An Approach of Improving Reproducing Image Using Genetic Algorithm

Tzu-Chun Chiu
Institute of Computer Science and Engineering
National Yang Ming Chiao Tung University
HsinChu City 300, Taiwan
alan890526@gmail.com

Chia-Yu Pai
Department of Computer Science
National Yang Ming Chiao Tung University
HsinChu City 300, Taiwan
jerrypai520@gmail.com

Ke-Syuan Shen
Department of Computer Science
National Yang Ming Chiao Tung University
HsinChu City 300, Taiwan
ksshen.cs08@nycu.edu.tw

ABSTRACT

This paper presents an overall approach of improving reproducing images using genetic algorithm. We found a paper shows how to use genetic algorithm reproducing images. The genetic algorithm starts from a randomly generated image of the same shape of the input image and reproduces the image to match the target image. However, we find out there exists some space of progress to be improve the whole algorithm. We present the changes we have done to the genetic algorithm to get better performance. In the experiment, we make a comparison of performance between different crossover methods in genetic algorithm. Parallel programming is also performed in the algorithm to accelerate the computing speed.

KEYWORDS

Reproducing image, Genetic algorithm, Crossover, Parallel

1. INTRODUCTION

Genetic algorithms are techniques based on the natural evolution by Darwin. Species of creatures inherit traits from their parents and try to survive in the environment. Those creatures that can fit the environment well can survive and have offspring. Genetic algorithms mimic this mechanism by introducing artificial selections and operators to explore and recombine partial solutions.

We use genetic algorithms to solve a problem called reproducing images. First, we have an input image as a target. Our program will generate a random image of the same size. After 20000 steps, our genetic algorithm will generate an image that is very similar to the target image. The start of the search for an optimal solution begins with a randomly generated population of chromosomes (the initial image). Each set of generations will have a newer set of chromosomes found from the application of the operators. A fitness function is definition related to the problems. The parent selection process makes sure that the fittest individuals will survive and have next generations by crossover. The algorithms terminate when the generation number reach. Finally, we could check the similarity of the image we produced with the target image.

2. RELATED WORK

The project Genetic Algorithm for Reproducing Images (GARI) [1] has already given a set of genetic algorithms to solve the exact same problem. Their solutions provide a good result after 20,000 generations. GARI uses the traditional structure of genetic algorithms. In the chromosome, they transform the image that has RGB three channels into a large 1-D array as shown in the Figure 1. About the crossover, GARI applied the simplest single point crossover method, which cuts the chromosome in a picked position and exchanges them to construct a new child as shown in the Figure 2. Besides, in parent selection, they selected from a given population are the best solutions within it. In mutation, GARI uses the random mutation that gives a randomly picked gene a random value between a self-defined range.

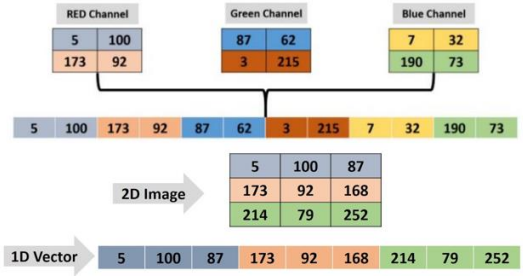

Figure 1: Data representation (GARI)

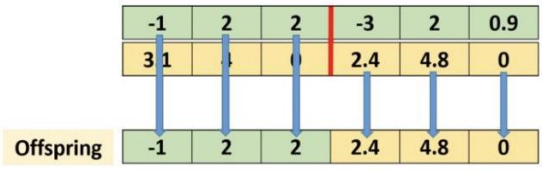

Figure 2: Single point crossover

The fitness function is the sum of target image for each pixel minus difference between target image and generated image. It can be defined as

$$Fitness = \sum_{pixel} Image_t - \sum_{pixel} abs(Image_t - Image_g) \quad (1)$$

where $Image_t$ is the target image and $Image_g$ is the generated image.

3. METHOD

It is well known that an image is composed of three channels. GARI directly encode the image into a one-dimensional vector. This chromosome representation is difficult to generate better offspring when combined with one point crossover. Therefore, we propose a new algorithm architecture for better performance. Our improvement of the algorithm can be divided into three parts. The first is to change the representation of the chromosome and the fitness function. Next, three different crossover methods were tried to reproduce the image, including single arithmetic, linear, and blend. Finally, the algorithm can be parallelized to speed up the calculation.

Next, the implement methods we use in the genetic algorithm are introduced.

3.1 Chromosome Representation

In terms of the representation of the chromosome, the image is separated to RGB three channels and each pixel value is normalized into 0 to 1 from range 0 to 255. Because the image has been divided into three RGB channels, the image will change to three 2D arrays from the original 3D array, and then connect each row in order to form a 1D array as shown in the Figure 3.

Figure 3: Chromosome

3.2 Fitness Function

The fitness function of genetic algorithms use the sum of the number of pixels to subtract the difference between the target image and the solution, so genetic algorithm deal with the problem of finding the maximum value. It can be defined as

$$Fitness = number\ of\ pixel - \sum_{pixel} abs(Image_t - Image_g) \quad (2)$$

where $Image_t$ is the target image and $Image_g$ is the generated image.

3.3 Single Arithmetic Crossover

The first method is single arithmetic crossover. In the single arithmetic crossover, we need to choose a value between zero and one as alpha. The way to generate offspring can be define as

$$Child = \alpha \times parent_1 + (1 - \alpha) \times parent_2 \quad (3)$$

The example as shown in the Figure 4, the value of alpha is 0.5, and the child's first gene can be calculate as 0.8 multiply 0.5 plus 0.6 multiply 0.5, so the first gene's value is equal to 0.7. Other genes also calculate in this way.

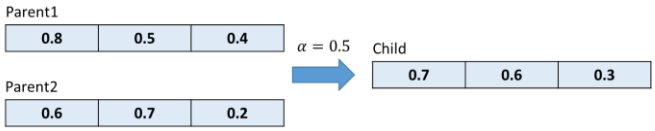

Figure 4 : Single arithmetic crossover

3.4 Linear Crossover

The second method is linear crossover. Given the two parents p_1 and p_2, generate three solutions C_1, C_2 and C_3 at one time. The solutions can be defined as

$$C_1 = 0.5 \times p_1 + 0.5 \times p_2 \quad (4)$$

$$C_2 = 1.5 \times p_1 - 0.5 \times p_2 \quad (5)$$

$$C_3 = -0.5 \times p_1 + 1.5 \times p_2 \quad (6)$$

so the solutions generated are located on the left, middle and right of the two parents respectively as shown in the Figure 5, and then pick the best fitness from these three solutions as the child.

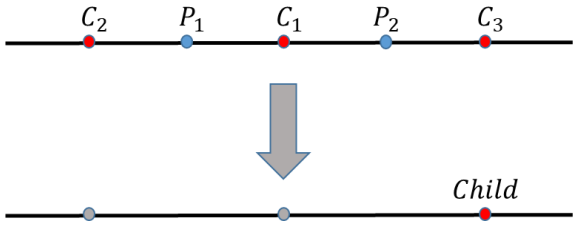

Figure 5: Linear crossover

3.5 Blend Crossover

The last one method is blend crossover. Given the two parents p_1 and p_2 which p_1 is less than p_2. We need to choose a value between zero and one as alpha, and then calculate the range of an interval. The limit of the interval can be defined as

$$lower\ limit = p_1 - \alpha(p_2 - p_1) \quad (7)$$

$$upper\ limit = p_2 + \alpha(p_2 - p_1) \quad (8)$$

As shown in the Figure 6, the child's gene values are randomly generated by uniform distribution, and they are in the yellow field.

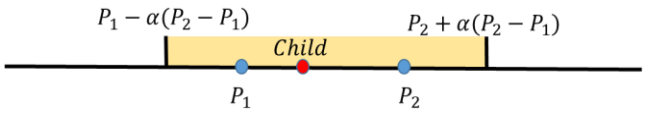

Figure 6: Blend crossover

3.6 Parallel

Because the proposed algorithm architecture divides the image into three channels of RGB and then performs genetic algorithms as shown in the Figure 7. These three genetic algorithms are independent and don't affect each other, so we can use the multi-process to simultaneously perform calculations on three genetic algorithms to speed up the calculation speed of the algorithm. According to our experimental results, the computing speed can increase about two times.

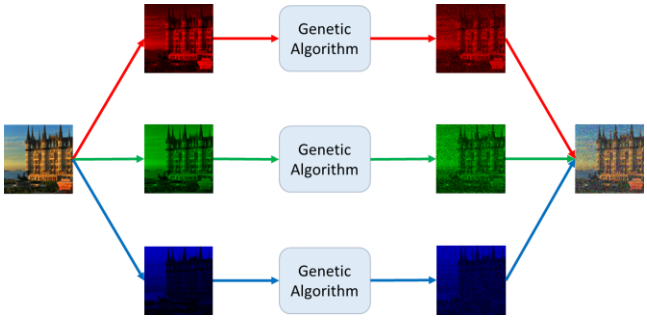

Figure 7: Algorithm architecture

4. EXPERIMENT

We conduct many experiments and statistical results to demonstrate the effectiveness and efficiency of our method.

4.1 Reproduction of the testing image

In order to facilitate the observation and comparison of the results, we restricted the parameters of genetic algorithm as below.

- Population size : 20
- Parent selection : Tournament(k = 3)
- Mutation : Random
- Percentage of gene to mutation : 0.05
- Survivor selection : Elitism
- Termination generation : 20000

In addition, the input image size we tested is 100×100 as shown in the Figure 8.

Figure 8: Test image

After changing the representation by splitting the 3D image into three 2D images, we can find out that the outcome we achieved looks more similar to the testing image than the work from GARI as shown in the Figure 9. The left image is reproduced by GARI, and the right image is reproduced by our algorithm.

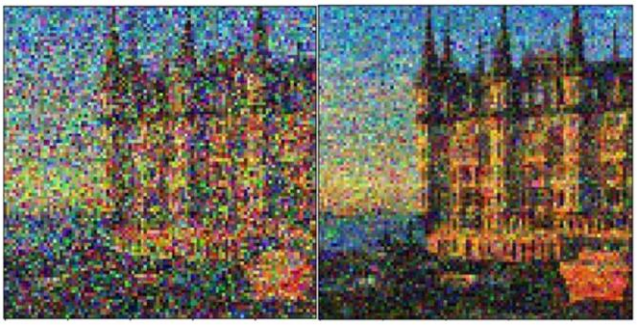

Figure 9: Comparison with GARI

Then there are the results of different crossover method. Single arithmetic crossover will generate the image that has fitness value around 9000 per channel. However, with different α, the style of image is different as shown in the Figure 10. The left image α is 0.5, and the right image α is 0.9. A small α brings smoother image but worse brightness.

Figure 10: Result of single arithmetic crossover

Linear crossover gives slightly better fitness value. It gives a smoother image than the former method. Meanwhile, it also gives distinguishable brightness as shown in Figure 11.

Figure 11: Result of linear crossover

Blend method has the best fitness value in all of crossover methods. In addition, similar to single arithmetic crossover, it gives different styles with different α. A small α also tends to give a image that has less noise but worse brightness as shown in the Figure 12. The left image α is 0.2, and the right image α is 0.5.

Figure 12: Result of blend crossover

Overall, after changing the chromosome, we get better performance than GARI throughout every crossover method. Between all the crossover methods, applying blend crossover can result in the best performance, but not so distinguishable. The fitness value comparison is shown in the Figure 13.

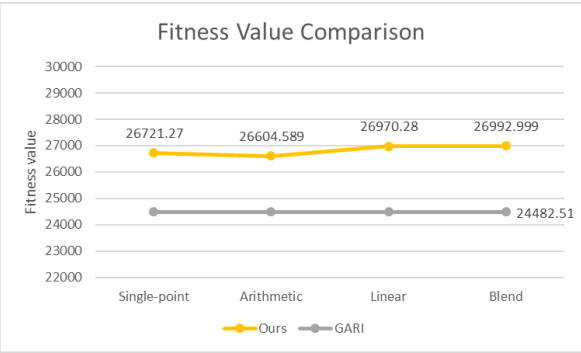

Figure 13: Fitness value comparison

4.2 Execution time

Since the method we used runs three individual genetic algorithms to produce an image, which can cause longer execution time if we run the genetic algorithms sequentially. However, after applying parallel programming to run the three genetic algorithms at the same time, the execution time is more than twice as fast, which is almost as fast as GARI. The execution time comparison is shown in the Figure 14

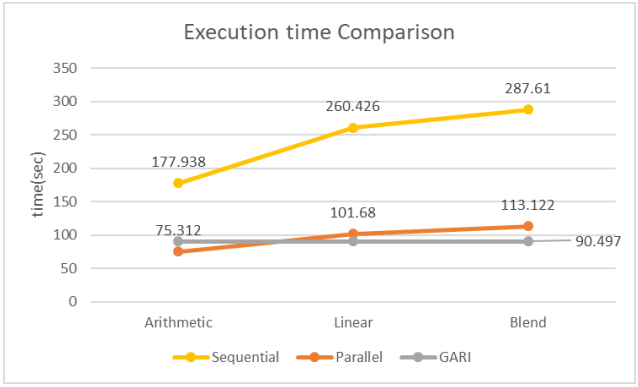

Figure 14: Execution time comparison

5. CONCLUSION

The final goal of this project is to develop and study the performance of an improved image reproducing technique that is based on the genetic algorithm. In this approach, image reproduction is considered as an optimization problem and it is solved using genetic algorithms. We modify the approach given by the other group to make it converge better and speed up the program. As we present above, the outcome is dramatically optimized after changing the data representation (chromosome). With different crossover methods, we compare their performance with different parameters. Also, we tried different parent selection methods and mutation parameters, which didn't give a better outcome so we excluded it from our experiment results.

REFERENCES

[1] https://github.com/ahmedfgad/GARI
[2] https://www.irjet.net/archives/V8/i4/IRJET-V8I4182.pdf