Research on heterogeneous distributed fractal image compression algorithm based on parallel genetic algorithm and iterative function

Han Jian Ning

School of Computer Science, Xi’an Shi You University, Xi’an, Shannxi, 710065, P, R, China
jnhan@126.com

Abstract. This paper briefly introduces the method and implementation of image compression of heterogeneous distributed genetic and iterative function system (IFS). Based on the traditional parallel genetic algorithm, a heterogeneous distributed model of parallel fractal image compression based on genetic algorithm is implemented, and the corresponding complex model design scheme is put forward. The results show that the algorithm has better search ability, its comprehensive performance such as signal-to-noise ratio/compression time is further improved, the decoding image is similar to the original image, and has high image quality effect.

1. Introduction

Image data compression is actually a kind of optimization processing problem, the purpose of which is to give the shortest description of these images or data under the condition of satisfying certain quality constraints. Fractal image compression is one of the most promising compression technologies at present. However, the fractal structure has infinite complexity, and with the expansion of sampling space, the amount of computation required to find an effective potential solution is very large, in order to solve the improvement of the speed and efficiency of the algorithm, parallelization is a better way to solve this problem.

The inherent parallelism of genetic algorithm is especially suitable for large-scale parallel processing, and the realization scheme of parallel genetic algorithm can be divided into three categories:

(1) Global scheme. The scheme is time-consuming in the evaluation of adaptive value and is effective far beyond the communication time, otherwise the communication time exceeds the calculation time, but it will reduce the speed of the whole evolution.

(2) Independent scheme, also known as coarse-grained model. Genetic algorithm based on coarse granularity model is called Distributed Genetic Algorithm (distributed Genetic ALGORITHM, DGA). The model is not very demanding for parallel system platform, which can be a loosely coupled parallel system, and the parallelism between the main development groups.

(3) Dispersed scheme, also known as fine-grained model, neighborhood model, suitable for connectors, array machines and single-instruction multi-data flow (Single instruction multiple DATA, SIMD) system.

The basic idea of fractal image compression algorithm proposed in this paper is: Given a two-value image, the fractal encoding of the image (an effective iterative function system (iterated function
2. The basic theory of IFS in iterative function system

(1) Iterative function System IFS

Set up a complete metric space \((X, d)\) with \(n\) compression mappings \(T_i: X \rightarrow X\) \((i = 1, 2, \cdots, n)\), whose compression factor is \(c_1, c_2, \cdots, c_n\), respectively) to form an iterative function system, recorded as \(\{X, T_1, T_2, \cdots, T_n\}\), \(c = \max\{c_1, c_2, \cdots, c_n\}\) called IFS Compression factor.

(2) Two-dimensional compression affine transform

The affine transformation \(w: R^2 \rightarrow R^2\) in two-dimensional Euclidean space \(R^2\) is in the following form:

\[
 w(X) = w(x, y) = [a \ b \ c \ d] [x \ y] + [e \ f] = AX + T, \quad a, b, c, d, e, f \in R
\]

known as a two-dimensional affine transformation, wherein \(A = [a \ b \ c \ d]\), \(T = [e \ f]\). When the linear part \(A\) is compressed, the affine transformation is called a two-dimensional compression affine transformation, and the iterative function system composed of the compression affine transformation defined on the two-dimensional space \(I^2 = [0, 1]^2\) can be recorded as IFS \(\{I^2, w_1, w_2, \ldots, w_n\}\). In order to ensure \(w(x, y) \in I^2\), the parameters of two-dimensional compression affine transform need to meet the parameters of two-dimensional compression affine transformation.

3. Fractal image compression algorithm based on genetic algorithm

3.1. Coding

(1) The encoding form of the gene

\(w_{kl} = [a_{kl}, b_{kl}, c_{kl}, d_{kl}, e_{kl}, f_{kl}]\), Where \(w\) is a compressed affine transformation, \(k\) refers to which chromosome the gene belongs to (that is, IFS), \(i\) refers to the gene is the first gene in the chromosome (that is, affine transformation).

(2) Chromosome coding

Chromosome coding is the encoding of individuals in the population. Each chromosome is essentially an IFS. It is clear that chromosomes are made up of a set of genes, so their encoding forms are defined as follows \(F_k = [w_{k1}, w_{k2}, w_{k3}, \ldots, w_{kn}]\).

Where \(F\) represents an IFS (called a chromosome), \(n\) represents the number of compressed affine transformations that are effective in the IFS (chromosome), and \(k\) refers to the chromosome as the \(k\) chromosome in the population. To ensure that the desired potential solution can be found, the 6 parameters of each compression transformation must meet the two-dimensional compression affine transformation parameter conditions.

3.2. Adaptive value Function Design

When evaluating the quality of chromosomes, its adaptive value function should embody three target optimizations of fractal image compression problem: Maximizing similarity measure \(S\), minimizing compression factor \(\beta\) minimization shrinkage factor \(\delta\).

(1) Maximize similar metrics \(S\)

In this system, because both the original image and the reconstructed image are two-value images, the \(S\) measurement method is the simplest and most effective, the specific way is as follows:

\[
 S(B, \bigcup_{i=1}^{n} w_i(B)) = \frac{|B \cap (\bigcup_{i=1}^{n} w_i(B))|}{|B \cup (\bigcup_{i=1}^{n} w_i(B))|}
\]

Description: The molecule represents the number of pixels of 1 when the original image \(B\) intersects with an image reconstructed by an iterative function, and the denominator represents the number of pixels of the original image \(B\) and the iteration of the iterative reconstructed image and then 1.
(2) Minimize compression Factor $\beta$

After the fractal encoding of an image, it can be reconstructed at any scale size, regardless of the size of the original image. In this system, the number $\beta$ of affine transformations in the IFS is used as the compression factor directly, and a penalty function is designed as follows:

$$R_\eta: \mathbb{N} \rightarrow [0, 1] \quad R_\eta(\beta) = \exp\left(-\frac{\beta^2}{4\eta^2}\right)$$

Wherein, $\eta$ is the number of affine transformations expected, and the system selected as 4 $\beta$ is the number of actual chromosome affine transformations. When $\beta > \eta$, a penalty is imposed, and when $\beta < \eta$ is rewarded.

(3) Minimize shrinkage Factor $\delta$

Since an IFS is made up of a set of compressed affine transformations, it is $\delta_i$ for each compression affine transformation in the IFS, $\delta = \max(\delta_i), i = 1, \ldots, \beta$.

$$\frac{1}{\delta^2}(a^2 + b^2 + c^2 + d^2 + ((a^2 + b^2 + c^2 + d^2)^2 - 4(ad - bc)^2)^{1/2})^{1/2}$$

Designing a penalty function $P_\delta: [0, 1] \rightarrow [0, 1] \quad P_\delta(\delta) = (1 - \delta^{10}) \exp\left(-\frac{\delta^2}{4\eta^2}\right)$

(4) Adaptive value function

To sum up, the adaptive value function can be given as follows:

$$F_\sigma(c) = S(B, \bigcup_{i=1}^{\beta} w_i(B))R_\eta(\beta)P_\sigma(\delta) \quad \text{Where c is the IFS absorption prime; B is the original image.}$$

3.3. Selection tactician

(1) Sampling space

In this system, an extensible sampling space is used, the size of the initial sampling space is $POPSIZE = 50$, and each generation thereafter consists of offspring and some parents. The size of the new population is $POPSIZE = POPSIZE + 5$.

(2) Sampling mechanization

In this system, the sampling size is determined to be 5, and after sorting all chromosomes by the size of the adaptive value, 5 of the best elite individuals are selected to enter the new generation directly. And other individuals in the new population will choose the corresponding individual from the parent in the form of a probability distribution, roulette.

(3) Selection probability

In order to strengthen the competition between chromosomes, the system first sorted from large to small according to the adaptive value of chromosomes. That is, fitness $[0][0]$ represents the chromosome with the largest adaptive value, Fitness$[0][2]$ indicates the actual position of the chromosome with the largest adaptive value in the population.

3.4. Designing genetic operators

Genetic algorithms consist of three basic operations: selection, crossover, and mutation. In this section, we will mainly introduce the two genetic operations of crossover and mutation. Considering the particularity of this problem, the genetic operators designed include: IFS hybridization operator and three mutation operators.

3.4.1. IFS hybridization operation

After selecting the selected operation of the two parent chromosomes, first according to the hybridization probability of $pc=0.5$, to determine whether to carry out hybridization operation, when determined to be hybridized operation, two parent chromosomes according to the number of genes contained (the number of compressed affine transformation) randomly selected hybridization points. When the number of genes of the resulting sub-chromosome does not conform to the maximum number of chromosomes, that is, the total number of compressed affine transformations contained in an IFS specified in the program $\text{MAXMAPS} = 5$, the excess genes will be automatically removed. In this paper, the method of removal at the end is used.
3.4.2. Mutation operators
According to IFS theory, three kinds of mutation operators are designed in this system, namely IFS mutation operator, affine mutation operator and self-replication mutation operator. Each dye in the group mutated with the mutation probability $pmu=0.1$, and the chromosome selected for mutation operation randomly selected three kinds of mutation operators with $1/3$ as the probability.

1. IFS mutation operators
The IFS mutation operator is implemented by adding and removing a compression affine transformation at the end of the chromosome. When the number of genes in the chromosome is less than the expected number of $\eta$, a randomly generated compression affine transformation is added at the end. Otherwise, a gene (compression affine transformation) is removed at the end.

2. The affine mutation operator
The affine mutation operator mutates the genes in the chromosome with 0.5 probability, and according to the characteristics of compression affine, five kinds of affine mutation operators are designed, including rotation, scaling, shearing, panning and shaking. The shaking is a parameter value in 6 parameters that change the compression transformation, and the 6 transformation parameters have an equal chance of being changed, each with a $1/6$ shaking probability. Take parameter A as an example, as follows: $a = a + (1 - F_p(c))\epsilon$.

3. Self-replicating mutation operator
The main reason of designing this operator is to further apply the change of gene parameters to the potential solution of chromosome, so that the algorithm is closer to nature. In the experiment of this system, the operator has played a good role and obtained a good potential solution.

3.5. Operator correct
After genetic manipulation of chromosomes in a new generation of groups, the compressibility and constraints of affine transformations in IFS are affected. Therefore, it is necessary to recycle each of the new generation groups, to ensure that the six parameters of the compression affine transformation in each IFS meet their respective constraints, and to correct the unsatisfied individuals.

4. Heterogeneous distributed parallel fractal image compression algorithm
Based on the fractal image compression algorithm implemented above, a physical model of fractal image compression distributed parallel system based on genetic algorithm is proposed as Client/server. Based on some ideas in the artificial ant colony algorithm, the concept of memory is introduced, and in the design of this distributed parallel system, two similar memories (that is, two dynamic linked tables) are designed in the main control machine, that is, the customer class. One is used to store optional server resources (called Optional Resource Memory), first by the topology diagram of the network in that region (which is known in this scenario), and by a certain optimal situation, the task assignment of each server according to the principle of first-in, in-place; Another memory, on the other hand, holds the machine number of the currently running task and is therefore called a task memory. When a computer is assigned a task, it is removed from the resource memory list and inserted into the run chain table. When it completes the task, it returns a completion signal and the corresponding run result, removes it from the shipping row, and hangs the machine number to the end of the resource memory chain. However, there is a potential problem with this operation: sometimes a machine with only a close distance can complete the operation quickly, there is no need to participate in the operation of the distance machine, the use of the above method may allow the farther machine to often participate in the operation, thus wasting time.

5. Experiment results

5.1. The main parameter settings
In this paper, the main parameter settings are as shown in table 1, according to the Serbinski Triangle, based on the better results obtained.
### Table 1. Main parameter empirical value settings

| Parameter | Description                                      | Value               |
|-----------|--------------------------------------------------|---------------------|
| POPSIZE   | Population size                                  | First value=50      |
| best      | Choosing the number of next generation on merit  | 5                   |
| pc        | selection probability                            | 0.5                 |
| pmu       | mutation probability                             | 0.1                 |
| ein()     | Number of standard compression affine transformations | 4                  |

### 5.2. Experimental results of fractal image compression for distributed parallelization

The original image is a Sierpinski triangular binary image of 128X128. Table 2 lists the best genetic solutions obtained after 980 generations with the Sierpinski triangular binary image as the original image. As you can see from table 2, the IFS system is made up of 3 compressed affine transform $w_1, w_2, w_3$, that is, $3 \times 6 = 18$ real numbers to represent the IFS system. A better Sierpinski triangle image can be obtained through the IFS system.

### Table 2. An optimal IFS solution for the Sierpinski triangle obtained

|   | $a/d$       | $b/e$        | $c/f$        |
|---|-------------|--------------|--------------|
| $w_1$ | 0.489605 15714935116 | 0.059443683951589 | 0.094334287004212 |
| $w_2$ | 0.010362212542184845 | 0.5508846344408153 | 0.011771287004967 |
| $w_3$ | 0.51165 106 9809691 | 0.00461643671801612 | 0.00623957840335489 |
| $w_1$ | 0.44823007344658905 | 0.118 887 367 903 179 | 0.0018 86685 40046 | 23 |
| $w_2$ | 0.572 808 265 379 667 2 | 0.371 498 676 055 827 5 | 0.376 037 2617818562 |

The relevant results obtained in the experiment are as follows: Compression factor $\beta = 0.5124535845304027$, Adaptive value $= 0.35554574636551795$, Similarity $= 0.9258011872824389$. The self-replication mutation operator introduced in the genetic operator is very effective, can obtain the optimal potential solution, and greatly reduces the evolutionary algebra. After Parallelization, the running time has also been greatly improved.

### 6. Conclusion

Because of an essential feature of fractal: scale irrelevance, image compression is carried out with fractal theory, and once a better IFS encoding is obtained, it is independent of the size of the original image, which can enlarge the size of the original image 10 times, but their IFS remains unchanged. In other words, the fractal compression encoding of an image does not change with the size of the image, and the image of any size can be reconstructed when the image is decoded.

### References

[1] Xiao Weishu. Graph Theory Algorithms [M]. Beijing: Press of Aeronautical Industry, 2000.

[2] BARNESLEY MF. Fractals Everywhere [ M ]. New York Academic Press, 1992.

[3] JACOUIN AE. Image Coding Based on A Fractal Theory of Iterated. IEEE Transactions Image Proceedings, 1992, 1(1) : 18 - 30.

[4] DASGUPTA D, NINO F. An Evolutionary Algorithm for Fractal Coding of Binary Images [ J ]. IEEE Transactions on Evolutionary Computation, 2000, 4(2) : 172 - 181.

[5] KENNEDY J, EBERHART R. Particile swarm optimization [ A ]. Proceedings, 2010.