Evolutionary algorithms are effective optimization methods that draw inspiration from genetic and natural selection. They are frequently employed to tackle challenging issues by emulating evolution. In this report, we investigate the use of evolutionary algorithms to generate an image that resembles the well-known Van Gogh painting "Terrasse du Café le Soir" in *Figure 1*.

FIGURE 1. VAN GOGH'S "TERRASSE DU CAFÉ LE SOIR".

Our goal is to create a filled circle image that closely resembles a specified RGB source image. We use genetic representation within an evolutionary algorithm to do this. The algorithm works with a population of people, each of whom has a set of genes that make up their own chromosome.

In this context, a gene represents a colored circle and encodes attributes such as its position, size, color, and transparency. By manipulating these gene attributes, we can generate a variety of circle combinations that collectively form an image resembling the desired painting.

In the context of the evolutionary algorithm, a person is a candidate solution represented by a chromosome, which is made up of a group of genes. Individuals in the population evolve across generations to explore and take advantage of the solution space through mechanisms like selection, crossover, and mutation.

The parameters we investigate in this experiment include population size, number of genes per individual, mutation probability, fraction of elites and parents, and tournament size. By examining the effects of these parameters over 10,000 generations, we gain insights into their impact on the image generation process and the algorithm's behavior.

This study enables us to observe the evolution of the image and analyze the convergence patterns. By evaluating the algorithm's performance and considering the various parameter settings, we can provide valuable recommendations for optimizing the algorithm for image generation tasks.

In summary, this report explores the application of evolutionary algorithms to image generation, employing the concepts of genes, individuals, and populations. We investigate the effects of different parameters and aim to create an image reminiscent of Van Gogh's "Terrasse du Café le Soir" painting.

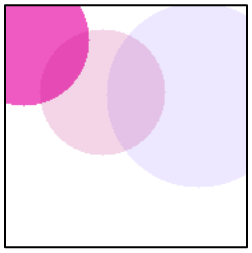


FIGURE 2. DEMONSTRATION OF AN INDIVIDUAL WITH 3 GENES.

Individuals in the evolutionary algorithm possess chromosome lists that store their own genes. In Figure 2, we present a 180 x 180 image as an example, showcasing an individual with 3 genes in its chromosome list. These genes are represented by circles and are initialized randomly. Thus, each circle acquires random values for its radius (r), center position (x, y), and color (R, G, B, A) when an individual is created within the Individual Class.

The population, consisting of individual objects, maintains its members in a population list. Through various operations such as selection, mutation, and crossover, the population is evaluated and updated.

During the evaluation phase, each individual's fitness value is determined. This fitness value indicates the resemblance of an individual's genes (circles) to the source image. The population is then sorted based on their fitness values, with the individual having the highest fitness occupying the first position.

In the context of evolutionary algorithms, elites refer to a group of top-performing individuals within a population. These elites are considered the best solutions found so far and are left unchanged throughout the evolutionary process to ensure their preservation.

Non-elites, on the other hand, participate in a tournament selection. In this process, a random group of individuals, determined by the tournament size parameter, is selected. These individuals compete against each other based on their fitness values, and the winners become parents for the next generation.
During crossover, two parents are chosen, and their genetic information is combined to create offspring or children. This involves exchanging traits between parents to generate new individuals with a mix of characteristics.

The fitness of the children is evaluated, and if they outperform their parents, they replace them in the population. This allows for the propagation of favorable traits and potentially improves the overall fitness of the population.

Following crossover, both the children and the non-elites undergo mutation. Mutation introduces random changes to the genetic material of individuals, providing exploration in the search space and preventing the algorithm from getting stuck in local optima. The probability of mutation is determined by the mutation probability parameter.

After mutation, the resulting individuals, including elites and non-elites, form the next generation. This cycle of selection, crossover, and mutation continues for multiple generations, enabling the population to evolve and converge towards better solutions.

To investigate the impact of each hyperparameter, the algorithm was executed for various values while maintaining default values for the remaining hyperparameters. This approach allowed for a systematic examination of the individual effects of each hyperparameter on the algorithm's performance and behavior.

Default values of the hyperparameters are as follows:

| | | | | | |
|---|---|---|---|---|---|
| **<num_inds>** | = | 20 | **<frac_elites>** | = | 0.2 |
| **<num_genes>** | = | 50 | **<mutation_prob>** | = | 0.2 |
| **<tm_size>** | = | 5 | **<mutation_type>** | = | “guided" |

In *Figure 3,* the process of evaluating the population with the default hyperparameters for 10000 generations is demonstrated. Best individual of the population is selected as the candidate solution at each generation and a resulting image of the best individual is created every 1000 generations. Best Fitness Trend is plotted for different intervals of the generation (iteration) in *Figure 4 and 5*. Best fitness value rises rapidly at the earlier generations and converges to a value slowly through the generations 1000 and 10000. Colors of the sky, light coming from the café, pavement and the tree can be spotted in the final generated image.

Altering the number of individual parameter, the images in Figure *6, 9, 12 and 15* are obtained. When the number of individuals is 40 and 60, the final fitness values are higher comparing to the default case.

## *Default Case*

*Iteration 1000*

*Iteration 2000*

*Iteration 3000*

*Iteration 4000*

*Iteration 5000*

*Iteration 6000*

*Iteration 7000*

*Iteration 8000*

*Iteration 9000*

*Iteration 10000*

*Source Image*

*Best fitness after 10000 generations:*
***-148634320.0***

FIGURE 3. RESULTS FROM THE DEFAULT CASE.

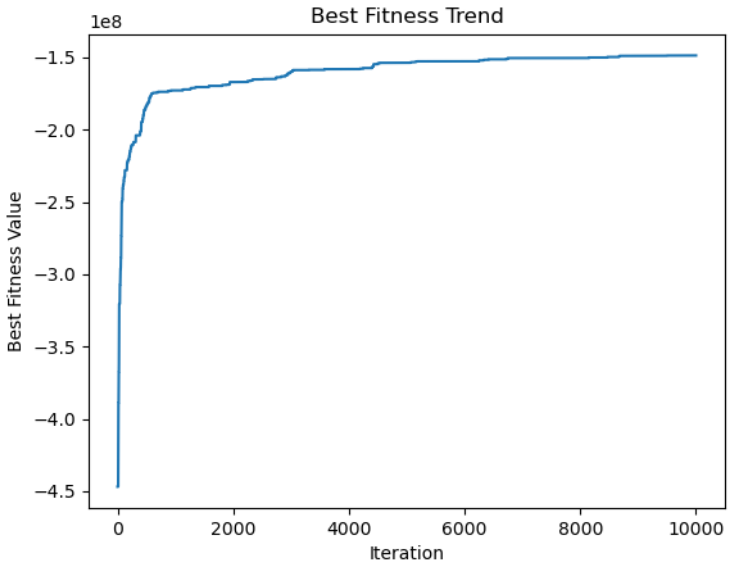



FIGURE 4. FITNESS PLOT FROM GENERATION 1 TO 10000, FOR THE DEFAULT CASE.

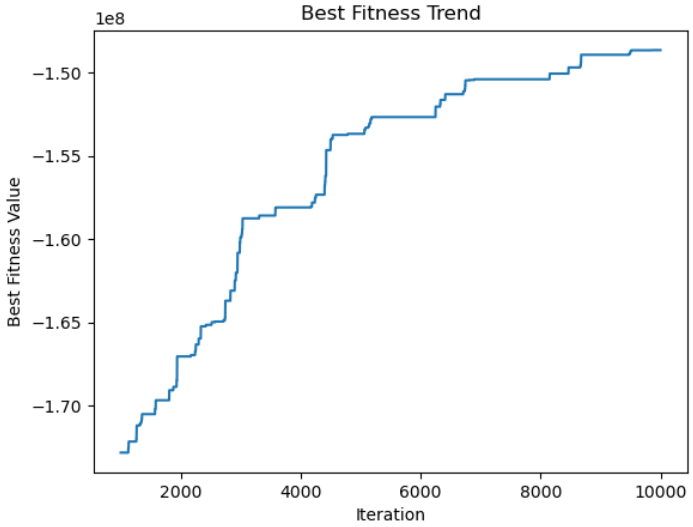



FIGURE 5. FITNESS PLOT FROM GENERATION 1000 TO 10000, FOR THE DEFAULT CASE.

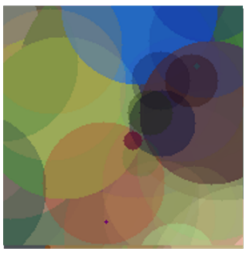
## *Individual number is set to 5*

*Other parameters are set to their default values*

*Iteration 1000*

*Iteration 2000*

*Iteration 3000*



*Iteration 4000*

*Iteration 5000*

*Iteration 6000*

*Iteration 7000*

*Iteration 8000*

*Iteration 9000*

*Iteration 10000*

*Source Image*

*Best fitness after 10000 generations:*
***-157048960.0***

FIGURE 6. RESULTS WHEN INDIVIDUAL NUMBER IS SET TO 5.

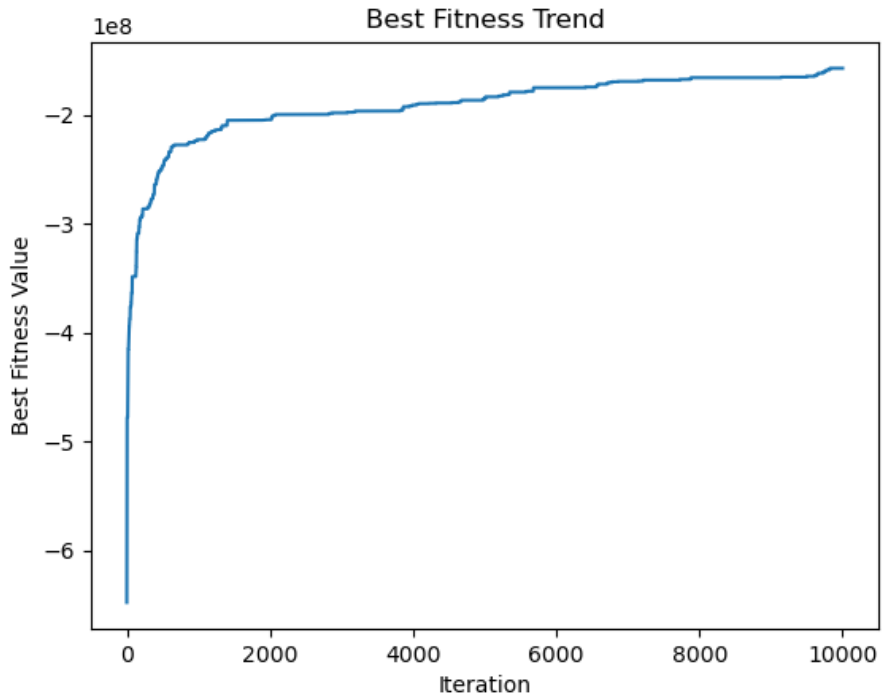



FIGURE 7. FITNESS PLOT FROM GENERATION 1 TO 10000, WHEN INDIVIDUAL NUMBER IS SET TO 5.

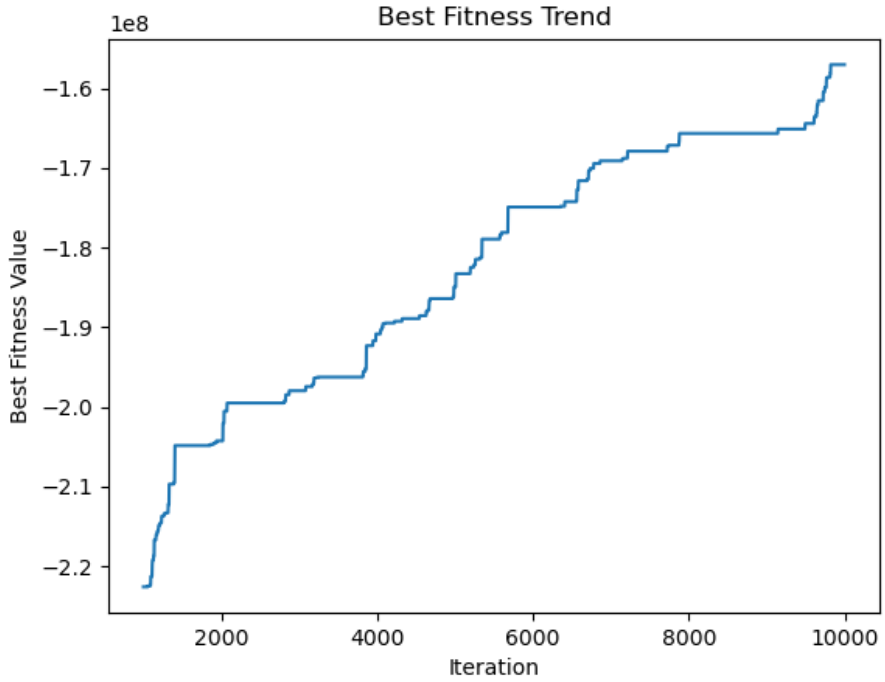



FIGURE 8. FITNESS PLOT FROM GENERATION 1000 TO 10000, WHEN INDIVIDUAL NUMBER IS SET TO 5.

***Individual number is set to 10***

*Other parameters are set to their default values*

*Iteration 1000*

*Iteration 2000*

*Iteration 3000*

*Iteration 4000*

*Iteration 5000*

*Iteration 6000*

*Iteration 7000*

*Iteration 8000*

*Iteration 9000*

*Iteration 10000*

*Source Image*

*Best fitness after 10000 generations:*
***-154604620.0***

FIGURE 9. RESULTS WHEN INDIVIDUAL NUMBER IS SET TO 10.

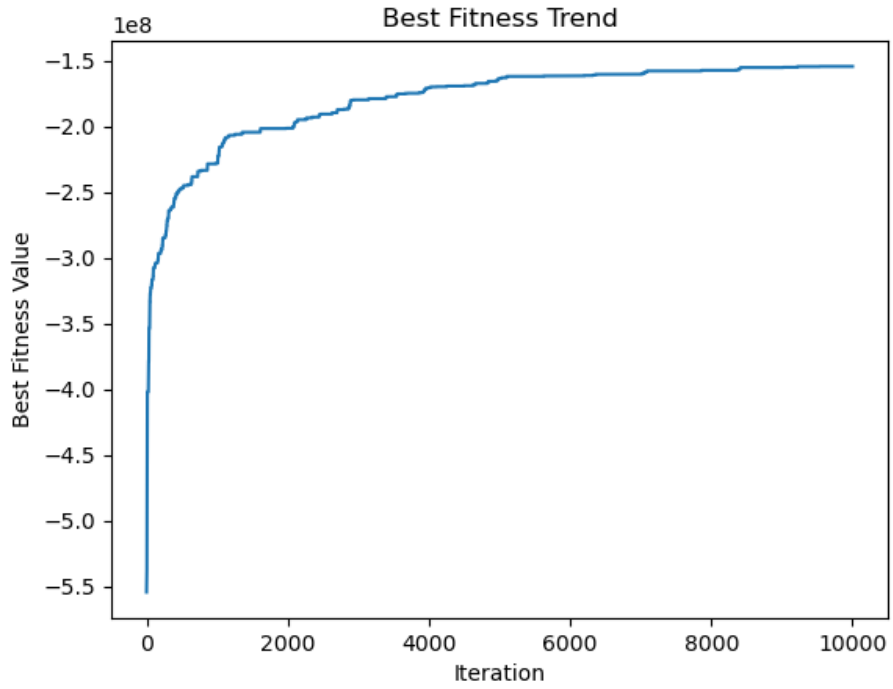



FIGURE 10. FITNESS PLOT FROM GENERATION 1 TO 10000, WHEN INDIVIDUAL NUMBER IS SET TO 10.

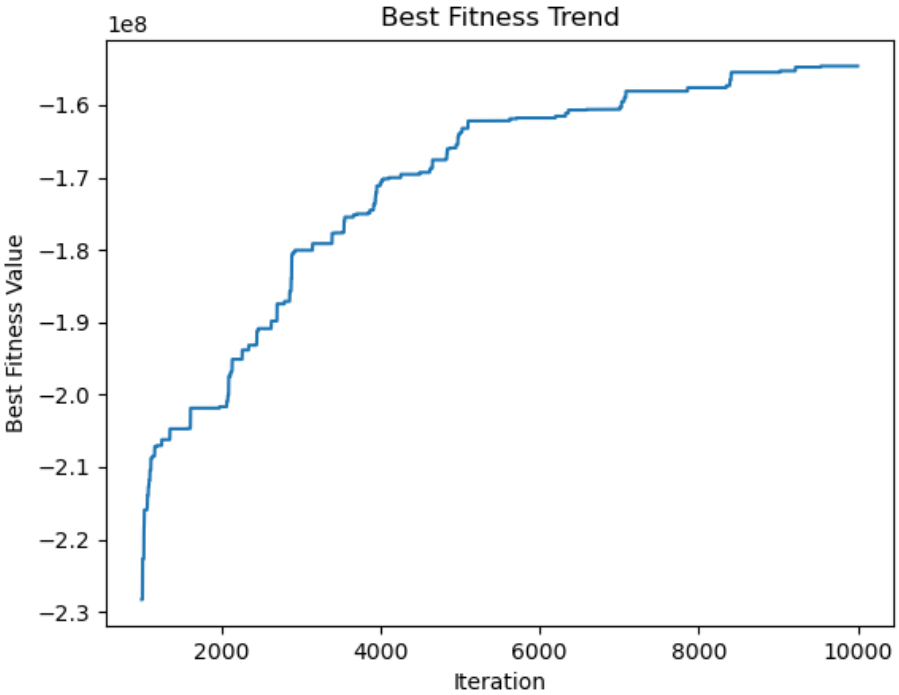



FIGURE 11. FITNESS PLOT FROM GENERATION 1000 TO 10000, WHEN INDIVIDUAL NUMBER IS SET TO 10.

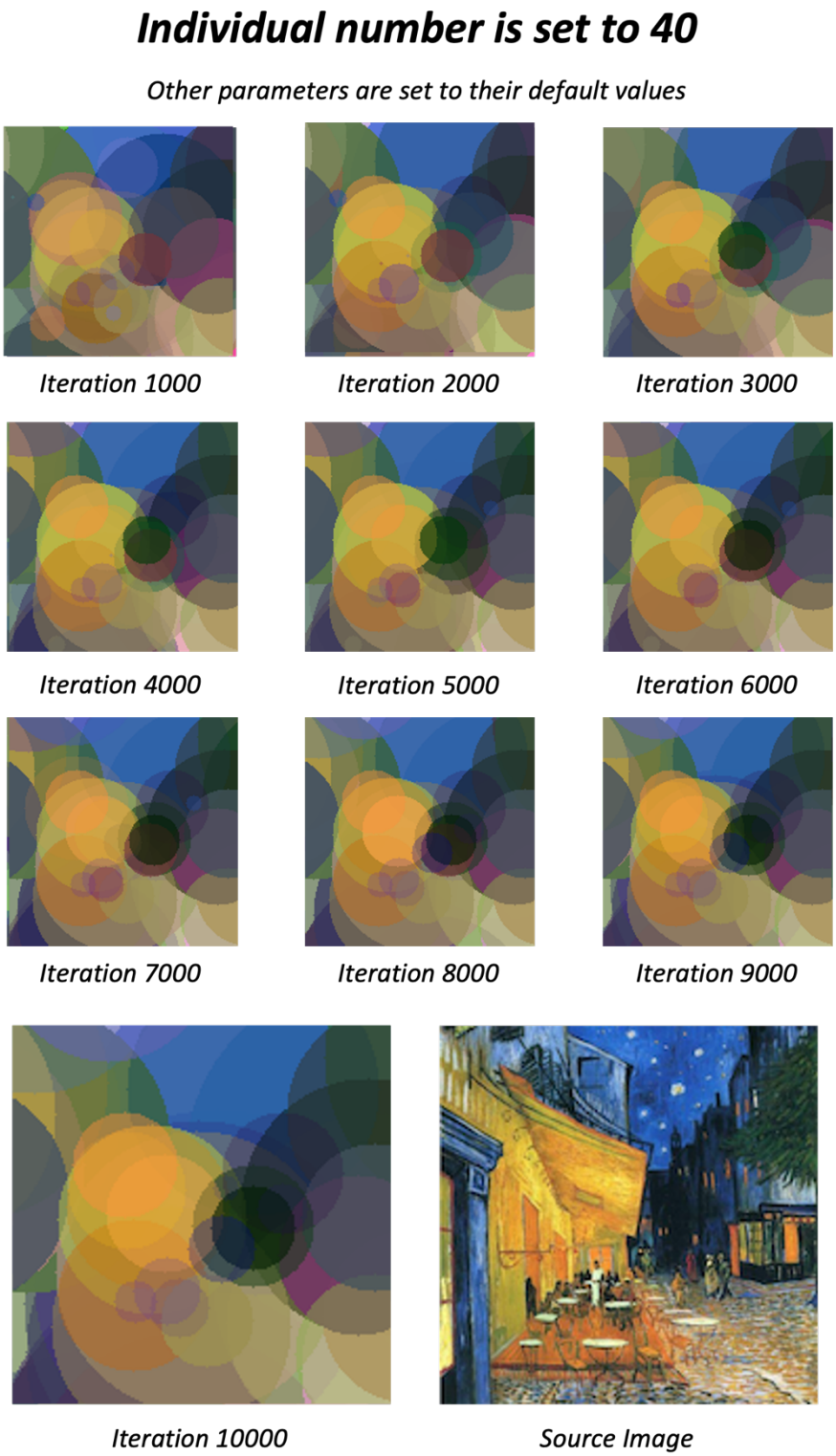



FIGURE 12. RESULTS WHEN INDIVIDUAL NUMBER IS SET TO 40.

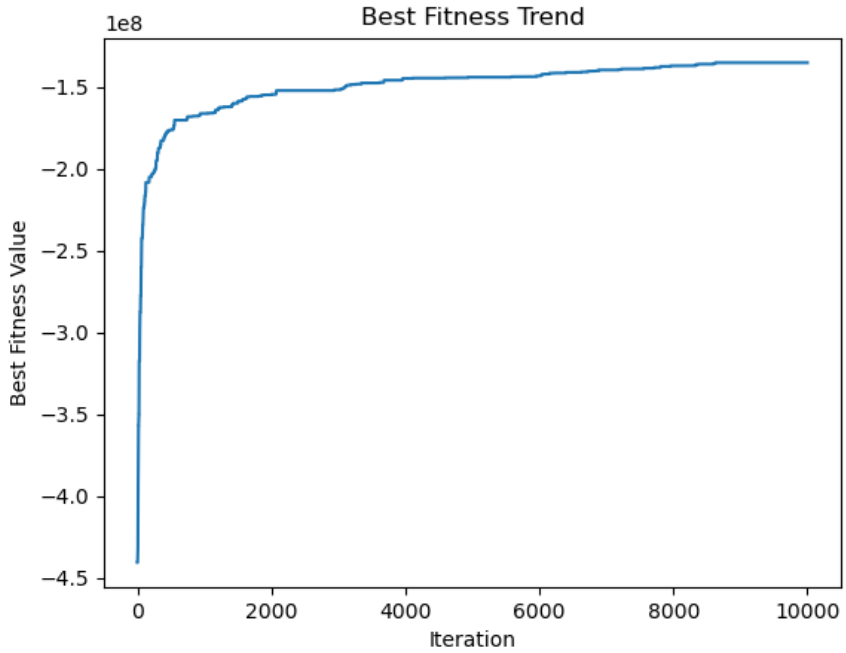



FIGURE 13. FITNESS PLOT FROM GENERATION 1 TO 10000, WHEN INDIVIDUAL NUMBER IS SET TO 40.

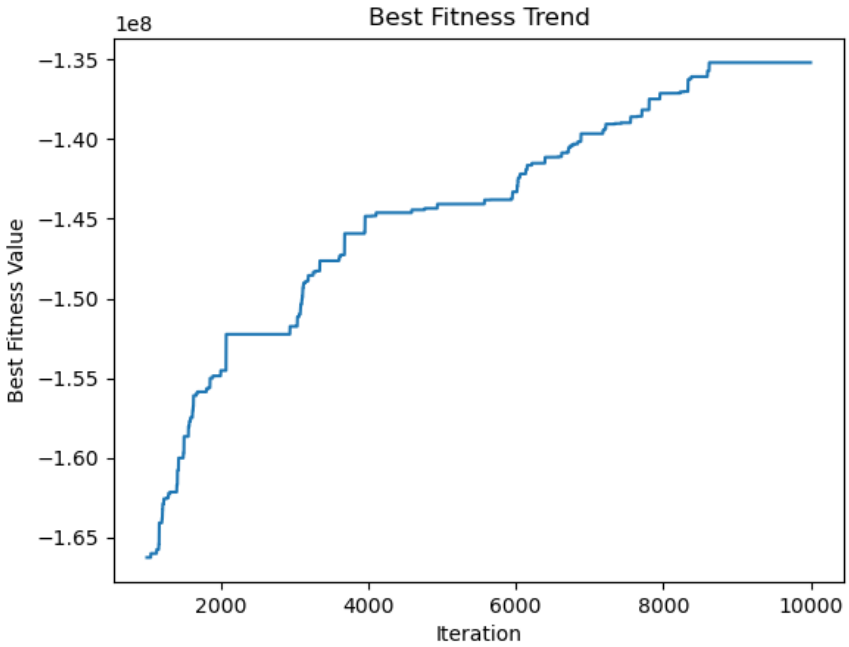



FIGURE 14. FITNESS PLOT FROM GENERATION 1000 TO 10000, WHEN INDIVIDUAL NUMBER IS SET TO 40.

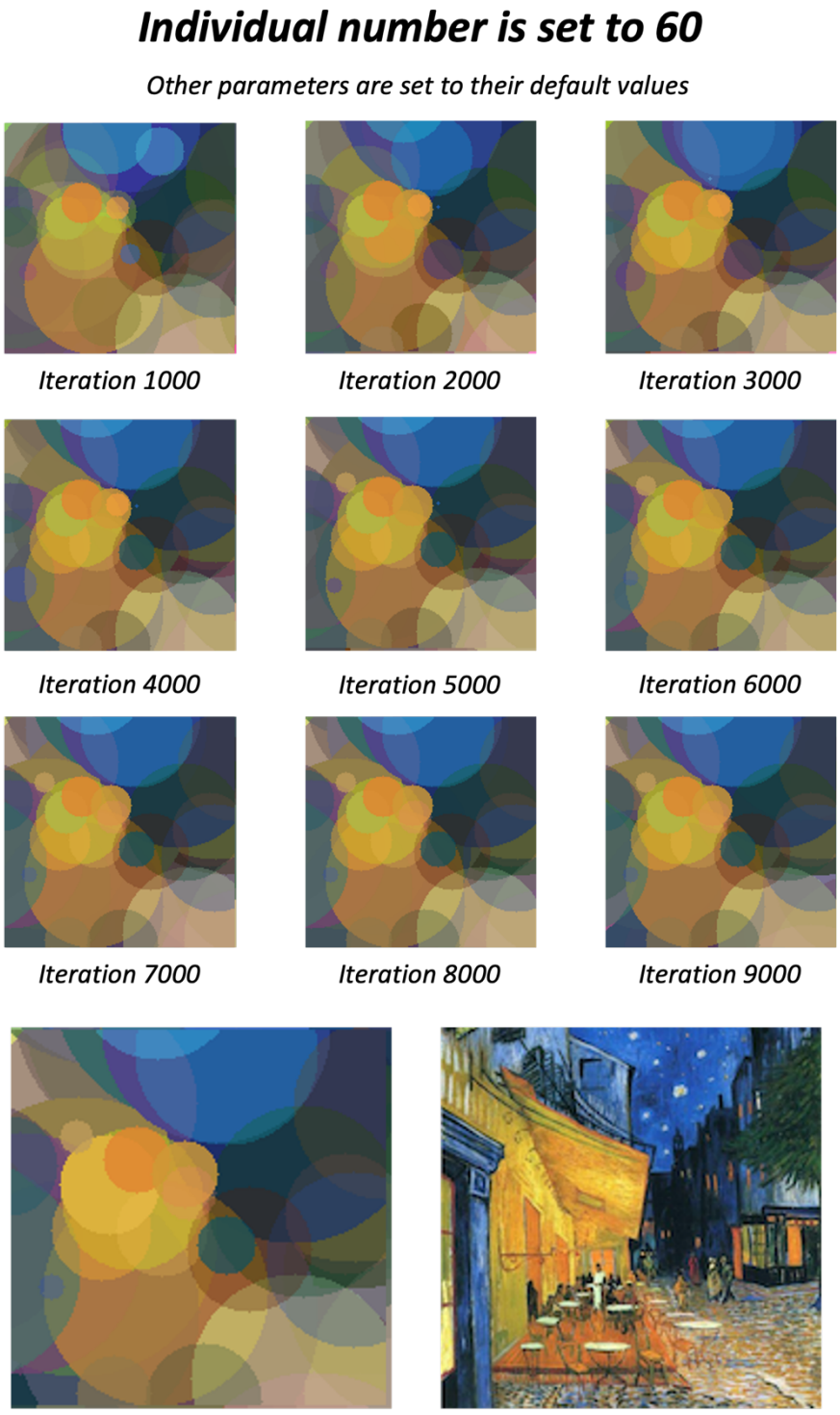



FIGURE 15. RESULTS WHEN INDIVIDUAL NUMBER IS SET TO 60.

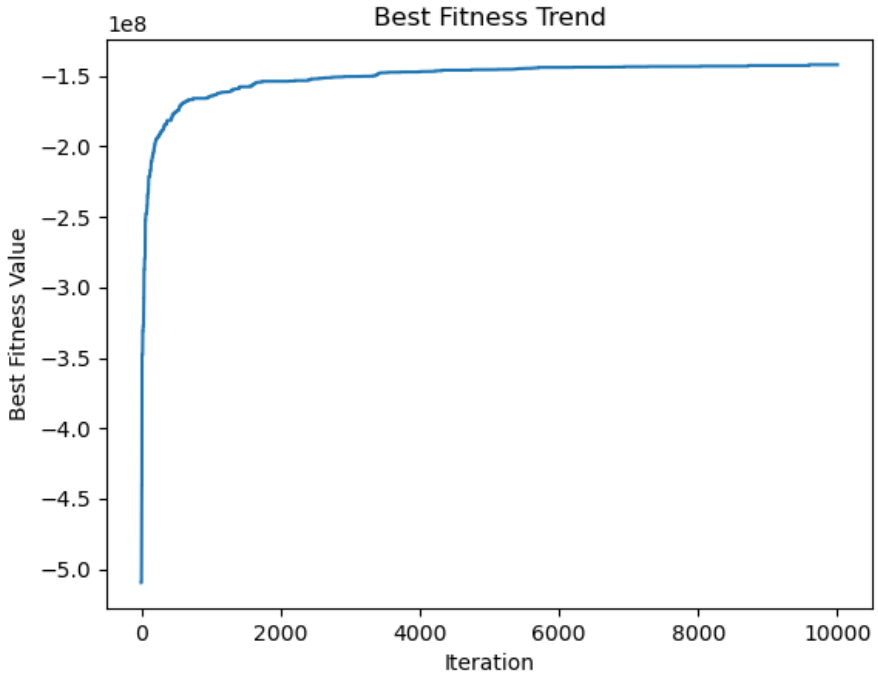



FIGURE 16. FITNESS PLOT FROM GENERATION 1 TO 10000, WHEN INDIVIDUAL NUMBER IS SET TO 60.

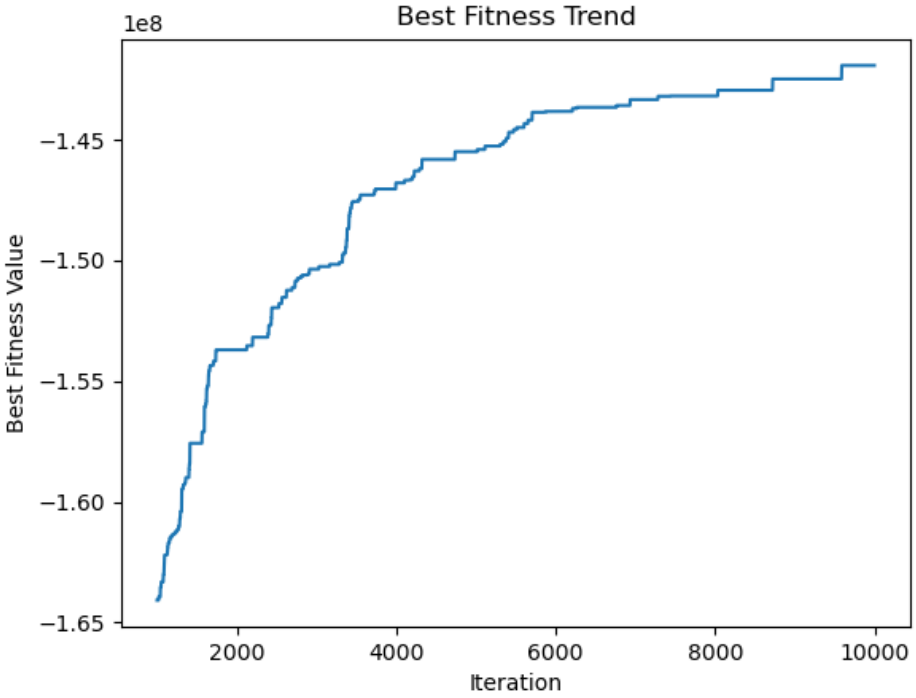



FIGURE 17. FITNESS PLOT FROM GENERATION 1000 TO 10000, WHEN INDIVIDUAL NUMBER IS SET TO 60.

***Number of genes is set to 15***

*Other parameters are set to their default values*

*Iteration 1000*

*Iteration 2000*

*Iteration 3000*

*Iteration 4000*

*Iteration 5000*

*Iteration 6000*

*Iteration 7000*

*Iteration 8000*

*Iteration 9000*

*Iteration 10000*

*Source Image*

*Best fitness after 10000 generations:*
***-171409020.0***

FIGURE 18. RESULTS WHEN NUMBER OF GENES IS SET TO 15.

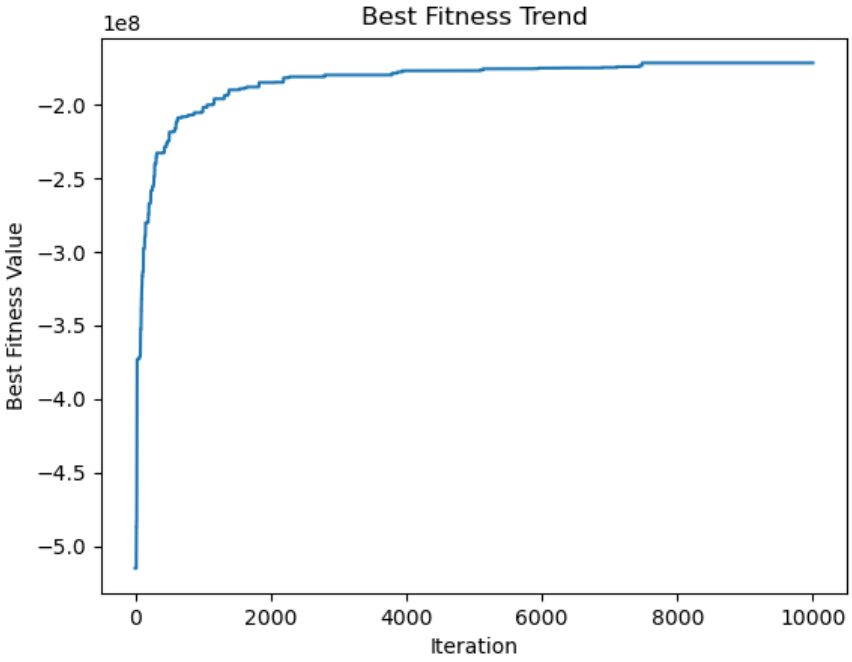



FIGURE 19. FITNESS PLOT FROM GENERATION 1 TO 10000, WHEN NUMBER OF GENES IS SET TO 15.

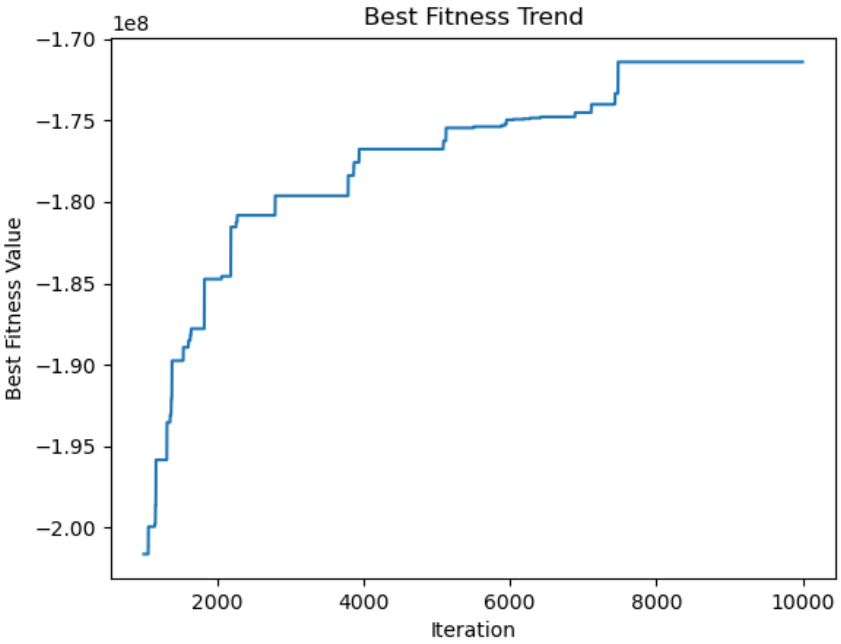



FIGURE 20. FITNESS PLOT FROM GENERATION 1000 TO 10000, WHEN NUMBER OF GENES IS SET TO 15.

## *Number of genes is set to 30*

*Other parameters are set to their default values*

*Iteration 1000*

*Iteration 2000*

*Iteration 3000*

*Iteration 4000*

*Iteration 5000*

*Iteration 6000*

*Iteration 7000*

*Iteration 8000*

*Iteration 9000*

*Iteration 10000*

*Source Image*

*Best fitness after 10000 generations:*
***-155974160.0***

FIGURE 21. RESULTS WHEN NUMBER OF GENES IS SET TO 30.

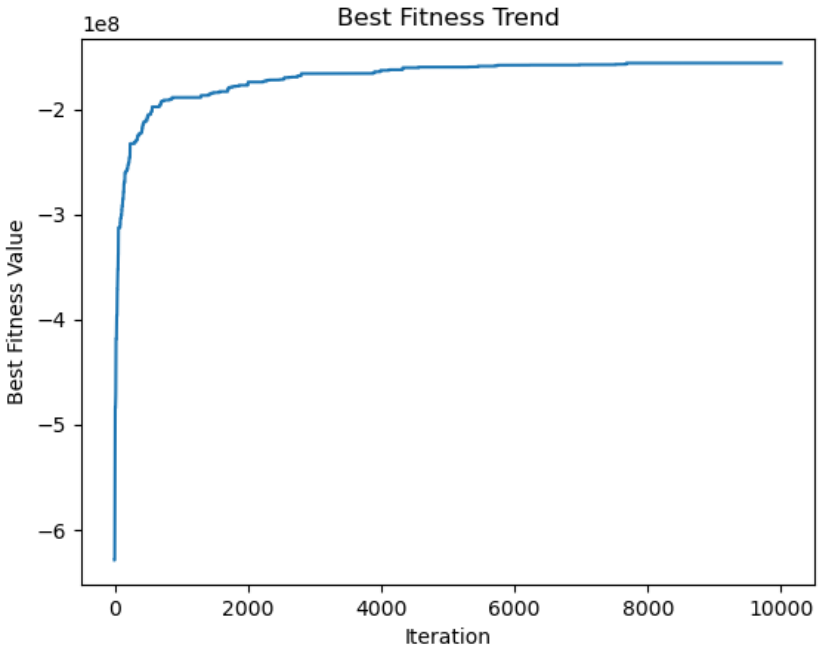



FIGURE 22. FITNESS PLOT FROM GENERATION 1 TO 10000, WHEN NUMBER OF GENES IS SET TO 30.

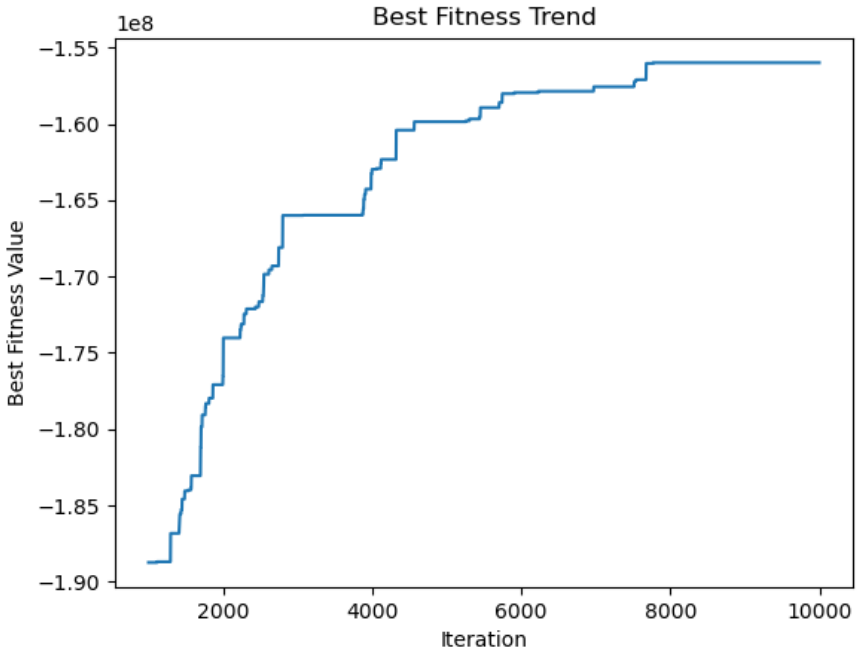



FIGURE 23. FITNESS PLOT FROM GENERATION 1000 TO 10000, WHEN NUMBER OF GENES IS SET TO 30.

***Number of genes is set to 80***

*Other parameters are set to their default values*

*Iteration 1000*

*Iteration 2000*

*Iteration 3000*

*Iteration 4000*

*Iteration 5000*

*Iteration 6000*

*Iteration 7000*

*Iteration 8000*

*Iteration 9000*

*Iteration 10000*

*Source Image*

*Best fitness after 10000 generations:*
***-140602960.0***

FIGURE 24. RESULTS WHEN NUMBER OF GENES IS SET TO 80.

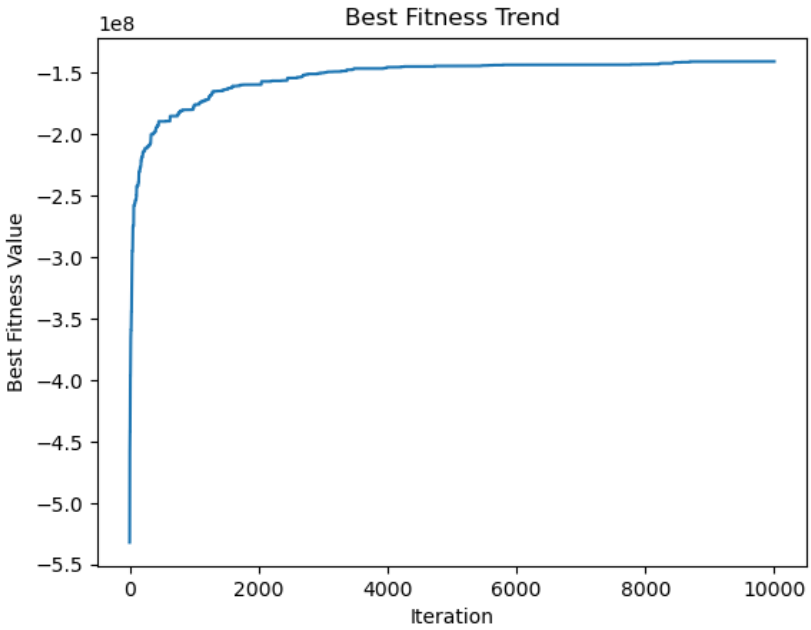



FIGURE 25. FITNESS PLOT FROM GENERATION 1 TO 10000, WHEN NUMBER OF GENES IS SET TO 80.

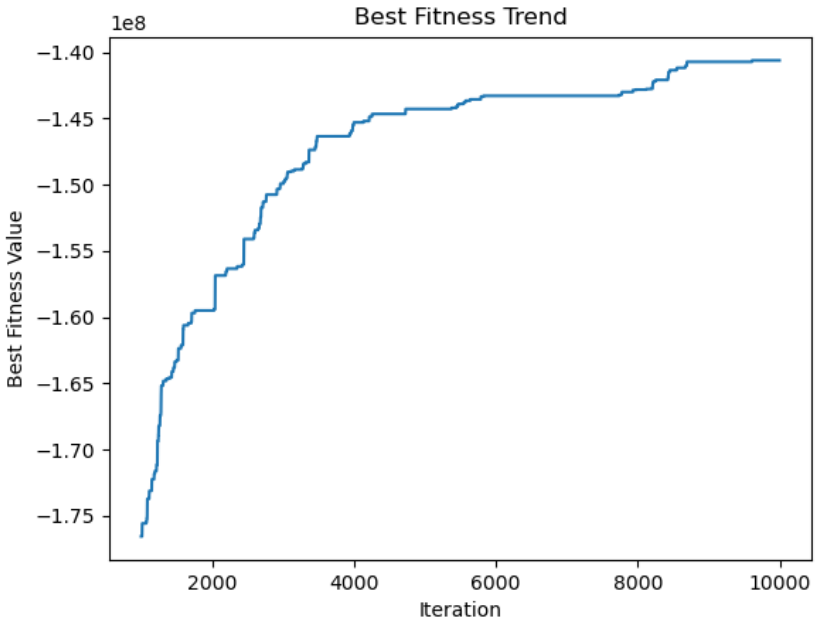



FIGURE 26. FITNESS PLOT FROM GENERATION 1000 TO 10000, WHEN NUMBER OF GENES IS SET TO 80.

***Number of genes is set to 120***

*Other parameters are set to their default values*

*Iteration 1000*

*Iteration 2000*

*Iteration 3000*

*Iteration 4000*

*Iteration 5000*

*Iteration 6000*

*Iteration 7000*

*Iteration 8000*

*Iteration 9000*

*Iteration 10000*

*Source Image*

*Best fitness after 10000 generations:*
***-145649520.0***

FIGURE 27. RESULTS WHEN NUMBER OF GENES IS SET TO 120.

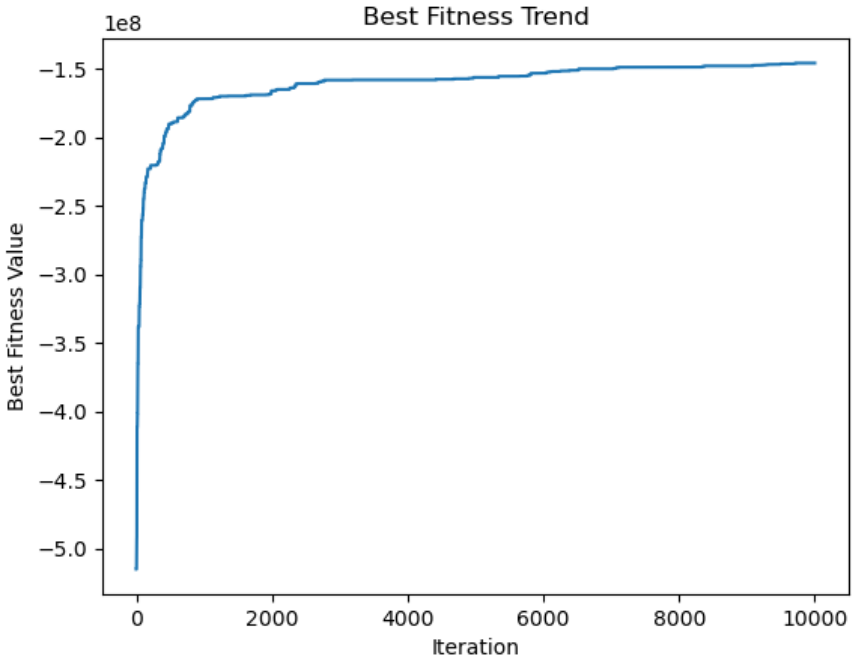



FIGURE 28. FITNESS PLOT FROM GENERATION 1 TO 10000, WHEN NUMBER OF GENES IS SET TO 120.

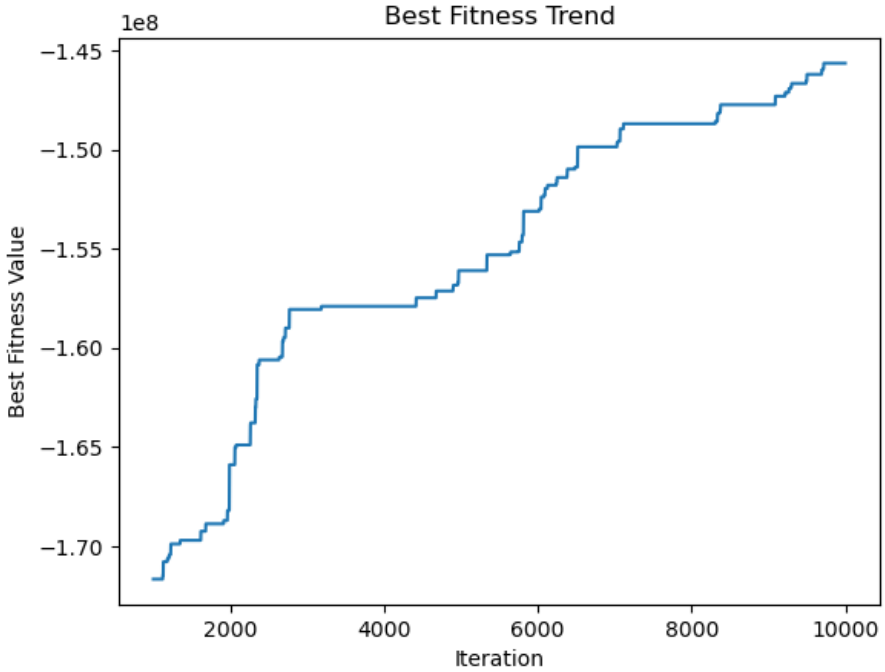



FIGURE 29. FITNESS PLOT FROM GENERATION 1000 TO 10000, WHEN NUMBER OF GENES IS SET TO 120.

***Tournament size is set to 2***

*Other parameters are set to their default values*

*Iteration 1000*

*Iteration 2000*

*Iteration 3000*

*Iteration 4000*

*Iteration 5000*

*Iteration 6000*

*Iteration 7000*

*Iteration 8000*

*Iteration 9000*

*Iteration 10000*

*Source Image*

*Best fitness after 10000 generations:*
***-135579460.0***

FIGURE 30. RESULTS WHEN TOURNAMENT SIZE IS SET TO 2.

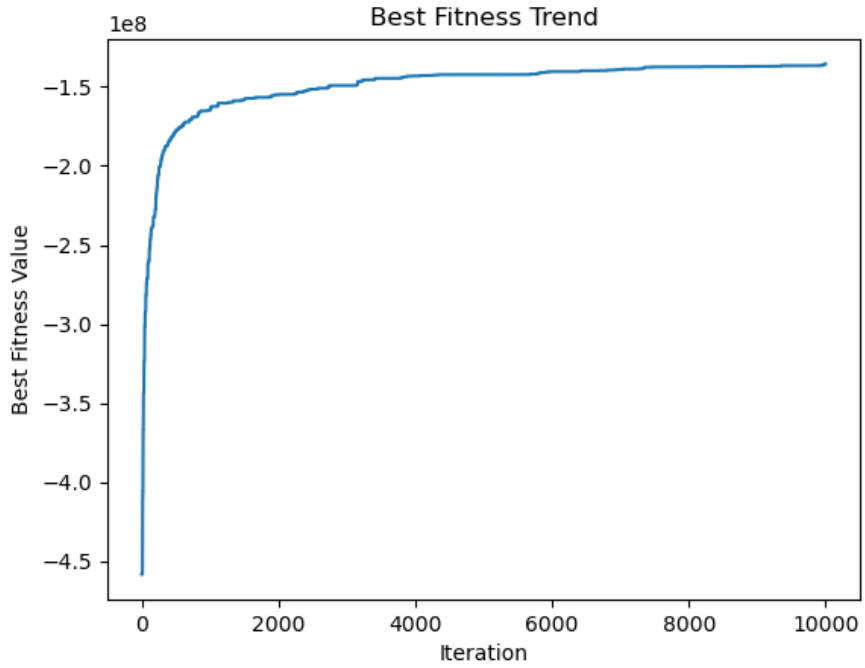



FIGURE 31. FITNESS PLOT FROM GENERATION 1 TO 10000, WHEN TOURNAMENT SIZE IS SET TO 2.

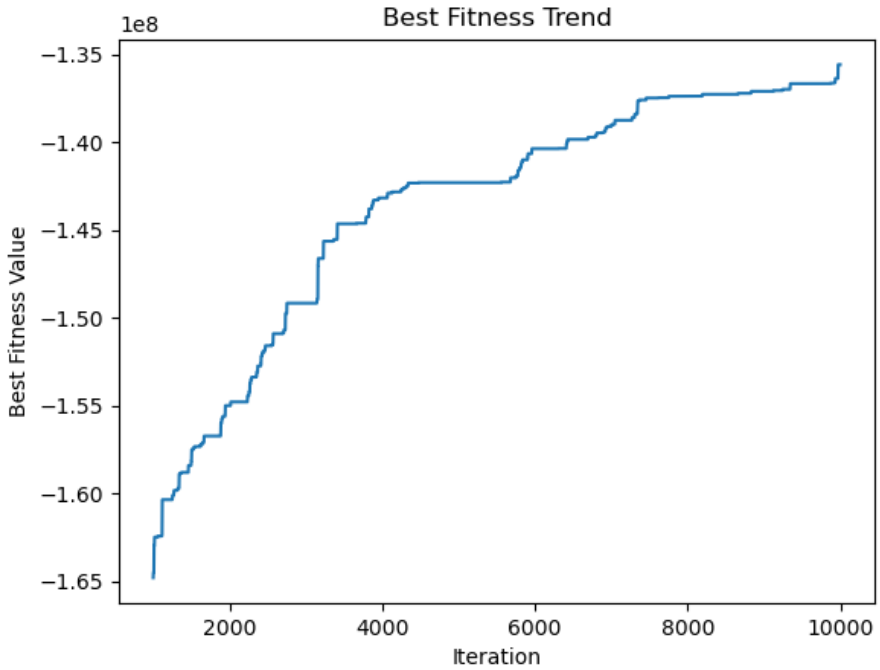



FIGURE 32. FITNESS PLOT FROM GENERATION 1000 TO 10000, WHEN TOURNAMENT SIZE IS SET TO 2.