Yacoub Al Karadsheh (ya217)

Abstract

In this research, I aim to explore the use of SGAs in generating artistic pieces such as paintings or drawings. My main objective is to develop an SGA algorithm that can take inputs such as color scheme, and line patterns on canvas size to generate an aesthetically pleasing and visually engaging art. Moreover I will go into how the art is created, such as the parameters of the population size, crossover, and mutation to generate new offspring with potentially improved parameter sets, as well as the fitness function of how it works. Additionally, the algorithm's artistic output will be presented visually, allowing us to admire the beauty of the images that are produced which I will then be evaluating. The research thus demonstrates the adaptability and strength of this computational technique by highlighting how the principles of genetic algorithms may be implemented in a variety of fields, from game-playing to the creation of art. Lastly I go into how there can be an upgrade of the artwork and the system as a whole, as well as some potential improvements and a final reflection on the project.

1. Introduction

A type of art that autonomously develops itself is known as generative art, also known as procedural art or algorithmic art. Some examples of art take shape through the use of mathematical formulas and following logical paths, while other examples use the power of chaos and produce artwork through random settings within the parameter. However, some artists choose to use both in combination to produce some pleasing art pieces.

My project idea was an attempt to grasp the flare of the chaos through the use of randomization, while combining the refinement of mathematical formulas to create some sort of patternized art work. With these two aspects in mind I believed that the use of an SGA would be the perfect implementation. An SGA approach gives me the control of the mathematical formula through the use of a fitness function, and coupled with randomization of the parameters of how the lines are drawn.

This paper is organized through the following sections. Section 2 (Background) introduces research regarding genetic algorithms as well as the processes I went through deciding my project. Section 3 (Methodology and design) outlines the steps of the genetic algorithm used in this approach and how the software was designed, implemented and tested. Section 4 (Results) describes the results of the SGA algorithm for generative art which demonstrated visually engaging and aesthetically pleasing pieces with potential for further artistic development. Section 5 (Evaluation) discusses the evaluation of the output through the lens of Colton's Creative Tripod, which consists of the three elements: skill, appreciation, and imagination.

2. Background

My project was inspired by Jonathan Chaffer who wrote a blog post about learning how to create generative art [1]. In this blog post he goes through an example of a simple

generative art through the use of drawing straight diagonal lines using Turtle Graphics in python. With each run the output came out the same. Using the techniques used by Danny Dijkzeul et al. [2] I took it upon myself to enhance this idea by incorporating an SGA algorithm and randomizing color pallets for each time a line is drawn. Additionally I gathered some inspiration from another project done by Misha Paauw and Daan van den Berg who used polygons to create art [3]. Through this paper, it gave me further insight on the combination of using an SGA algorithm with shapes and lines, and my main inspiration of using Jonathan Chaffer's project. I was able to better understand the randomness needed as well as how it may correlate to a better output through use of my fitness function.

With these three papers I was able to get a foundation on the project as well as an idea on how it may look. Through further investigation and testing I was able to determine the tools I needed. First I went through testing python libraries that could work well with the idea I had in mind. After multiple attempts I found that using the DEAP library from python coupled nicely with my methodology. The novel evolutionary computation framework DEAP allows for quick idea testing and prototyping. It aims to make data structures transparent and algorithms more explicit [4]. Moreover DEAP allowed me to use the "tools" library, which is used to register methods and functions to key words, which made the design process substantially easier to work with.

3. Methodology and design

The Python programming language and the DEAP (Distributed Evolutionary Algorithms in Python) genetic algorithm package were used in the technical construction of this innovative software. Using a genetic algorithm, a population of individuals are compiled, each of which represents a potential answer to the issue. In this instance, each individual represents a drawing created by the Python graphics module's turtle. A fitness function is then used to assess each individual's fitness, scoring each drawing according to a variety of factors, such as how much of the canvas is covered or how much contrast is present in the picture.

The population is then subjected to several genetic operators, such as crossover and mutation, by the genetic algorithm to produce new individuals. The fittest individuals are chosen to make up the next generation, while the weakest individuals are eliminated. In the hope that population fitness will increase over time and eventually converge on an ideal solution, this process is repeated for several generations.

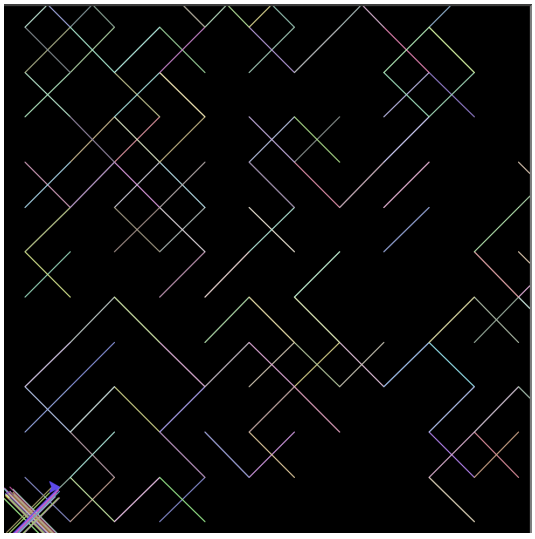
The turtle graphics module, which enables the generation of digital drawings using Python code, was also integrated as part of the project. The turtle module offers a straightforward and natural way to draw shapes and lines on a canvas and allows for the customization of many elements including color, thickness, and drawing speed. The fitness function is intended to assess the drawings produced by the turtle module according to specific standards and provide a fitness score as a result. resulting in the creation of art with a range of colors and patterns that could be seen as visually pleasant.

The technical implementation of the creative software involved a combination of genetic algorithms, fitness functions, and the turtle graphics module in Python. The goal was to create a

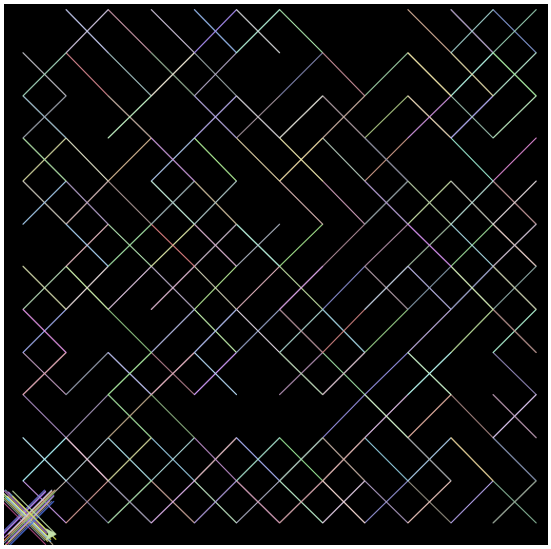
system that could generate aesthetically pleasing and visually interesting drawings through an iterative and evolutionary process.

4. Results

Here are some example outputs that were created through my program. Please note that with each time the program is run the output may vary due to the randomness.



(Fig 1) With the parameters of population size: 3 and number generations: 3



(Fig 2) With the parameters of population size: 5 and number generations: 5

The SGA algorithm's ability to produce artistic works has the intriguing property that altering certain parameters might provide remarkably varied results. For instance, modifying population size or number generation can have an impact on the population's variety and could

result in either more or less variation in the offspring, which can result in interesting patterns and works of art. The artistic criteria that the algorithm optimized for, such as emphasizing particular colors or shapes, can also be changed by modifying the fitness function. The two instances given above show how this is possible.

A greater population size tended to produce more intricate and complicated artwork, whereas a smaller population size resulted in simpler paintings, as was discovered through parameter experimentation. More random and chaotic art was created when the mutation rate was higher; on the other hand, more structured and orderly artwork was created when the mutation rate was lower. Changes to the fitness function that prioritized particular hues or shapes had a large effect on the result.

Overall, the code's findings show how adaptable and flexible SGAs can be when creating artistic works because even minor adjustments to the parameters can provide wildly diverse results. This opens up a vast array of opportunities for experimenting and investigating the algorithm to create various forms of art.

5. Evaluation

I made the decision to assess the results of the SGA algorithm I developed to produce creative works based on Colton's Creative Tripod, which consists of three components: skill, appreciation, and imagination.

Skill

The technical competence and artistic execution of the task are referred to as skill. The technical skill in the SGA algorithm is the capacity to produce visually beautiful art from the input parameters. The algorithm's technical expertise is demonstrated by the creation of visually appealing and agreeable sight of art.

Appreciation

The power of the artwork to provoke an emotional response from the audience is referred to as appreciation. The generated artwork produced by the SGA algorithm could not have the same emotional impact as a work of art made by a human artist with a particular intention or message. However, the algorithm's capacity to produce original and aesthetically pleasing art might be able to win the viewer over with the color palettes as well as the patterns it could make through each generation.

Imagination

The originality and inventiveness of the artwork are referred to as imagination. The inventiveness and uniqueness of the SGA algorithm is demonstrated through the generation of creative and aesthetically beautiful art based on the input parameters. The algorithm may not, however, have the same amount of creativity as a human artist who can produce art based on their own experiences and feelings because the algorithm is constrained by the input parameters.

Overall, the SGA algorithm for creating artistic creations displays a high level of technical proficiency and appreciation in terms of Colton's Creative Tripod. Because, as could be seen, the algorithm can produce visually appealing art that demonstrates its technological prowess. In contrast to art made by human artists, the algorithm may fall short in terms of imagination because it lacks the intensity and profundity that frequently result from human expression and creativity.

6. Conclusion

In summary, this research study investigated the usage of SGAs in producing artistic works utilizing the Python DEAP module. The goal was to create an SGA algorithm that could use inputs like color scheme, line pattern, and canvas size to produce an appealing and eye-catching work of art. The research highlighted how the principles of genetic algorithms may be applied in a range of domains, from game-playing to the creation of art, demonstrating the flexibility and power of this computational technique.

The SGA algorithm for generative art produced visually interesting and aesthetically pleasing works that had the potential to advance the field of art. Colton's Creative Tripod was used to evaluate the output, and the results showed that the algorithm produced artwork that exhibited the three components of talent, appreciation, and creativity.

Future work on this project will focus on improving the fitness function, investigating more intricate line patterns and shapes, and investigating how deep learning can be used to produce more complex generative art. Overall, the project has shown that SGAs may be used to create art, and there are many opportunities for additional research in this area.

References

[1]
Chaffer, Jonathan. “Learn How to Make Generative Art: From Zero to Random().” *Atomic Spin*, 14 Dec. 2021, spin.atomicobject.com/2021/12/14/generative-art-zero-random/#:~:text=There%20are%20many%20tools%20available. Accessed 30 Apr. 2023.

[2]
Dijkzeul, Danny, et al. “Painting with Evolutionary Algorithms.” *Artificial Intelligence in Music, Sound, Art and Design*, 2022, pp. 52–67, https://doi.org/10.1007/978-3-031-03789-4_4.

[3]Paauw, Misha, and van. “Paintings, Polygons and Plant Propagation.” *Springer EBooks*, 24 Apr. 2019, pp. 84–97, https://doi.org/10.1007/978-3-030-16667-0_6. Accessed 30 Apr. 2023.

[4]“Overview — DEAP 1.3.3 Documentation.” *Deap.readthedocs.io*, deap.readthedocs.io/en/master/overview.html. Accessed 30 Apr. 2023.