Genetic Algorithm using Discrete Cosine Transform for Fractal Image Encode

N. A. Dheringe, B. N. Bansode

Abstract*— A genetic algorithm using discrete cosine transformation is proposed to speedup the fractal encoder. By using discrete cosine coefficients, the optimal Dihedral transformation between the range block and domain block can be found to save a large number of the redundant MSE computations. Moreover, combining the discrete cosine transformation technique with the genetic algorithm, the length of the chromosome is shortened to smooth the landscape of the search space since the optimal Dihedral index was determined. Hence the encode velocity is accelerated further. Experiments show that the encoding speed of the proposed method is 100 times faster than that of the full search method, while the cost is the 1.1dB loss at the retrieved image quality.

***Index Terms*—Discrete Cosine Transform, Fractal Image Encode, Genetic Algorithm.**

I. INTRODUCTION

Fractal image compression was original proposed by Barnsley and first realized by Jacquin in 1990. The underlying premise of fractal image compression is based on the partitioned iteration function system (PIFS) which utilized the self-similarity property in the image to achieve the purpose of compression. To encode an image according to the self-similarity property, each block must find the most similar domain block in a large domain pool. For the conventional full searching method, the encoding process is time consuming since a large amount of computations of similarity measurement are required to find the best match. Therefore, focal aim of fractal image compression is to speed up the encoder. In the past, some classification methods are adopted to reduce the encoding time. But, these methods either the speedup ratio is limited or the fractal encoding algorithm is complex.

Recently, the genetic algorithm is focused gradually. It was developed by John Holland in 1975 over the course of the 1960s and 1970s and finally popularized by one of his students, David Goldberg. Genetic algorithm is a global search technique mimicking the natural selection and natural genetics. GA is capable of solving many large complex problems where other methods have experienced difficulties, especially when the search space has very rough landscape riddled with many local optima. Since natural images have such characteristics, GA is well suited for the search of the best match in fractal image compression [1].

In this paper, a genetic algorithm using discrete cosine transformation is proposed to speedup the fractal encoding speed.

Manuscript received January 15, 2014.

Ms. N. A. Dheringe, PG Student, Electronics Department, AVCOE, Sangamner, Maharashtra, India.

Prof. B. N. Bansode, Assistant Professor, Electronics Department, AVCOE, Sangamner, Maharashtra, India.

First, by using the discrete cosine coefficients, the optimal Dihedral block between of the range block and domain block can be determined in advance. The range block does the similar match only with the Dihedral block. Another seven Dihedral blocks are ignored to save seven eighths redundant MSE computations. Further, combining the discrete cosine coefficients into the GA, the length of the chromosome in GA is shortened to smooth the landscape of the search space since the optimal Dihedral index was determined. Hence the encode velocity is accelerated further. Finally, the proposed method also attempts to compare with the full search method and baseline genetic algorithm to demonstrate the performance of the proposed method.

II. GENETIC ALGORITHM

The term genetic algorithm, almost universally abbreviated nowadays to GA, was first used by John Holland, whose book Adaptation in Natural and Artificial Systems of 1975 was instrumental in creating what is now a flourishing field of research and application that goes much wider than the original GA. Many people now use the term evolutionary computing or evolutionary algorithms (EAs), in order to cover the developments of the last 10 years. However, in the context of metaheuristics, it is probably fair to say that GAs in their original form encapsulate most of what one needs to know. Holland's influence in the development of the topic has been very important, but several other scientists with different backgrounds were also involved in developing similar ideas. The common thread in these ideas was the use of mutation and selection the concepts at the core of the neo-Darwinian theory of evolution. Other students of Holland's had completed theses in this area before, but this was the first to provide a thorough treatment of the GA's capabilities in optimization.

Nevertheless, using GAs for optimization is very popular, and frequently successful in real applications, and to those interested in metaheuristics, it will undoubtedly be the viewpoint that is most useful. Unlike the earlier evolutionary algorithms, which focused on mutation and could be considered as straightforward developments of hill-climbing methods, Holland's GA had an extra ingredient the idea of recombination. It is interesting in this regard to compare some of the ideas being put forward in the 1960s in the field of operational research (OR). A popular technique, which remains at the heart of many of the metaheuristics described in this handbook, was that of neighbourhood search, which has been used to attack a vast range of combinatorial optimization problems. The basic idea is to explore 'neighbours' of an existing solution—these being defined as solutions obtainable by a specified operation on the base solution.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Basic Concept :

Assume we have a discrete search space X and a function

$$f : \chi \mapsto \mathbb{R}.$$

The general problem is to find

$$\min_{x \in \chi} f.$$

Here x is a vector of decision variables, and f is the objective function. We assume here that the problem is one of minimization, but the modifications necessary for a maximization problem are nearly always obvious. Such a problem is commonly called discrete or combinatorial optimization problems (COP).

One of the distinctive features of the GA approach is to allow the separation of the representation of the problem from the actual variables in which it was originally formulated. In line with biological usage of the terms, it has become customary to distinguish the 'genotype' the encoded representation of the variables, from the 'phenotype' the set of variables themselves. That is, the vector x is represented by a string s, of length l, made up of symbols drawn from an alphabet A , using a mapping

$$c : \mathcal{A}^l \mapsto \chi$$

In practice, we may need to use a search space

$$\mathcal{S} \subseteq \mathcal{A}^l$$

to reflect the fact that some strings in the image A' of under c may represent invalid solutions to the original problem. The string length l depends on the dimensions of both x and A and the elements of the string correspond to 'genes', and the values those genes can take to 'alleles'. This is often designated as the genotype–phenotype mapping. Thus the optimization problem becomes one of finding where the function g(s) = f(c(s))

$$\min_{s \in \mathcal{S}} g(\mathbf{s})$$

It is usually desirable that c should be a bijection. (The important property of a bijection is that it has an inverse, i.e., there is a unique vector x for every string s, and a unique string s for every vector x.) In some cases the nature of this mapping itself creates difficulties for a GA in solving optimization problems.

Both Holland's and Goldberg's books claim that representing the variables by binary strings is in some sense 'optimal', and although this idea has been challenged, it is still often convenient from a mathematical standpoint to consider the binary case. Certainly, much of the theoretical work in GAs tends to make this assumption. In applications, many representations are possible some of the alternatives that can be used in particular COPs. The original motivation for the GA approach was a biological analogy.

In the selective breeding of plants or animals, for example, offspring are sought that have certain desirable characteristics that are determined at the genetic level by the way the parents' chromosomes combine. In the case of GAs, a population of strings is used, and these strings are often referred to in the GA literature as chromosomes. The recombination of strings is carried out using simple analogies of genetic crossover and mutation, and the search is guided by the results of evaluating the objective function f for each string in the population. Based on this evaluation, strings that have higher fitness (i.e., represent better solutions) can be identified, and these are given more opportunity to breed. It is also relevant to point out here that fitness is not necessarily to be identified simply with the composition f(c(s)); more generally, fitness is h(f(c(s))) where f: R→ R is a monotonic function.

Perhaps the most fundamental characteristic of genetic algorithms is that their use of populations of many strings. Holland also used mutation, but in his scheme it is generally treated as subordinate to crossover. Thus, in Holland's GA, instead of the search moving from point to point as in NS approaches, the whole set of strings undergoes 'reproduction' in order to generate a new population.

III. SELF SIMILARITY

Subsets of fractals when magnified appear similar or identical to the original fractal and to other subsets. This property is called self-similarity and it makes fractals independent of scale and scaling. Thus there is no characteristic size associated with a fractal. A typical image does not contain the type of self-similarity found in fractals. But, it contains a different sort of self-similarity. The figure 2.2.1 shows regions of Lena that are self-similar at different scales. A portion of her shoulder overlaps a smaller region that is almost identical, and a portion of the reflection of the hat in the mirror is similar to a smaller part of her hat.

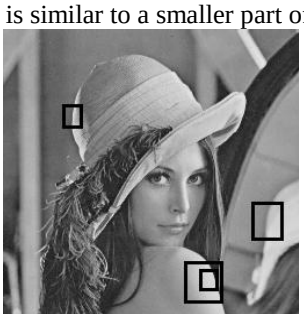

Fig : Self Similarity In Lena Image.

The difference here is that the entire image is not self-similar, but parts of the image are self- similar with properly transformed parts of itself. Most naturally occurring images contain this type of self-similarity. It is this restricted redundancy that fractal image compression chemes attempt to eliminate.

IV. PARTITIONED ITERATED FUNCTION SYSTEM (PIFS)

Fractal image compression uses a special type of IFS called a partitioned iterated function system (PIFS). A PIFS consists of a complete metric space X, a collection of sub domains Di ⟶ X, i = 1,2,.....n, and a collection of contractive mappings Di ⟶X, i = 1,2,.....n.

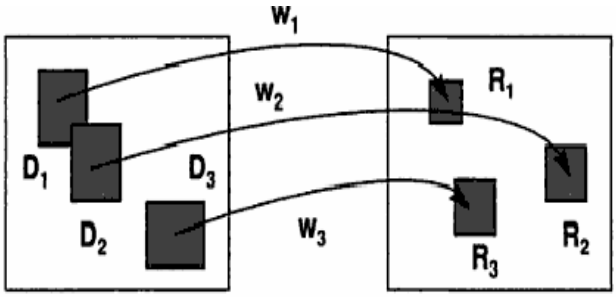

Fig : A Partitioned Iterated Function System.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

V. BASIC FRACTAL IMAGE ENCODING

The fundamental principle of fractal coding consists of the representation of an image by a contractive transform of which the fixed point is close to that image. Banach's fixed point theorem guarantees that, within a complete metric space, the fixed point of such a transform may be recovered by iterated application thereof to an arbitrary initial element of that space. Images are represented within this framework by viewing them as vectors within a Hilbert space, the metric being derived from the inner product via the norm.

Encoding is not as simple, since there is no known algorithm for constructing the transform with the smallest possible distance, given the constraints on the transform, between the corresponding fixed point and the image to be encoded. The usual approach is based on the collage theorem, which provides a bound on the distance between the image to be encoded and the fixed point of a transform, in terms of the distance between the transform of the image and the image itself. A suitable, although suboptimal, transform may therefore be constructed as a "collage" or union of mappings from the image to itself, a sufficiently small "collage error" (the distance between the collage and the image) guaranteeing that the fixed point of that transform is close to the original image.

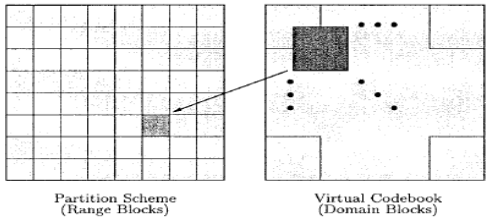

Fig.: One Of The Block Mappings In A PIFS Representation.

In the original approach, devised by Barnsley, this transform was composed of the union of a number of affine mappings on the entire image—an iterated function system (IFS). While a few impressive examples of image modeling were generated by this method, no automated encoding algorithm was found. Fractal compression became a practical reality with the introduction by Jacquin1 of the partitioned IFS (PIFS), which differs from an IF in that each of the individual mappings operates on a subset of the image, rather than the entire image. Since the image support is tiled by "range blocks," each of which is mapped from one of the "domain2 blocks" as depicted in Fig.2.4.1, the combined mappings constitute a transform on the image as a whole. The transform minimizing the collage error within this framework is constructed by individually minimizing the collage error for each range block, which requires locating the domain block which may be made closest to it under an admissible block mapping. This transform is then represented by specifying, for each range block, the identity of the matching domain block together with the block mapping parameters minimizing the collage error for that block. Distances are usually measured by the MSE, equivalent to the distance derived from the inner product, since optimization3 of the standard block mappings is simple under this measure [5].

Let an original image be partitioned into non-overlapping regions called range blocks (R) and overlapping regions called domains blocks (D). The size of each domain block should be larger than that of the range block to satisfy the property of contraction. Let D' denote the down sampled domain block of D and the D' size is equal to the size of R. The transformations are composed of a geometric transformation and a massic transformation. The geometric transformation consists of moving the domain block to the location of the range block and adjusting the size of domain block to match the size of range block. The massic transformation adjusts the intensity and orientation of the pixels in the domain block after it has been operated on by the geometric transformation. The geometric and massic transformation ti can be depicted as follows:

$$t_i\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} A_i & B_i & 0 \\ M_i & N_i & 0 \\ 0 & 0 & s_i \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} E_i \\ F_i \\ o_i \end{bmatrix}$$

where si controls the contrast and oi controls the brightness. z = f (x, y) is the gray level value at (x, y) and Ai, Bi, Mi, Ni can be used to denote the eight-symmetry such as: Identity mapping

- Rotation by 90 degrees
- Rotation by 180 degrees
- Rotation through -90 degrees
- Reflection about mid-vertical axis
- Reflection about mid-horizontal axis
- Reflection about diagonal
- Reflection about cross diagonal.

Ei and Fi are used for position offset. The i in si and oi denotes one of the above mentioned eight symmetries.

In practice, we compare a range block and down sampled domain blocks using MSE metric as follows :

$$MSE = \sum_{k=1}^{n\times n}\left(s_i \times a_k + o_i - b_k\right)^2$$

Where ak represents the pixel value of the sampled domain blocks (D') after eight transformations and bk represents the pixel value of the range blocks and the block size for both R and D' is n by n. This MSE metric allows easy computation for optimal values of si and oi. in equation of ti. This will give us contrast and brightness settings that make the affine transformed ak values have the least squared distance from the bk values. The minimum of MSE occurs when the partial derivatives with respect to si and oi are zero, which occurs when

$$Si = \frac{n\times n\left(\sum_{k=1}^{n\times n} a_k b_k\right) - \left(\sum_{k=1}^{n\times n} a_k\right)\left(\sum_{k=1}^{n\times n} b_k\right)}{n\times n\sum_{k=1}^{n\times n} a_k^2 - \left(\sum_{k=1}^{n\times n} a_k\right)^2}$$

$$Oi = \frac{\sum_{k=1}^{n\times n} b_k - Si\sum_{k=1}^{n\times n} a_k}{n\times n}$$

There are many best matching criteria to choose. The MSE is usually used in fractal image coding and the minimal MSE always denotes better matching. We use equation of MSE to find the optimal si and oi and then quantize them for storage or transmission. In addition, the encoder must record the position of the best matched domain block (D') and its transformation for each range block so as to reconstruct the decoded block on the decoder

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

side.

Suppose the data to be dealt with is 512 × 512 pixel image in which each pixel can be one of the 256 levels of gray(ranging from black to white). Let Ri be the 8x8 pixel non-overlapping range block (i=1,..,4096) and let D be the collection of all the 16×16 overlapped sub-squares of the image. The collection of D contains 497×497=247009 squares when we shift the position of D with one pixel at one step. For each Ri, search through all of collection of Di to find the one which minimizes the MSE as equation of MSE; that is, find the part of the image that most looks like the image above R. There are 8 ways to map one square onto another, so that this means comparing 8×247009=1976072 squares with each of the 4096 range blocks. In addition, we must fulfill the down-sampling operation for each Di to get the same size of R to carry out the later MSE computation. Choosing 1 from each 2×2 sub-square of Di or averaging the 2×2 sub-square corresponding to each pixel of R can achieve the goal of down-sampling. It is obvious that the huge computation is needed from the above descriptions about the conventional fractal encoding. The time to search the best matched domain block for every range block is a time consuming job in practical application. Therefore, we develop a new encoding algorithm to reduce the time in this research. A lot of people have been making efforts in fractal improvement. Some investigate region-based image coding methods and some combine fractal with other algorithm such as wavelet in, genetic algorithms, discrete cosine transform. Saupe and Jacob employ a variance condition to decide whether or not to quadtree partition a block further. C.K. Lee and W.K. Lee use the variance matching technique. The best matched domain block is searched within the searching window for in the neighborhood of the domain block with the closet variance to that of the range block. Some uses a non-symmetric window to search the for the best matched domain block based on the local variance method. Almost all of them used the characteristic of image content to pruning the unnecessary computation to decrease time [2].

VI. FRACTAL IMAGE ENCODE

The fractal image compression is based on the local self similarity property in a nature image. The fundamental idea is coming from the Partitioned Iterated Function System (PIFS). Suppose the original gray level image f is of size 256×256 . Let the range pool R be defined as the set of all no overlapping blocks of size 8×8 of the image f, which makes up (256/8)2 = 1024 blocks. For obeying the Contractive Mapping Fixed-Point Theorem, the domain block must exceed 2 times than the range block in length. Thus, let the domain pool D be defined as the set of all possible blocks of size 16×16 of the image f, which makes up (256-16+1)2 =58081 blocks.

For each range block v from the R, the fractal affine transformation is constructed by searching all of the domain blocks in the D to find the most similar one and the parameters representing the fractal affine transformation will form the fractal compression code of v. To execute the similarity measure between range block and domain block, the size of the domain block must be first sub sampled to 8×8 such that its size is the same as v. Let u denote a sub-sampled domain block. The similarity of two image blocks u and v of size n×n is measured by mean square error (MSE) defined as

$$MSE(u,v)=\frac{1}{n\times n}\sum_{j=0}^{n-1}\sum_{i=0}^{n-1}(u(i,j)-v(i,j))^2$$

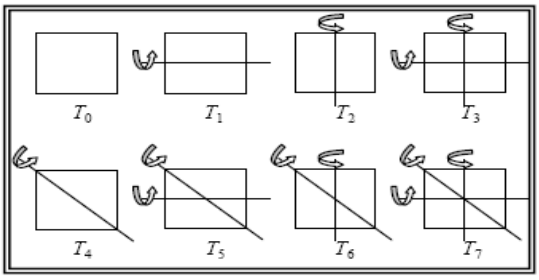

Fig.: The Diagram Of Eight Transformations In The Dihedral Group.

The fractal affine transformation allows the eight transformations of the domain block u in the Dihedral. The eight transformations Tk : k = 0,1, …. 7 can be expressed by the diagrams in Fig. 1. Thus for a given block from the range pool, there are 58081×8 = 464,648 MSE computations must be done in order to obtain the most similar block from the domain pool. Thus, in total, one needs 1024×464,648 = 475,799,552 MSE computations to encode the whole image using this full search compression method. The fractal affine transformation also allows the contrast scaling p and the brightness offset q on the transformed blocks. Thus the similarity is to minimize the quantity d = ‖p .uk + q -v‖ . Here, p and q can be computed directly by,

$$p=\frac{[N<u_k,v>-<u_k,\bar{1}><v,\bar{1}>]}{[N<u_k,u_k>-<u_k,\bar{1}>^2]} \quad \text{and}$$

$$q=\frac{1}{N}[<v,\bar{1}>-p<u_k,\bar{1}>]$$

respectively, where N is the number of pixels of the range block and $\bar{1}=[1 \quad 1 \quad \cdots \quad 1]^T$.

Finally, as u runs over all the 58081 blocks in the domain pool, a set of parameters tx , ty , p, q, and k are obtained and constitute the fractal compression code of v, in which tx and ty represent the position of the domain block. For 256× 256 image, both tx and ty require 8 and 8 bits, respectively. For contrast p, brightness q, and the Dihedral index *k* , 5, 7 and 3 bits are required, respectively. Hence one needs 31 bits in total to encode a range block. Finally, as *v* runs over all 1024 blocks in the range pool, the encoding process is completed[1].

VII. PROPOSED METHOD

In this section, we propose a genetic algorithm using discrete cosine transformation. First, two discrete cosine coefficients: the lowest vertical coefficient F(1, 0) and the lowest horizontal coefficient F(0,1) are used to determine the optimal Dihedral index between the range block and domain block. The technique saves seven eighths MSE computations. Second, the discrete cosine transformation will be combined into the genetic algorithm in order to reduce the MSE computations further. To execute the similar match between the range block and domain block, the lowest vertical coefficient F(1, 0) and the lowest horizontal coefficient F(0,1) will be used to determine the optimal Dihedral block.

The quantity F(m, n) is the DCT of an image block f (i, j) of size N× N defined by

$$F(m,n)=\frac{2}{N}C_mC_n\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}f(i,j)\times \cos(\frac{(2i+1)m\pi}{2N})\cos(\frac{(2j+1)n\pi}{2N})$$

where m, n = 0,1, …. N -1 and

$$C_k=\begin{cases}1/\sqrt{2}, & if\ k=0\\ 1, & otherwise\end{cases}.$$

Typically, for N = 8 , we have

$$F(1,0)=\frac{\sqrt{2}}{8}\sum_{i=0}^{7}\sum_{j=0}^{7}f(i,j)\cos\theta_i \quad \text{and}$$

$$F(0,1)=\frac{\sqrt{2}}{8}\sum_{i=0}^{7}\sum_{j=0}^{7}f(i,j)\cos\theta_j\ ,$$

Where $\theta_i=(2i+1)\pi/16\ ,\ i=0,1,\cdots,7$

The magnitude of F(1, 0) reflects the intensity variation between the left half and right half of the image block f and the magnitude of F(0,1) reflects the intensity variation between the upper half and lower half. Moreover, their signs also show respectively the varied direction of the brightness from shine to dark at horizontal and vertical. If F(1, 0) >0, the brightness at the left half is higher than the one at the right half. Opposite, if F(1, 0) <0, the brightness at the left half is smaller than the one at the right half. Similarly, If F(0,1) >0,the brightness at the upper is higher than the one at the lower. On the contrary, if F(0,1) <0, the brightness at the upper is smaller than the one at the lower. Hence, if the domain block is taken the Dihedral transformation such that both of the magnitude relation and the brightness varied direction of F(1, 0) and F(0,1) of the range block and the transformed block are the same, the MSE is the least and the transformed block is the optimal. The range block does the similar match only with the optimal transformed block. The others are ignored to save seven eighths MSE computations. Once the optimal transformed block of the domain block is determined by the F(1, 0) and F(0,1) , the technique is combined into the genetic algorithm to reduce the MSE computations further. The setup of the proposed method is summarized as follows:

Chromosome Formation :

For traditional GA method, the chromosome is composed of x-coordinate and y-coordinate of an image and the Dihedral index. The complicate landscape reduces the evolutionary velocity of the GA is slower and hard to find the better solution. But for the proposed GA method, the chromosome is only formed by x-coordinate and y-coordinate of an image since the Dihedral index has determined in advance. The chromosome is shortened and the landscape is smoother. Hence the evolutionary velocity for GA can be speedup and the optimal solution can be found.

Fitness Function:

The distance of both range block and sub-sampled domain block is measured by MSE. The fitness value is defined as the reciprocal of MSE

Initial Population :

Chromosomes are initialized randomly. The major questions to consider are firstly the size of the population, and secondly the method by which the individuals are chosen. The size of the population has been approached from several theoretical points of view, although the underlying idea is always of a trade-off between efficiency and effectiveness. Intuitively, it would seem that there should be some 'optimal' value for a given string length, on the grounds that too small a population would not allow sufficient room for exploring the search space effectively, while too large a population would so impair the efficiency of the method that no solution could be expected in a reasonable amount of time.

Unfortunately, from this viewpoint, it appeared that the population size should increase as an exponential function of the string length. In Reeves , the initial principle was adopted that, at the very least, every point in the search space should be reachable from the initial population by crossover only. This requirement can only be satisfied if there is at least one instance of every allele at each locus in the whole population of strings. On the assumption that the initial population is generated by a random sample with replacement (which is a conservative assumption in this context), the probability that at least one allele is present at each locus can be found. For binary strings this is easily seen to be from which we can calculate that, for example, a population of size 17 is enough to ensure that the required probability exceeds 99.9% for strings of length 50. For q-ary alphabets, the calculation is somewhat less straightforward, but expressions are given in that can be converted numerically into graphs for specified confidence levels.

The results of this work suggested that a population size of would be sufficient to cover the search space. Finally, as to how the population is chosen, it is nearly always assumed that initialization should be random. Rees and Koehler, using a model-based approach that draws on the theoretical work of Vose , have demonstrated that sampling without replacement is preferable in the context of very small populations. More generally, it is obvious that randomly chosen points do not necessarily cover the search space uniformly, and there may be advantages in terms of coverage if we use more sophisticated statistical methods, especially for non-binary alphabets. One such simple idea is a generalization of the Latin hypercube which can be illustrated as follows.

Suppose each gene has 5 alleles, labelled We choose the population size to be a multiple of 5, say m, and the alleles in each 'column' are generated as an independent random permutation of which is then taken modulo 5.

Individual	Gene					
1	0	1	3	0	2	4
2	1	4	4	2	3	0
3	0	0	1	2	4	3
4	2	4	0	3	1	4
5	3	3	0	4	4	2
6	4	1	2	4	3	0
7	2	0	1	3	0	1
8	1	3	3	1	2	2
9	4	2	2	1	1	3
10	3	2	4	0	0	1

Fig : An Example Of Latin Hypercube Sampling For L=6 And A=5.

Figure shows an example for a population of size 10. For this figure note that each allele occurs exactly twice for each gene. To obtain search space coverage at this level with simple random initialization would need a much larger population. Another point to mention here is the possibility of 'seeding' the initial population with known good solutions.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Some reports have found that including a high-quality solution, obtained from another heuristic technique, can help a GA find better solutions rather more quickly than it can from a random start. However, there is also the possibility of inducing premature convergence [3].

Selection :

Selection mechanism selects two parents from the mating pool to execute the crossover operation. In our proposed method, the ranking selection is adopted to avoid precocity of population and maintain the good genes to end into the offspring.

The basic idea of selection is that it should be related to fitness, and the original scheme for its implementation is commonly known as the roulette-wheel method. It uses a probability distribution for selection in which the selection probability of a given string is proportional to its fitness.

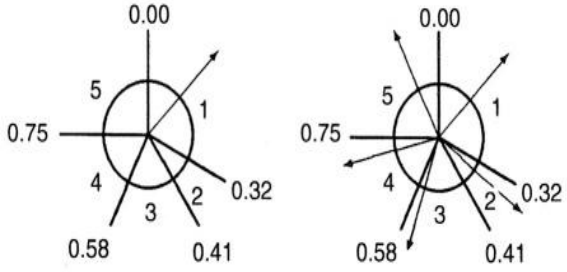

Fig : A Simple Example Of Roulette-Wheel Selection (RWS). For figure suppose that there are 5 strings in a population with fitness{32, 9, 17, 17, 25} respectively. The probability of selection of each individual is proportional to the area of a sector of a roulette wheel(or equivalenatly to the angle subtended at the centre). The no on the spokes of the wheel are the cumulative probabilities for use by a pseudo random no generator. On the left we have standard roulette wheel selection, with a single pointer that has to be spun 5 times. On the right we have SUS, using 5 connected equally spaced pointers; one spin provides 5 selections

Pseudo-random numbers are used one at a time to choose strings for parenthood. For example, in Figure 3.3.2, the number 0.13 would select string 1, the number 0.68 would select string 4. Finding the appropriate number for a given pseudo-random number r requires searching an array for values that bracket r this can be done in time for a population of size M.

However, RWS has a high stochastic variability, and the actual number of times that chromosome C is selected in any generation may be very different from its expected value For this reason, sampling without replacement may be used, to ensure that at least the integral part of is achieved, with fractions being allocated using random sampling. In practice, Baker's stochastic universal selection (SUS) is a particularly effective way of realizing this outcome. Instead of a single choice at each stage, we imagine that the roulette wheel has an equally spaced multi-armed spinner. Spinning the wheel produces simultaneously the values for all the chromosomes in the population. From the viewpoint of statistical sampling theory, this corresponds to systematic sampling. Experimental work by Hancock clearly demonstrates the superiority of this approach, although much published work on applications of GAs still appears to rely on the basic roulette-wheel method.

An associated problem is that of finding a suitable measure of fitness for the members of the population. Simply using the objective function values f (x) is rarely sufficient, because the scale on which f (x) is measured is important. (For example, values of 10 and 20 are much more clearly distinguished than 1010 and 1020.) Further, if the objective is minimization rather than maximization, a transformation is clearly required. Some sort of scaling is thus usually applied, and Goldberg gives a simple algorithm to deal with both minimization and maximization. The method is cumbersome, however, and it needs continual re-scaling as the search progresses. Two alternatives provide more elegant solutions [3].

Crossover :

The crossover operation is used on the two parents to generate the temporarily offspring. In our proposed method, the uniform crossover is used.

Given the stress on recombination in Holland's original work, it might be thought that crossover should always be used, but in fact there is no reason to suppose that it has to be so. Further, while we could follow a strategy of crossover-AND-mutation to generate new offspring, it is also possible to use crossover-OR-mutation. There are many examples of both in the literature. The first strategy initially tries to carry out crossover, then attempts mutation on the off spring (either one or both). It is conceivable that in some cases nothing actually happens at all with this strategy the offspring are simply clones of the parents.

Others always do something, either crossover of mutation, but not both. (Even then, cloning is still possible with crossover if the parents are too alike.) The mechanism for implementing such choices is customarily a randomized rule, whereby the operation is carried out if a pseudo-random uniform deviate exceeds a threshold value. In the case of crossover, this is often called the crossover rate, often denoted by the symbol. For mutation, we have a choice between describing the number of mutations per string, or per bit; bit-wise mutation, at a rate denoted by is more common. In the -OR- case, there is the further possibility of modifying the relative proportions of crossover and mutation as the search progresses.

Davis has argued that different rates are appropriate at different times: high crossover at the start, high mutation as the population converges. He has further suggested that the operator proportions could be adapted online, in accordance with their track record in finding new high-quality chromosomes [3].

Mutation:

The mutation operation is applied to the temporarily offspring to generate the chromosome of the next generation. Its goal is to maintain the diversity of the population and to avoid pre-maturity.

Firstly, we note that in the case when crossover-OR-mutation is used, we must first decide whether any mutation is carried out at all. Assuming that it is, the concept of mutation is even simpler than crossover, and again, this can easily be represented as a bit-string. We generate a mask such as using a Bernoulli distribution at each locus with a small value of p in this case. (The above example would then imply that the 2nd and 6th genes are assigned new allele values.) However, there are different ways of implementing this simple idea that can make a substantial difference to the performance of a GA. The naive idea would be to draw a random number for every gene in the string and compare it to but this

is potentially expensive in terms of computation if the strings are long and the population is large.

An efficient alternative is to draw a random variate from a Poisson distribution with parameter where is the average number of mutations per chromosome. A common value for is 1 in other words, if l is the string length, the (bit-wise) mutation rate is which as early as 1964 was shown to be in some sense an 'optimal' mutation rate. Having decided that there are (say) m mutations, we draw m random numbers (without replacement) uniformly distributed between 1 and l in order to specify the loci where mutation is to take place. In the case of binary strings, mutation simply means complementing he chosen bit(s).

More generally, when there are several possible allele values for each gene, if 0 1 0 0 0 1 we decide to change a particular allele, we must provide some means of deciding what its new value should be. This could be a random choice, but if (as in some cases) there is some ordinal relation between allele values, it may be more sensible to restrict the choice to alleles that are close to the current value, or at least to bias the probability distribution in their favour. It is often suggested that mutation has a somewhat secondary function, that of helping to preserve a reasonable level of population diversity an insurance policy which enables the process to escape from sub-optimal regions of the solution space, but not all authors agree. Proponents of evolutionary programming, for example, consider crossover to be an irrelevance, and mutation plays the major role. Perhaps it is best to say that the balance between crossover and mutation is often a problem-specific one, and definite guidelines are hard to give.

However, several authors have suggested some type of adaptive mutation: for example, Fogarty experimented with different mutation rates at different loci. Reeves varied the mutation probability according to the diversity in the population (measured in terms of the coefficient of variation of fitnesses). More sophisticated procedures are possible, and anecdotal evidence suggests that many authors use some sort of diversity maintenance policy [3].

Stopping Criterion:

When a pre-specified number of iteration is reached, the evolutionary process is stopped.

Connecting the above operational strategies, a GA algorithm incorporates with the discrete cosine transformation is proposed. The detailed steps are given as follows.

Initially, set the population size, the crossover mask, crossover rate, mutation mask, mutation rates.

1. Calculate the F(1, 0) and F(0,1) of all the range blocks and domain blocks.
2. Generate the initial population of chromosomes randomly.
3. Find the optimal Dihedral blocks for all the chromosomes. Calculate the fitness values of these chromosomes based on the style of the optimal transformed block.
4. Rank the chromosomes according to their fitness values.
5. If a pre-specified number of iterations is reached, then stop and record the fractal code. Otherwise, go to next step.
6. Select the parent chromosomes according to the select mechanism.
7. Perform the uniform crossover to generate the temporary offspring.
8. Perform the mutation operation on the temporary offspring to generate the chromosomes of the next generation and go to step 3) [1].

VIII. PERFORMANCE COMPARISON

The images Lena and Pepper are tested to demonstrate the encoding time and retrieved quality of the proposed in comparison to the full searching method and tradition GA method. The parameters of the traditional GA method are the same as the ones of the proposed GA method. The size of the tested images is 256×256. The sizes of the range block and domain block are chosen to be 8×8 and 16×16, respectively. The software simulation is done using BCB on a Pentium 2.0GHz, Windows XP PC. The related GA parameters are set as follows:

- Population size=280
- Crossover rate =0.6
- Mutation rate=0.05
- The number of iteration is 25
- Elitism

Table : The Performance Comparisons Of Full Search, Traditional GA, And Proposed GA Methods.

Image	Method	PSNR(dB)	#MSE	Time(sec)
Lena	Full search	28.91	475,799,552	3141.88
	Traditional GA	27.45	3,560,683	27.94
	Proposed GA	27.81	3,438,296	30.17
Pepper	Full search	29.84	475,799,552	3139.91
	Traditional GA	28.35	3,599,405	28.39
	Proposed GA	28.43	3,447,345	31.02

Table shows some comparative results for full search method, traditional GA method, and proposed GA method, in which the GA parameters of both the proposed GA method and traditional GA method are the same. The table shows that, comparing with the traditional GA method, the encoding time of the proposed GA method is the 7.4%-8.5% slower than that of the traditional GA method. But at the quality of the retrieved image, the improvement is about 0.08dB-0.36dB.

Moreover, in comparison with full search method, the encoding velocity of the proposed GA method is about one hundred times faster than that of the full search method. The cost is about 1.1dB-1.4dB loss at the retrieved image. The quality of the retrieved image is acceptable relatively. The results of retrieved images by proposed GA, traditional GA, and full search methods are shown on Fig. 3.4.1. Fig. 3.4.1(a) is the original Lena image and Fig. 3.4.1(b) is the retrieved image using the full search method. Fig. 3.4.1(c) and Fig. 3.4.1(d) are the retrieved images using the traditional GA method and proposed GA method, respectively[1].

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

(a) (b)

(c) (d)

Fig : (a).Original image, Lena of size 256×256. (b).Full searching method, MSE computations=475,799,552, PSNR=28.91 dB, Time=3141.88 sec. (c).Traditional GA method, MSE computations=3,560,683, PSNR=27.45 dB, Time=27.94 sec. (d).Proposed GA method, MSE computations =3,438,296, PSNR=27.81 dB, Time=30.17 sec.[1]

IX. CONCLUSION

In this, a genetic algorithm using discrete cosine transformation is proposed to speed up the fractal encoder. First, two discrete cosine coefficients are used to determine the optimal Dihedral transformation block between the range block and domain block. Only the optimal transformed block executed the similar match with the range block and the others are ignored to save seven eighths MSE computations.Second, combining the discrete cosine transformation technique with the genetic algorithm, a GA using discrete cosine transformation is proposed to reduce the MSE computations further. Experiments show that comparing with the traditional GA, the encoding speed of the proposed GA method is slower slightly than that of the traditional GA method.But the reward is the improvement of quality of the retrieved image.Compared with the full search method, the encoding time is 100 times faster, while the cost is the 1.1dB decay at the retrieved image quality [1].The performance of the traditional image coding system in terms of speed is greatly improved, which can raise the performance of coding system. The experimental results show that our proposed method makes the encoder much faster than the conventional fractal compression method. Compared to other published methods, the proposed method gets better performance in terms of running time. [2]There are some properties of fractals that make them ideal for other applications. Fractals have no characteristic size, and an encoded image can be decoded at any resolution. Realistic detail is artificially generated at all scales. Fractals perform very well for highly sampled, natural images. Synthetic images and half tones are not compressed by this method. The encoding step is time consuming, but the decoding step is very fast. All these properties make fractals a good choice for transmitting images over the Internet. The resolution independence property is desired, as target browsers have different resolutions. Images are encoded only once, but decoded many times. A sizeable portion of the images over the Internet are natural images. The decoding step works is the natural way that images load up in browsers [8]. A conclusion section is not required. Although a conclusion may review the main points of the paper, do not replicate the abstract as the conclusion. A conclusion might elaborate on the importance of the work or suggest applications and extensions.

REFERENCES

1. Ming-Sheng Wu, “Genetic Algorithm using Discrete Cosine Transformation for Fractal Image Encode ”, IEEE, pp. 309-312, 2012.
2. Yung-Gi, Wu, “Fast Fractal Image Encoder”, International Journal of Information Technology, Vol. 13 No. 1, pp. 15-26, 2007.
3. Colin Reeves, “ Genetic Algorithms”.
4. Meenu Bansal, Sukhjeet K. Ranade, “A Review On Fractal Image Compression”, International Journal of Advances in Computing and Information Technology, pp. 265-276, July 2012.
5. Brendt Wohlberg And Gerhard De Jager, “A Review Of The Fractal Image Coding Literature”, IEEE Transactions On Image Processing, Vol. 8, No. 12, Pp 1716-1729, December 1999.
6. N.A. Koli & M.S.Koli, “A Survey On Fracytal Image Compression Key Issues”, Information technology journal, pp. 1085-1095, 2008.
7. Dr. Fadhil Salman Abed, “ A Proposed Encoding and Hiding Text in an Image by using Fractal Image Compression”, IJCSE, ISSN : 0975-3397 Vol. 4 No. 01 January 2012
8. Viswanath Sankaranarayanan, “Fractal Image Compression Project Report”, 1998.
9. Regina K. Ferrell, Shaun S. Gleason, Kenneth W. Tobin, Jr., “ Application of Fractal Encoding Techniques for Image Segmentation”.
10. S.K. Ghosh J. Mukhopadhyay V.M. Chowdary A. Jeyaram, “Relative Fractal Coding And Its Application In Satellite Image Compression”.