Optimization of Fractal Image Compression Based on Genetic Algorithms

Faraoun Kamel Mohamed
Evolutionary Engineering and Distributed Information Systems Laboratory, EEDIS
Computer Science Department
University of Sidi Bel-
Tel/Fax: 213 4857 77 50
Kamel_mh@yahoo.fr

BOUKELIF Aoued
Communication Networks, Architecture and Multimedia Laboratory
Electronics Department
University of S.B.A.
213 72 13 10 21
aboukelif@yahoo.fr

Abstract

The fractal image compression problem put forward three major requirements: speeding up the compression algorithm, improving image quality or increasing compression ratio. Major variants of the standard algorithm were proposed to speed up computation time. But most of them lead to a bad image quality, or a lower compression ratio. In this paper we present an implementation based on genetic algorithms. The main goal is to accelerate image compression without significant loss of image quality and with an acceptable compression rate. Results shown in section 3 prove that genetic compression is a good choice.

I. INTRODUCTION

A major challenge of both theoretical and practical interest is the resolution of the inverse problem: finding an IFS whose attractor is a target of two dimensional (2D) shapes [1]. An exact solution can be found in some particular cases, but in general, no exact solution is known.
As the function to be optimized is extremely complex, most of them make some a priori restrictive hypotheses, such as use of affine IFS, with a fixed number of functions.

The major inconvenient of the current fractal compression algorithm, is its high computational demands. To find existing redundancies (called self-similarities in fractal terms), this algorithm must perform many tests and comparisons between different areas of the compressed image [2]. We cannot find easily similar parts in any natural images, so algorithm complexity is very high, which lead to a very slow compression process.

Genetic algorithms are generally used when we want to solve an optimization problem which is multimodal, multidimensional, and have a large search space with different optima. Such problems, does not have deterministic algorithms to get the global optimum, and if exist, the algorithm is an exhaustive search along the solution space, which lead to exponential time and machine resources consuming. With NP-Hard problems, using deterministic search is impossible. The points. Algorithms of this kind are best suited for the problems described above, and their use to solve different complexes problem has prove their capacities. Both exploitation of best solution and exploration of the entire search space are assured, and an appropriate optimal solution can be found in reasonable number of iterations.
The genetic algorithms are principally destined to complex problems, were no exact solution exist, and an exhaustive brows of the related search space lead to an NP-Hard problem, or high computation time. Our goal is to accelerate the compression process, by improving the standard compression algorithm with a genetic search technique.
This idea was exploited by some authors in different ways, because the optimisation can be viewed from different angles, and be applied on different parameters. Our approach is to use genetic algorithm to optimise the search of similarities in the target image, the standard optimisation methods are sufficient for the calculation of related parameters when the similarity is detected.

II. GENETIC ALGORITHM FOR IFS INVERSE PROBLEM

Genetic algorithms work with a population of individuals which are iteratively adapted towards the optimum by means of a random process of selection, recombination and mutation [4]. During this process, a fitness function measures the quality of the population, and selection favours those individuals of higher quality. Most of the evolutionary algorithms described in the literature for solving the IFS inverse problem follow the optimization problem. In this case, each individual is an IFS model consisting of a number of transformations and its fitness is given by some convenient measure of similarity between the target image and the IFS attractor [3, 5].
To generate the IFS code of a given image by the use of genetic algorithms, two different approaches of representation can be considered:

1) Consider the whole IFS of the coded image as an individual, and then iterate the genetic algorithm on a

population of IFS, each IFS is constituted by a fix number of transformations (depending on the partition) as genes [6].

2) For each range bloc we associate a population of transformation as individuals, each transformation (individual) is represented by its parameters as genes [7].

Our work is based on the second approach, in the following, the main elements of the used algorithm are presented.

III. GENETIC COMPRESSION SCHEMA

Genetic algorithms are used to improve compression schema, principally to accelerate coding time. For each range domain R_i, the set of all possible domain blocks is genetically browsed until we find an appropriate solution. The GA. search space parameters are the domain block coordinates and the isometric flip. The luminance and contrast (S and O) parameters are computed as done in the standard algorithm. The fractal compression scheme for a single image can be seen as in the following algorithm.

1. P ← Generate (LIFS) randomly
2. For all LIFS $p_i \in P$ evaluate by applying (p_i) to generate an image and measuring its distance (using the L^1 or L^2 metric) to the original image;
3. While termination criteria not met;
4. Do reproduce $p_i \in P$ according to evaluation;
5. Apply the desired mutation operator to some $p_i \in P$, selected in some way, creating new LIFS;
6. Apply the desired mating operator to some $p_i , p_j \in P$ selected in some way, creating new LIFS;
7. Evaluate new LIFS (as above);
8. Replace the worst old strings with the best new strings.

Figure 1. The genetic fractal compression algorithm

A. Chromosomes codification

A chromosome in our algorithm is constituted by 5 genes, from which only 3 genes are submitted to genetic modification, the two others are computed by the RMS equation. We have the genes :

1) X_{dom} , Y_{dom}, flip : which are optimised by genetic search;

2) Contrast O, and luminance S: which are computed form the RMS equation.

This will improve both compression speed and reconstruction quality, the following figure show our chromosomal representation of the IFS:

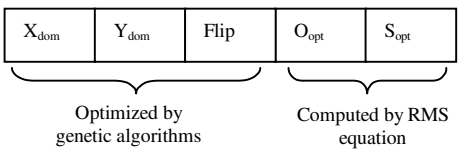

Figure 2. The chromosome codification.

B. The Fitness Function

The fitness function assign to each individual in the population numeric values that determine its quality as a potential solution. The fitness denotes the individual ability to survive and to produce offspring. In our case, the fitness is given by the inverse of the RMS error between the coded range block, and the domain block determined by the transformation coordinates X_{dom} and Y_{dom}, and transformed with corresponding luminance and contrast values.

$$RMS=\frac{1}{n}\left[\sum_{i=1}^{n^2} b_i^2+S(S.\sum_{i=1}^{n^2} a_i^2-2\sum_{i=1}^{n^2} a_i b_i+2.o.\sum_{i=1}^{n^2} a_i)+o.(on^2-2\sum_{i=1}^{n^2} b_i)\right]$$

$$s=\frac{n^2.\left(\sum_{i=1}^{n^2} a_i b_i\right)-\left(\sum_{i=1}^{n^2} a_i\right)\left(\sum_{i=1}^{n^2} b_i\right)}{n^2\sum_{i=1}^{n^2} a_i^2-\left(\sum_{i=1}^{n^2} a_i\right)^2} \qquad o=\frac{1}{n^2}.\left[\sum_{i=1}^{n^2} b_i - S\sum_{i=1}^{n^2} a_i\right]$$

Fitness function (T)= 100 / (RMS(R_i ,T(R_i)))

C. Genetic Operators

The two principal operators used in our implementation are: crossover operator, and mutation operator. Their patterns and structure is presented in the following.

1) Crossover operator: The crossover operator combines two individuals in the current population, to produce two offspring individuals included in the new generation. According to our chromosome representation, the crossover operator compute result coordinates for the offspering individuals by using a linear combinaison of the parents coordinates.

For the first offspring :

$$X_{dom} = a^* X_{dom}^{p1} + (1-a)^* X_{dom}^{p2}$$
$$Y_{dom} = a^* Y_{dom}^{p1} + (1-a)^* Y_{dom}^{p2}$$

For the second offspring:

$$X_{dom} = (1-a)^* X_{dom}^{p1} + (1-a)^* X_{dom}^{p2}$$
$$Y_{dom} = (1-a)^* Y_{dom}^{p1} + (1-a)^* Y_{dom}^{p2}$$

is a random real number in [0,1]. The figure 2 present the schema used by the crossover operator.

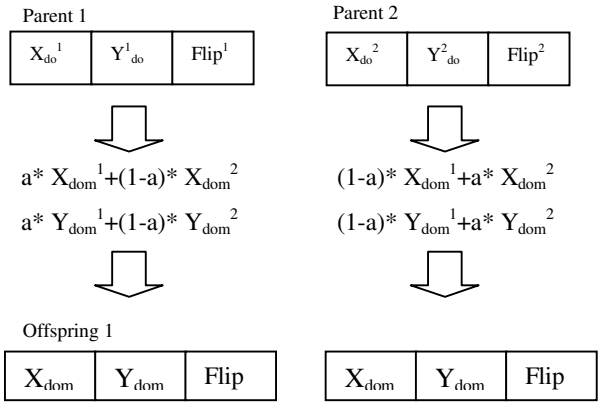

Figure 3. The Crossover operator pattern

2) Mutation Operator: Mutation operator modifies the chromosome genes randomly according to the mutation probability. The genes X_{dom}, Y_{dom} and flip are changed with a random generated value respectively in [0, L], [0, W], and [0, 7] intervals (L and W are the target image dimensions).Figure 4 illustrates the mutation operator schema.

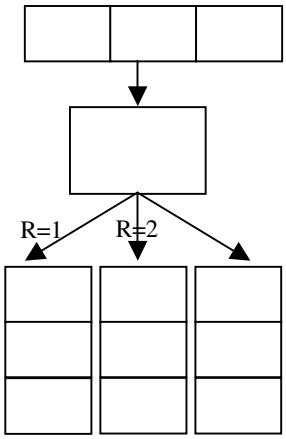

Figure 3. Mutation operator schema

3) Selection Process: To avoid the premature convergence effect, linear scaling is applied to each individual fitness. Then, the Roulette wheel method is used as a selection process.

4) Termination criteria: Any genetic algorithm must find the optimal solution for a given problem in a finite number of steps. In our implementation, two criteria can cause the termination of the algorithm when applied to a given range block:

1) An acceptable value of fitness for the best in individual in the population is reached;

2) A maximum predefined count of generations is reached.

This maximum count is a predefined parameter of the algorithm; it was determined experimentally and fixed to 20 generation in our implementation.

5) The parameters of the algorithm: The behaviour of the genetic algorithm can be controlled using many initial conditions and parameters .We can control convergence speed, solutions quality and algorithm evolution when adjusting and modifying these parameters. In our algorithm, we have two different sets of parameters: the genetic evolution parameters given by:

- Population size;
- Crossover rate;
- Mutation rate;
- Number of generations.

And the fractal compression pattern parameters given by:

- The lowest block size used for ranges decomposition (in the case of QuadTree schema);
- The number of flips and isometrics applied to each domain block;
- The decomposition error limit, this parameter is introduced to improve the QuadTree decomposition schema;
- The RMS error limit fixed to decide if a given transformation is accepted.

The number of bits used to quantify and code luminance and contrast parameters, fixed experimentally to 5 and 7 bits respectively.

In table 1, the set of optimal values of all the algorithm parameters is given. These values ensure compromise between execution time and solutions optimality.

TABLE 1. OPTIMAL PARAMETERS OF OUR GENETIC COMPRESSION ALGORITHM

Population Size	100
Maximum generations	20
Crossover rate	From 0.7 to 0.8
Mutation rate	0.1
RMS limit	5.0
Decomposition error limit	10.0
Flips and isometrics count	8

IV. SIMULATION AND RESULTS

All presented results were obtained on a PIII-INTEL 800MHz with 128Mo of RAM size.

A. Genetic Compression Algorithm with Regular Partition

The decomposition schema is a regular partition with 8x8 and 4x4 block size. The genetic algorithm optimises the domain block search. Results are as follow:

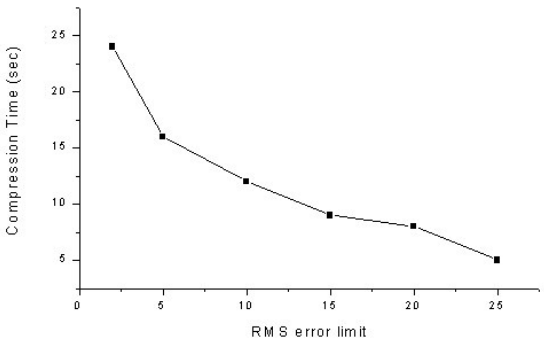

Figure 4. Lenna image Compression ratio variation for different RMS error

Figure 5. Decompressed Barb image, compressed with 8x8 genetic algorithm

B. Genetic algorithms with Quadtree decomposition:

The genetic compression algorithm was used with Quadtree partitioning. Different parameters were used for each test, and the obtained results are given in both table forms and graphical forms. Examples of reconstructed images are also given to illustrate reconstruction quality.

Figure 6. Decompressed Lena image using QuadTree decomposition with RMS=5.0 (ratio 9.14:1)

TABLE 2. DIFFERENT COMPRESSION RESULTS OF LENA IMAGE WHILE APPLYING DIFFERENT VALUES OF RMS ERROR LIMIT

RMS Limit	Execution Time	Quality (dB)	Compression Ratio	Ranges count
0.0	2 m 44 s	35.66	4.29 :1	4069 blocks
2.0	1 m 56 s	35.03	6.35 :1	2770 blocks
4.0	49 sec	34.89	9.28 :1	2023 block
5.0	43 sec	34.80	9.82: 1	1792 blocks
8.0	36 sec	34.50	9.95 :1	1768 blocks
10.0	33 sec	30.50	10.05 :1	1750 blocks
15.0	21 sec	22.33	13.66 :1	1288 blocks
20.0	14 sec	19.36	19.34 :1	910 blocks
25.0	15 sec	19.01	26.25 :1	670 blocks

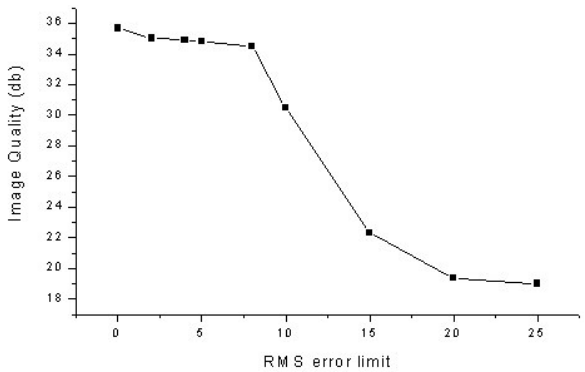

Figure 7. Lena image quality variation according to RMS limit values

TABLE 3. DIFFERENT COMPRESSION RESULTS OF LENA IMAGE WHILE APPLYING DIFFERENT VALUES OF POPULATION SIZE

Population size	Execution Time	Quality (dB)	Compression Ratio	Ranges count
5	9 sec	29.62	8.30 :1	2119
10	11 sec	29.98	8.35 :1	2107
20	14 sec	30.21	8.72 :1	2017
50	23 sec	32.11	9.36 :1	1879
100	44 sec	32.23	9.83 :1	1789
250	2 m 24 sec	33.74	10.35 :1	1699
500	7 m 4 sec	34.56	10.83 :1	1624
1000	23 m 4 sec	35.12	10.97 :1	1603

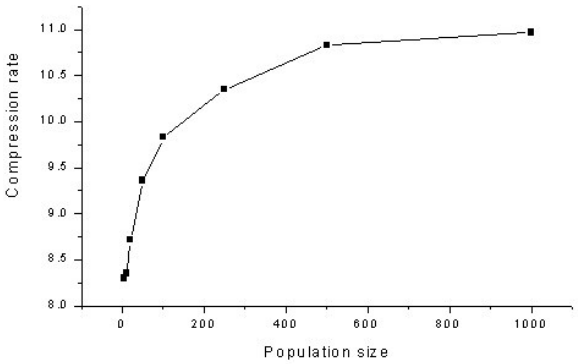

Figure 8. Lena image compression ratio variation according to population size values

V. CONCLUSION

It is clear that the best image quality is always obtained using the standard schema, but its computation time makes it unpractical. So we must accept less quality in favour of quick compression. Our main goal was to accelerate standard compression schema, without greatly decreasing both image quality and compression ratio. Further more this work demonstrates the genetic algorithm ability to solve complex problems.

VI. FUTURE WORKS: DECOMPOSITION WITH EVOLUTIONARY COMPUTATION

Here, for a fixed size square block partition a fractal code is required as in standard fractal coding, but for each range the best D codebook entries are kept in a list together with the optimal scaling and offsets parameters. We take N times this configuration as the starting population for the evolution. The offspring are built by randomly merging two neighbouring blocks. The fractal code is modified by only considering the transformations kept in the lists of those two blocks. A selection is performed by only keeping the fittest configurations in terms of collage error.

The main improvement introduced our approach lies in the stochastic search (random mutations), and not in the crossover schema. Results on the effects of crossover rate and mutation rate may provide some insight on that point.

REFERENCES

[1] M.F. Barnsley and S. Demko. Iterated function systems and the global construction of fractals. In Proceedings of the Royal Society of London A399, pages 243 275, 1985.

[2] Barnsley, M., Hurd, L., Fractal Image Compression, AK Peters, Wellesley, 1993.

Ruhl, M., Evolutionary fractal image compression, in: Proc. ICIP-96 IEEE

[4] D.E. Goldberg. Genetic Algorithms in Search, Optimization & Machine Learning. Addison- wesley, Reading, MA, 1989.

[5] M.F. Barnsley and A.D. Sloan. A better way to compress images. Byte Magazine, pages 215 223, January 1988.

[6] D.E. Hoskins and J. Vagners. Image compression using Iterated Function Systems and Evolutionary Programming: Image compression without image metrics. In Proceedings of the 26th Asilomar Conference on Signals, Systems and Computers, 1992

[7] L. Vences and I. Rudomin. Fractal compression of single images and image sequences using genetic algorithms. Manuscript, Institute of Technology, University of Monterrey, 1994.