

IMAGE COMPRESSION USING GENE EXPRESSION PROGRAMMING

Robert Gempeler

rrgemp@cc.usu.edu

ABSTRACT

Gene expression programming (GEP) has a variety of applications for solving problems. The purpose of our study was to test whether it was possible and efficient to utilize GEP for compressing digital images. A few grey-scale images were employed, segmented, and put through our GEP software, both in the frequency and spatial domains. Our results were minimal at best, but with more time and research, GEP could effectively be applied to image compression and theoretically compete with JPEG standards.

1. INTRODUCTION

The following introduction will give a short overview of GEP to describe how it differs from genetic algorithms and gene programming. Also, the JPEG image compression design will be discussed because many of the techniques employed by JPEG are used in the project.

1.1. An Overview of GEP

Gene expression programming is a learning algorithm that can be used for a variety of situations to solve otherwise complex problems. GEP originated in 1999 by Ferreira and is based on the ideas of genetic algorithms (GA) and genetic programming (GP). GEP is similar to its predecessors in the sense that it uses populations of individuals, calculates the fitness of each individual, and applies various genetic operations, i.e. mutation and crossover, to introduce genetic variation. The one main difference between GEP and GAs and GP is that GEP stores the individuals as symbolic strings which are then presented as an expression tree for evaluation [1, 2]. The advantage of using GEP is that the chromosomes are relatively small and easy to manipulate. Also, expression trees are easily evaluated by performing a level-order traversal of the tree.

The chromosome of a GEP is expressed as a linear, symbolic string of some fixed size, but can produce expression trees of various sizes due to the structure of the chromosome. A chromosome is composed of one or more genes and the genes consist of two parts, a head and a tail. The head is of a predetermined length and contains symbols for both functions and terminals. The tail can only contain terminals and

its length is given as a function of the head length:

$$t = h(n - 1) + 1 \quad (1)$$

For multigenic chromosomes, a linking function must be chosen beforehand to link the different genes into a more complex and complete expression tree.

Choosing an appropriate fitness function to evaluate the fitness of each individual is critical to the success of any program that utilizes a GEP system. The purpose of the fitness function is to find the most optimal solution out of all the possibilities and carry that individual over into the next generation, a.k.a, selection and replication. There are a number of ways to select individuals according to the fitness function. This study utilized two such schemes, tournament selection and elitism, which will be discussed later.

At the heart of GEP are the genetic operators that introduce genetic variation into the population and ensure that the population migrates towards the optimal values. GEP makes efficient use of a number of genetic operators including mutation, crossover, and transposition. Without the use of genetic variation, the GEP could become “stuck” at a certain point.

The GEP process consists of the following steps: 1) Generate an initial random population, 2) evaluate fitness according to fitness function, 3) check stopping condition, usually until a predetermined fitness is reached or a set number of generations (iterations) are performed, 4) if stopping condition is not met, return to step 2. The GEP process continues in this fashion until the stopping condition is met.

1.2. JPEG Image Compression

Image compression is the process of compressing the data resembling a digital image by eliminating unimportant or redundant segments of an image for more efficient storage or transmission. There are two types of image compression, lossless and lossy. Lossless compression maintains the information of the entire image, whereas lossy compression loses some of the image fidelity, however the portions that are removed have minimal impact on the visual quality of the image.

Our project attempted to utilize the reduced bit rate advantage gained from lossy compression in much the same way JPEG does, but with an intermediate step where we employ GEP. JPEG is based upon the use of the discrete cosine trans-

form (DCT) for its image compression scheme. The DCT transforms an image’s pixel-values from the spatial domain to the frequency domain. It is best suited for smaller image blocks, typically an 8 x 8, because of the introduction of rounding-error of the floating-point values created by using larger blocks. Figure 1 shows an example of the values of the elements in a 2-D matrix before and after the DCT is applied.

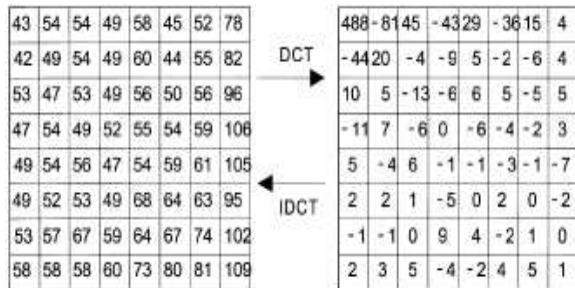


Fig. 1. Example of a matrix in the spatial domain being transformed by the DCT into the frequency domain. Notice the large DC value in the upper left corner of the transformed values. The DC and neighboring values contain the useful information for the image.

Values are then quantized to discard the less useful data found in the lower right regions and linearized using a quantization table and a zig-zig linearization approach which starts with the first element in the upper left-hand corner and initially moves right.

Using genetic algorithms for image compression is not a new idea. Research has been done where support vector learning was employed to achieve compression rates and quality which greatly surpassed JPEG [3]. Another study utilized Genetic algorithms to narrow the search space for fractal image compression [4]. However, there have been no significant breakthroughs in implementing a GEP system for compressing image data. This is the purpose of our study.

2. THE PROPOSED GEP SYSTEM

We initially began the development of the GEP program in MATLAB because of the ease of using the image processing toolbox which is supplied therein. However, we found it troublesome to implement other methods in MATLAB which could be more efficiently performed in a different language. Thus, Java was the language of choice because it also has an image processing toolbox of sorts and it made it much more simple to write different GEP methods, like evaluation of the expression trees for example.

Figure 2 shows a basic approach to the GEP system. The program that was developed for this study reads in a 256 x 256 grey-scale image and segments the image into 8 x 8 blocks.

For each of these blocks, a vector of hash-tables is employed mapping indexes on the interval [1, 64] to the pixel-values given in the 8 x 8 blocks. In the frequency domain, the discrete cosine transform is utilized to narrow the search space by eliminating unnecessary values (lossy compression)[3]. The DCT values are then quantized using the JPEG quantization table and linearized by employing the zig-zag approach starting in the upper left-hand corner. For the spatial domain, the same steps are employed except for the application of the DCT.

A random population of a predetermined size is first generated. The nature of the individuals, i.e. number of genes, head length, function set, etc., is decided beforehand and entered into an object that contains all the parameters needed for a given run of the program. This population then becomes the *parent* population. The *parent* population is ran through a tournament selection procedure which is a selection process that randomly selects two individuals from the population and evaluates their fitnesses. The individual with the higher fitness becomes the “winner” and is placed into a new population. Individuals can be chosen zero, one, or two times in tournament selection meaning that the new population will contain zero, one, or two of each of the individuals from the *parent* population. This new population becomes the *children* population and is the same size as the *parent* population. The fitness of an individual is calculated by using the given fitness function. In this case, a root mean-squared error (RMSE) function was applied to find the error, E_i , between the expected value and the value produced by the GEP.

The *children* population is the population that may become altered by the genetic operations that could be performed on it. The genetic operations that we utilize are: mutation, crossover, and transposition. For a more complete explanation of these operations see [1]. These operations are not necessarily performed on the population. It is dependent on the probabilities that are chosen beforehand, P_x . For this study, the probabilities chosen came from Ferreira’s research. A random number n_x is then generated within the interval [0, 1] and if $n_x > P_x$, then that genetic operation is performed on that individual.

Elitism is then applied by comparing the first individual in the *parent* population with the first individual in the *children* population, the one with the best fitness being carried over into the next generation. This continues for each individual in both populations. The now “genetically richer” population becomes the new *parent* population. The fitness is evaluated and if the fitness is below a predetermined threshold fitness, then the process starts over with this population ready to go through the tournament selection again. Besides a fitness threshold as a stopping condition, we also used a maximum number of generations and timed experiments.

After the GEP is produced, it is evaluated for each pixel-value in the image. Since we are still in the frequency domain, it is necessary to take the inverse DCT to get back to the spa-

tial domain. First we need to delinearize and unquantize our GEP produced results and then take the inverse DCT of those values. The resulting values are the final pixel-values used for the image.

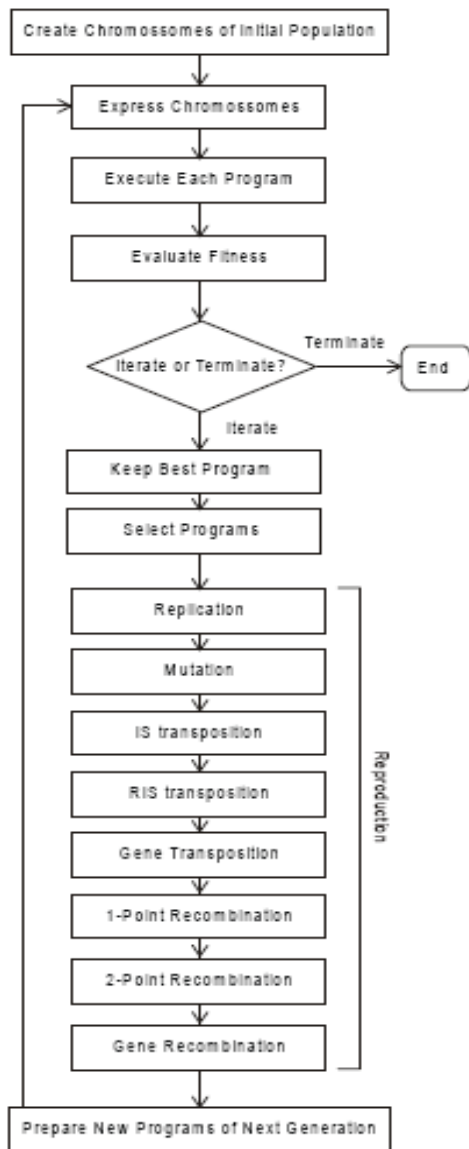


Fig. 2. Flowchart of GEP process

3. EXPERIMENTAL RESULTS

We have tested our system in both the frequency and the spatial domains on numerous 256 x 256 grey-scale images. The GEP system that was developed seems to be working properly, and when given enough time, validly approximates the expected values and generates a function for those values. The GEP-produced image quality seems to be better in the frequency domain versus the spatial domain. One reason for

this could be that the values from the spatial domain could be too complex for the GEP to efficiently approximate, thus causing the process to plateau at a certain point.

The frequency domain could be better because after the DCT is applied to the original values, linearized, and quantized, a number of zeros are left over, and the remaining values seem to be smaller than the originals, except for the DC value, making it easier for GEP to approximate. To make things even less complex, a difference string is computed for each non-zero DCT value which subtracts the previous value from the next value, keeping the first value the same to be able to reproduce the string when GEP is finished.

As far as actual compression is concerned, the encoding scheme that was devised uses a bit-string to store the necessary information to rebuild the image. From our encoding scheme, we reached a compression ratio of 1.58 to 1 on a 256 x 256 grey-scale image using 8 x 8 blocks. And obviously the compression ratio goes up when bigger block sizes are used due to the fact that less information needs to be encoded.

Due to time constraints, all of our tests were run until a maximum number of generations was reached, typically a couple hundred. Figures 3 and 4 are plots of the GEP produced values versus the expected values in the both the frequency and spatial domains, respectively. As one can observe, the GEP produced values follow the curve given by the actual values, but never reach 100% accuracy of the expected values in our tests.

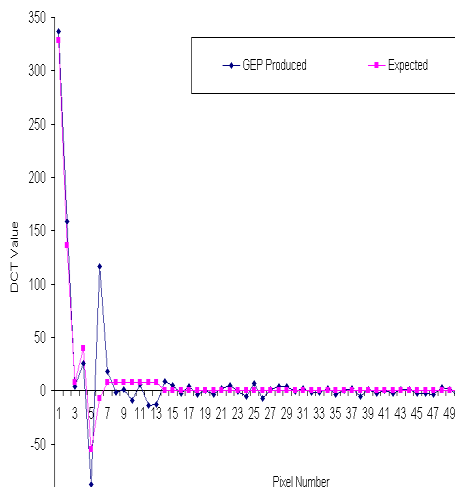


Fig. 3. Frequency Domain; GEP Produced Values versus Expected Values of an 8 x 8 block of *Lena*

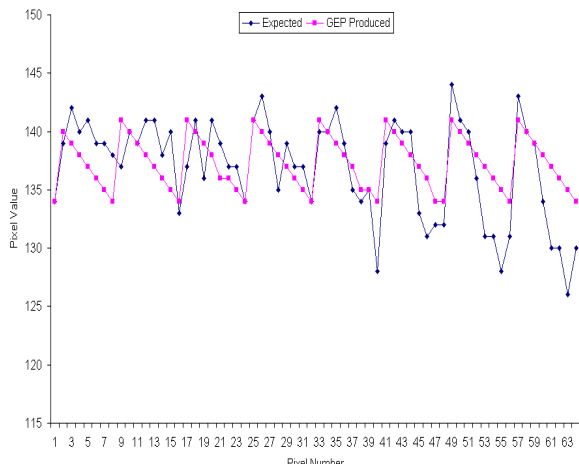


Fig. 4. Spatial Domain; GEP Produced Values versus Expected Values of an 8 x 8 block of *Lena*

Figures 5 - 7 are images of the original *Lena* image, the GEP produced image from the frequency domain, and the GEP produced image from the spatial domain, respectively. The resulting images were produced after 200 iterations, and it is easy to observe the quality of the image is dramatically worse than the original image.



Fig. 5. Original *Lena* image



Fig. 6. GEP produced *Lena* image in the frequency domain after 200 iterations



Fig. 7. GEP produced *Lena* image in the spatial domain after 200 iterations

4. CONCLUSION AND DISCUSSION

Due to the lack of compression, poor image quality, and the time it takes to run one test, this technique does not seem to be a practical compression scheme. More tests could be run at length to allow the GEP to fully achieve near 100% accuracy, however the encoding scheme that was chosen still only provides for miniscual compression compared to the size of the raw image. Future studies could focus on a more efficient encoding scheme that uses less bits than we were using at present.

A distributed system was also attempted to evolve numerous populations at one time and then select the best individuals from those populations and create a “genetically richer” population of individuals to evolve. This task was unfortunately unaccomplished due to time constraints, but future work could employ such a system to implement a more efficient and complete process.

5. REFERENCES

- [1] C. Ferreira, “Gene expression programming in problem solving,” *WSC6 tutorial*, 2001
- [2] C. Ferreira, “Gene expression programming: a new adaptive algorithm for solving problems,” *Complex Systems*, 2001
- [3] J. Robinson, V. Kecman, “Combining support vector machine learning with the discrete cosine transform in image compression,” *IEEE Trans. Neural Networks*, vol. 14, no. 4, pp. 950-958, July 2003
- [4] S. K. Mitra, C. A. Murthy, M. K. Kundu, “Technique for fractal image compression using genetic algorithm,” *IEEE Trans. Image Processing*, vol. 7, no. 4, pp. 586-593, April 1998