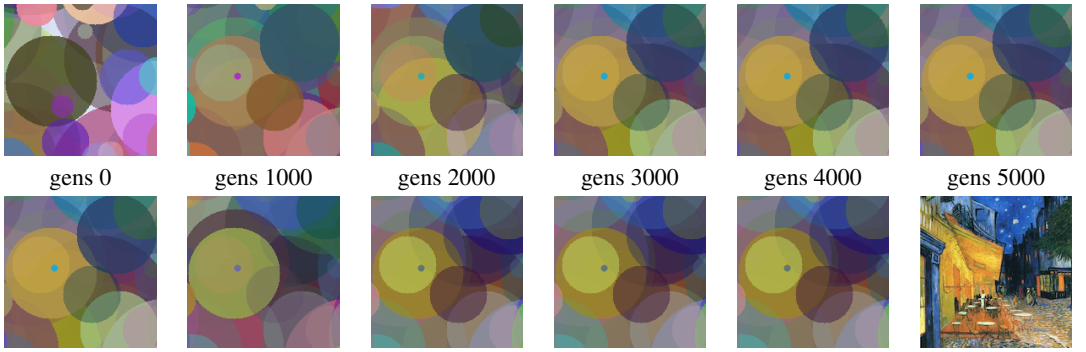
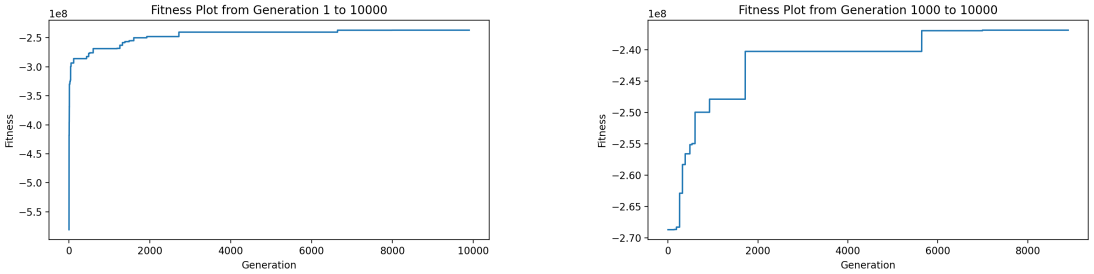


Fig. 39: Best Images for < *mutation_prob* > = 0.75



gens 1 to 10000 gens 1000 to 10000

Fig. 40: Fitness Plots for < *mutation_prob* > = 0.75

The mutation probability is the probability that a gene is mutated. The higher mutation rates change the populations more genes hence the good genes individuals are lost. The lower mutation rates change the population less and the process is slower. Again middle to smaller range of mutation rates are better. The best is experimentally found to be 0.2.

1.7. Mutation Type < *mutation_type* >

There is two types of mutation. The first is the guided mutation. The second is the unguided mutation. The guided mutation is the mutation where the mutation is done by changing the gene slightly. The unguided mutation is the mutation where the mutation is done by changing the gene to a complete random value.

1.7.1. Mutation Type = unguided

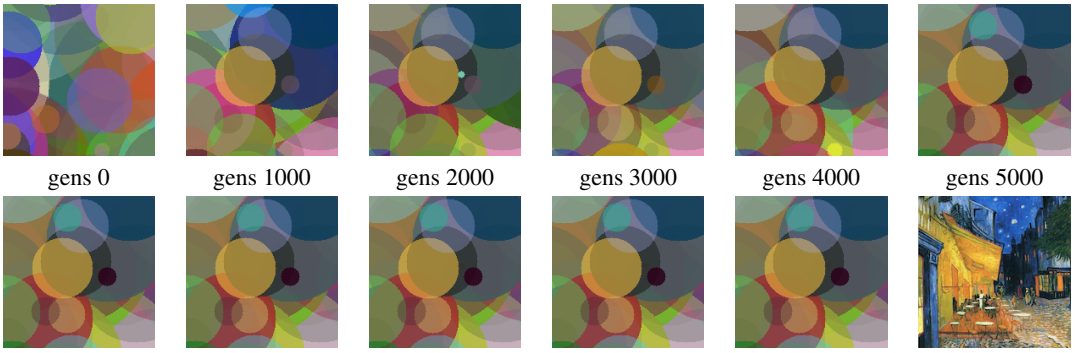
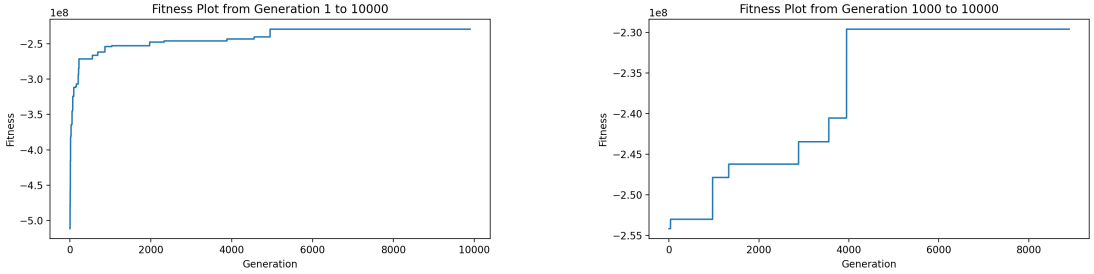


Fig. 41: Best Images for < *mutation_type* > = unguided



gens 1 to 10000 gens 1000 to 10000

Fig. 42: Fitness Plots for < *mutation_type* > = unguided

The results clearly show that the guided mutation is better. The reason is thought to be move the genes into more correct locations or deviate around the correct colors. The complete random mutation is not able to do this since a somewhat necessary gene is completely changed to a random value.

2. Discussions

The results show that the genetic algorithm is able to generate images that are similar to the original image. However, the results can be improved by doing modifications to the algorithm/process.

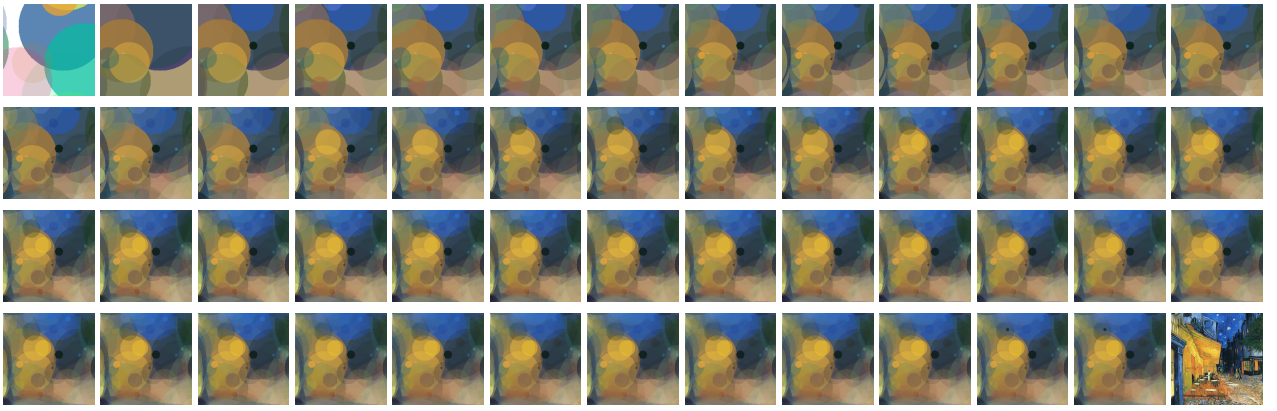
2.1. Multiple Passes

The observations of the fitness plots show that the fitness is not increasing after a certain number of generations. The algorithm deviates around the near correct colors but the improvement is very slow. The proposed method is to do multiple passes. The first pass is to generate the image with the best fitness with smaller number of generations (fast). Then the next passes take the previous best image as input and overlay a complete new population on top of it. Then the second population try to evolve to correct the previous image. The overlayed number of genes are collected over the whole process and the resultant image seem like it has higher number of genes. The number of genes for each pass is reduced to increase the speed of the process. Other parameters are kept the same.

The parameters for this process is given below.

- Pop. Size: **20**
- **Number of Genes: 10**
- Tournament Size: **5**
- Fraction of Elites: **0.2**
- Fraction of Parents: **0.6**
- Mutation Prob.: **0.2**
- Mutation Type: **guided**
- Generation Size: **500**
- **Number of Passes: 50**

The results are given below.



original

original

Fig. 43: All the passes of the multipass process

The detail level is more than the previous results with fitness value reaching -128303874. The process is around the median value in the evaluation time with 2302.0123467445374 seconds total. A better solution with same amount of processing power at 50 passes.

2.2. Monochromatic Images

The monochromatic images are faster to produce since the solution space is more limited. The results are given below.

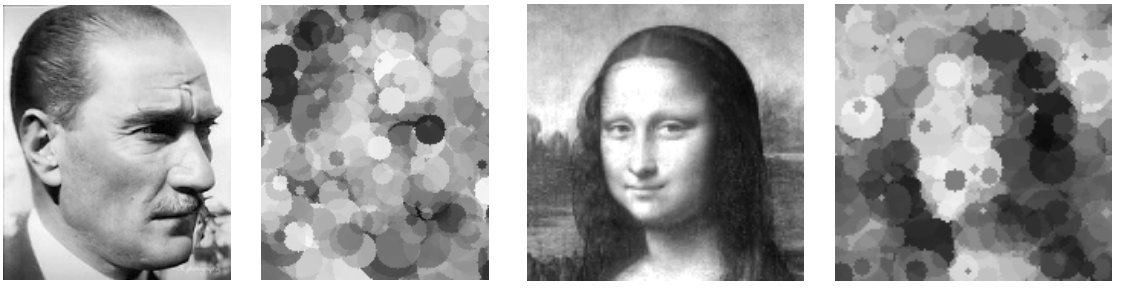


Fig. 44: All the passes of the multipass process