COMPUTATIONAL INTELLIGENCE FOR OPTIMIZATION

IMAGE RECREATION

USING GENETIC ALGORITHMS

Beatriz Santos, 20230521;

Catarina Ferreira, 20230533;

Diana Silva, 20230586;

Francisco Campos, 20230565;

Tomás Castilho, 20230518;

Git Repository: beatrizdsantosss/cifo-project (github.com)

1. Introduction

The objective of this project is to try and recreate and optimize an image, using genetic algorithms and implementing some of the algorithm provided during the practical classes. In this report, we will explain the implementations we chose and describe the various experiments conducted during the development of this image recreation optimization problem, using different algorithm combinations. Our aim is to determine which combinations of genetic algorithms are most effective for image recreation.

This project was made with the equal contribution of all members of the group, which was crucial due to the time-consuming process of running the code various times for each experiment.

Our work can be found in the following Git Repository: beatrizdsantosss/cifo-project (github.com)

2. Approach

The first step of this project was selecting the image to be recreated. Initially, we considered using the famous 'Starry Night' by Van Gogh. However, preliminary experiments revealed that the complexity of this artwork made it challenging to achieve satisfactory results within a reasonable timeframe. Therefore, to ensure more efficient and rapid progress, we opted for a simpler geometric image of a penguin, that with its clear lines and distinct colour regions, provided a more manageable challenge for our algorithm.

After selecting the image, we had the option of choosing between pixels, polygons, and 2D shapes for the image recreation process. After researching various projects and their methodologies, we decided to use polygons, specifically triangles, which offer a good balance between simplicity and flexibility, beneficial for GA's mutation and crossovers process.

Our project was formulated as a minimization problem, which means we aim to minimize the fitness value to get the closest possible approximation to the target image.

To decide which approach works better, we conducted six experiments by testing three types of crossovers (two-point crossover, uniform crossover, and blend crossover) with two types of fitness functions (SSIM and Euclidean distance). After identifying the best combination, we performed a targeted grid search to explore different parameter combinations further. This allowed us to fine-tune the genetic algorithm's parameters, such as mutation rates, crossover probabilities, and population sizes, to optimize performance and achieve better results.

As we had 2 different combinations that were favourable, we applied the same approach as we did for the crossover methods and fitness functions and ran both combination 5 times.

3. Algorithm

Colour Class: Extracts the dominant colour palette from our target image. The KMeans clustering algorithm is applied to the image pixels to identify the most prominent colours to get a more accurate and closer image recreation to our target image. Even though the image appears to have only three solid colours to the naked eye, the palette extracted is more detailed and varied slightly with each run due to the stochastic nature of the KMeans algorithm.

Triangle Class: Defines the triangles the compose each painting. Supports several types of mutations, which are chosen randomly to introduce diversity and explore new configurations. These mutation types include:

- **Shift Mutation**: involves moving the entire triangle by altering the coordinates of all vertices;
- **Point Mutation:** mutation modifies one of the triangle's vertices;
- **Color Mutation:** changes the colour of the triangle to a new random colour from the palette;
- **Swap Mutation:** involves swapping two of the triangle's vertices, helping maintaining the general shape of the triangle while potentially creating new configurations

For initial testing, we decided to use 100 triangles in each painting to balance detail and computational efficiency, and we tested a 0.01 mutation rate.

Painting Class: A painting is initialized with the specified number of triangles. This class includes a draw method, which uses the Python Imaging Library to draw the painting onto an image, allowing visualization and comparison with the target image.

Fitness Calculation: Initially, we used the Structural Similarity Index (SSIM) to quantify the visual similarity between the generated painting and the target image. However, calculating SSIM was very time-consuming. To address this issue, we tried with the Euclidean distance that provided a reasonable measure of image similarity while being less computationally intensive.

Crossover: We experimented with three different types of crossovers: Two-point crossover, blend crossover, and uniform crossover. Two-point crossover combines parts of two parent paintings by selecting two crossover points and swapping the segments between them. Blend crossover averages the attributes of the parents to produce offspring. Uniform crossover randomly chooses each attribute from one of the parents.

Population Class: Manages the collection of paintings and the evolutionary process and guides the evolution of the individuals to approximate the target image. The population is initialized with randomly generated paintings and evolves over the specified number of generations, applying selection, crossover, mutation, and elitism to evolve the population.

Selection: In this method, a subset of the population is randomly chosen, and the individual with the best fitness in this subset is selected as a parent. We chose this method because of its straight-forward implementation. In this project, a tournament size of 3 was used to provide a moderate selection pressure, ensuring that fitter individuals have a higher chance to be chosen while allowing fewer fit individuals to be chosen, maintaining genetic diversity. Also, larger tournament sizes would increase computational load.

4. Experiments

We conducted six different experiments to combine the three types of crossovers with the two types of fitness functions. Each experiment was run five times to get a better perception and draw accurate conclusions when plotting the values, as each run produces slightly different results. For these experiments, the parameters used were as follows: population_size = 50; num_triangles = 100; generations = 500; mutation_rate = 0.01; crossover_prob = 0.7; and elitism = True. Elitism was always set to true to ensure that the best-performing individuals from each generation were preserved in the subsequent generation, which is crucial for achieving good approximations of the target images. We chose small population sizes and limited the runs to 500 generations to make the experiments easy to run and analyse. Using 100 triangles per painting was considered sufficient to capture the image details while maintaining computational efficiency. The mutation rate of 0.01 was chosen to prevent excessive randomness and ensure a balance between maintaining genetic diversity and stability. The crossover probability of 0.7 was selected to promote a mix of genetic material that explores new solutions and helps escape local optima

We conducted a grid search to fine-tune the parameters and further optimize the performance of the genetic algorithm. The grid search involved systematically varying key parameters such as population size, number of triangles, mutation rate, crossover probabilities and number of generations, to identify the optimal settings that yielded the best image reconstruction. With this grid search we aimed to test 16 different combinations.

5. Results

We plotted the mean of each generation throughout each run to help us visualize which combination gave the best results. As the fitness functions are not in the same scale and it is not correct to normalize them, we had to decide which combination was the most promising and was achieving better results. At this point, we based our decision on visual inspection of the recreated images and the computational efficiency of the algorithm, as both the Euclidean distance and the SSIM fitness functions were recreating different forms of image reconstruction. Based on our interpretation of what was the best result visually and how efficient our algorithm was, the combination that was getting better results was the uniform crossover with the Euclidean distance.

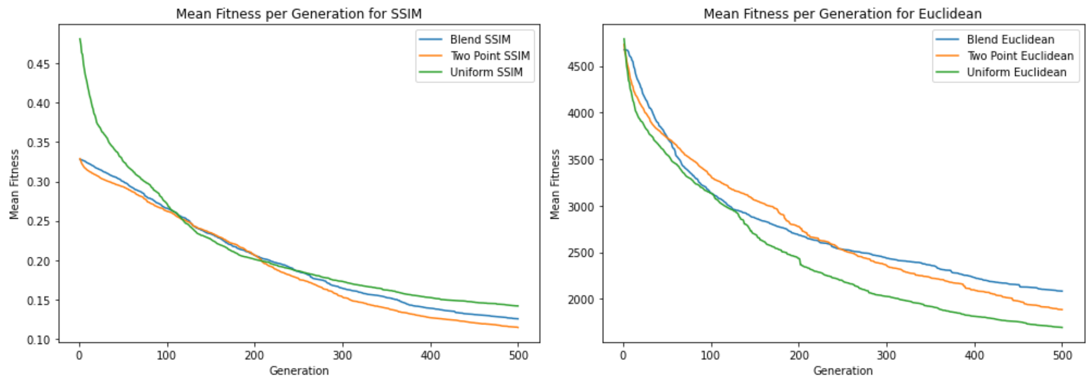

Figure 1: Comparison of mean fitness values for different cross over methods and different fitness functions

After this decision, the grid search was implemented so we could determine the best parameters for the combination (two-point crossover and Euclidean distance) we chose earlier. We chose the 16 following combinations:

Table 1 - Grid Search Combinations and Results

Combination	Parameters	Best Fitness
1	(50, 100, 500, 0.005, 0.6, True)	2001.4080043809158
2	(50, 100, 500, 0.005, 0.7, True)	1844.3711665497267

3	(50, 100, 500, 0.01, 0.6, True)	1699.3478160753318
4	(50, 100, 500, 0.01, 0.7, True)	1714.9559761113403
5	(50, 150, 500, 0.005, 0.6, True)	1767.682946684727
6	(50, 150, 500, 0.005, 0.7, True)	1962.2331665732286
7	(50, 150, 500, 0.01, 0.6, True)	1739.2886476947983
8	(50, 150, 500, 0.01, 0.7, True)	1612.87383226353987
9	(100, 100, 500, 0.005, 0.6, True)	1564.2832224376762
10	(100, 100, 500, 0.005, 0.7, True)	1556.2406626225907
11	(100, 100, 500, 0.01, 0.6, True)	1486.3969860034028
12	(100, 100, 500, 0.01, 0.7, True)	1453.921937381784
13	(100, 150, 500, 0.005, 0.6, True)	1604.4444521391197
14	(100, 150, 500, 0.005, 0.7, True)	1690.5537554304506
15	(100, 150, 500, 0.01, 0.6, True)	1551.4976635496425
16	(100, 150, 500, 0.01, 0.7, True)	1355.993362815615

Between the 16 combinations, there were 2 promising ones, with the lowest fitness values: Combination 16 with population size = 100, number of triangles = 150, 500 generations, mutation rate of 0.01 and a crossover probability of 0.7; Combination 12 with population size = 100, number of triangles = 100, 500 generations, mutation rate of 0.01 and a crossover probability of 0.7.

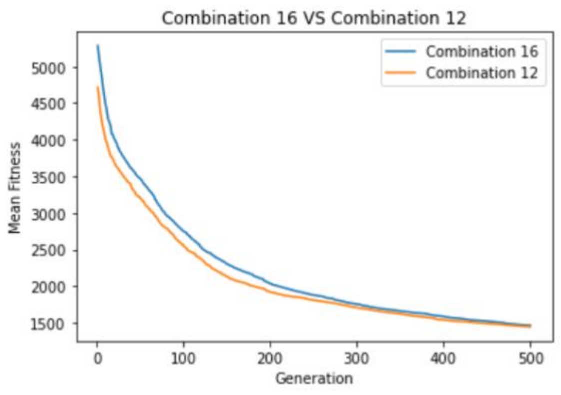

Figure 2 - Comparison between Combination 16 and 12

As we can see in Figure 2, after 5 runs, there isn't much difference between Combination 16 and Combination 12. Nonetheless, Combination 12 was showing constantly lower fitness values, so we decided to select it as the final combination for a last experiment. Even though the results were already satisfactory, we decided to conduct a last experiment with more generations as the algorithm was learning rapidly. For the final experiment we used 3000 generations, while keeping the parameters from Combination 12. By analysing the results, we concluded that by increasing the number of generations, the combination chosen was adequate for what we were expecting to achieve, as you can see with the figures below (Figure 4).

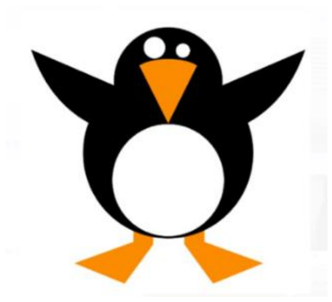

Figure 4 - Original Image

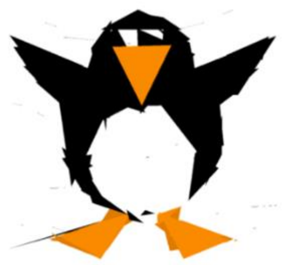

Figure 3 – Last and best experiment

6. Challenges and Future Work

Throughout the project, we faced several challenges. Initially, when attempting to replicate Van Gogh's "Starry Night," we struggled to achieve more than random triangles that failed to capture the painting's details, which lead us to switch to the penguin image. However, even with the penguin image, the process remained highly time-consuming, limiting our ability to explore a wide range of parameters and methods within the available timeframe. More time and better resources would let us test for more generations to get a perfect image recreation. To improve our work there are some additional thing we could do such as: combining mixed probabilities for genetic operators, such as adaptive mutation and crossover rates that adjust based on the population's state; hybrid fitness functions that integrate both SSIM and Euclidean, balancing structural and pixel-wise comparisons; expand the use of triangles to include different polygons, such as rectangles, pentagons, and hexagons, or even combine polygons with pixels.

7. Conclusion

In this project, we were tasked with solving an optimization problem using Genetic Algorithms (GAs) as covered in the course. We successfully applied a genetic algorithm to the problem of image recreation and optimization. Through the implementation and experimentation with various algorithm combinations — using different fitness functions, crossovers, and parameters — we were able to determine that the more effective model was achieved using Combination 12 with 3000 generations. Even though we couldn't achieve the 'Starry night' as we wanted in the beginning, we were still able to achieve an almost perfect image recreation using a simpler image.

8. References

AnalytixLabs. (2024, January 29). A complete guide to genetic algorithm — advantages, limitations & more. Medium. https://medium.com/@byanalytixlabs/a-complete-guide-to-genetic-algorithm-advantages-limitations-more-738e87427dbb

Proost, S. (2020, January 12). Genetic Art Algorithm. 4DCu.be. https://blog.4dcu.be/programming/2020/01/12/Genetic-Art-Algorithm.html

Vanneschi, L., & Silva, S. (2024). Lectures on Intelligent Systems. Springer.