3rd International Conference on Pattern Recognition and Image Analysis (IPRIA 2017) April 19-20, 2017

A Constructive Genetic Algorithm for LBP in Face Recognition

Ali Nazari
ACL Laboratory at Sharif University of Technology
ali.nazari.ir@gmail.com

Saeed Bagheri Shouraki
ACL Laboratory at Sharif University of Technology
bagheri-s@shairf.edu

***Abstract*—LBP coefficients are essential and determine the priority of gray differences. The objectives of this paper are to reveal this and propose a method for finding an optimal priority through the genetic algorithm. On the other hand, the genetic operators such as initialization and cross-over operators, generate invalid coefficients, defective chromosomes. This paper also recommends a rectifying method for correcting defective chromosomes. Results on the FERET and Extended Yale B datasets indicate that the proposed method has markedly higher recognition rates than LBP.**

***Keywords*— face recognition; local binary pattern; genetic algorithm; constructive operator;**

I. Introduction

Face recognition algorithm is identifying people's name based on their face images. This subject is instrumental in people's identification since people are easily photographed without any physical contact in comparison to other biometric methods. Consequently, people experience slight discomfort in presence of surveillance cameras. However, other biometric methods disturb people because they need to contact them physically.

One of the essential extracting methods in face recognition is LBP (Local Binary Pattern). LBP was first proposed to describe image texture [1]. Then, it has been widely employed as a feature extracting method in many applications of image and video processing [2, 3, 4, 5, 6]. Regarding enhancement of LBP, various LBP methods were proposed. Most of these LBP extensions have a rigid structure to extract information from images. Meanwhile, image data can be exploited to enhance LBP functions.

In this paper, LBP is modified to become a better feature extractor for various applications, in particular face recognition through exploitation of application data. Hence, the LBP coefficients are converted to chromosomes, possible solutions in the genetic algorithm. Meanwhile, the genetic operators such as cross-over and mutation operators make new chromosomes invalid. This paper proposes a constructive method to correct these defective chromosomes, particularly it rectifies chromosomes violating the summation constraint, vital to draw histograms.

The paper is organized as follows. In Section II, we delineate LBP. Section III illustrates the proposed method. Section IV defines new constructive components of the genetic algorithm. Section V articulates differences between the proposed method and comparable methods. In Section VI, experiments and evaluation protocols are stated. Ultimately, we conclude the paper in Section VII.

II. LBP

LBP [7] encodes details of each block of an image in the form of differences between gray values of the center pixel with neighboring pixels.

$$LBP_{p,r} = \sum_{i=0}^{p-1} s(g_i - g_c) * 2^i$$
$$s(x) = \begin{cases} 1 & if \;\; x \geq 0 \\ 0 & if \;\; x < 0 \end{cases} \tag{1}$$

Applying the formula "(1)" to an image produces a number for every pixel. It moves a block over each pixel and thresholds other pixels with the central pixel in the block and finally concatenates all these binary values in one bit-stream. One example of using the formula "(1)" is shown in Fig. 1. Then

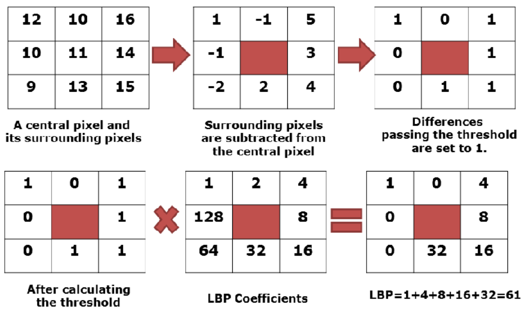
12 10 16 / 10 11 14 / 9 13 15
A central pixel and its surrounding pixels
1 -1 5 / -1 3 / -2 2 4
Surrounding pixels are subtracted from the central pixel
1 0 1 / 0 1 / 0 1 1
Differences passing the threshold are set to 1.
1 0 1 / 0 1 / 0 1 1
After calculating the threshold
1 2 4 / 128 8 / 64 32 16
LBP Coefficients
1 0 4 / 0 8 / 0 32 16
LBP=1+4+8+16+32=61

Fig. 1. An example of LBP for a pixel and its neighborhood.

a histogram of these final values are calculated. A step before histogram, it actually extracts micro-structure information of an image. The final step, e.g. histogram of all values, depicts macro-structure information of an image or summarizes micro- structure information. The entire process from the beginning to the end to create a feature vector is illustrated in Fig. 2.

III. Constructive Genetic Algorithm for LBP

LBP prioritizes gray differences with the powers of two. In other words, these coefficients, i.e. the powers of two, determine the priority of gray differences to each other. The

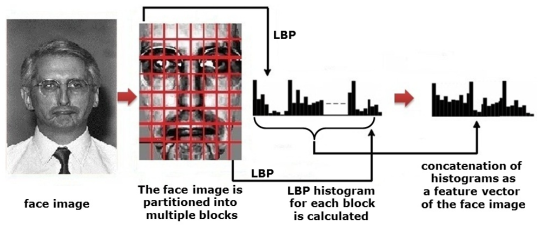
LBP
face image
The face image is partitioned into multiple blocks
LBP
LBP histogram for each block is calculated
concatenation of histograms as a feature vector of the face image

Fig. 2. The entire process of acquiring a feature vector for an image. First the FERET image [13] is preprocessed by the CSU system, then it is divided into blocks. Next, LBP or the constructive GA for LBP is calculated for each block. Ultimately, histograms are concatenated to each other.

LBP coefficients are discrete numbers whereas they can be real numbers in a different order. This paper exhibits an automatic way of acquiring an optimal coefficient set and arrangement for gray differences. These notions are represented in the formula “(2)”.

$$constructiveGA_{p,r} = \sum_{i=0}^{p-1} s(g_i - g_c) * A_i \qquad (2)$$

$$\sum_{i=0}^{p-1} A_i = \sum_{j=0}^{p-1} 2^j = B, \quad 0 \le A_i \le L, \quad A_i \in \Re$$

This formulation “(2)” generalizes LBP too. The B variable is the total number of bins in the histogram and A_i is a real number. Another essential constraint added according to a specific application, here face recognition, is the recognition rate. When the recognition rate is appended to the formula “(2)” as one of the constraints, the equation will be nonlinear with many local optimums. A practical approach to solve this problem is usage of the genetic algorithm.

A typical example of the formula “(2)” illustrated in Fig. 3 uses the coefficients obtained from the proposed method, the constructive genetic algorithm.

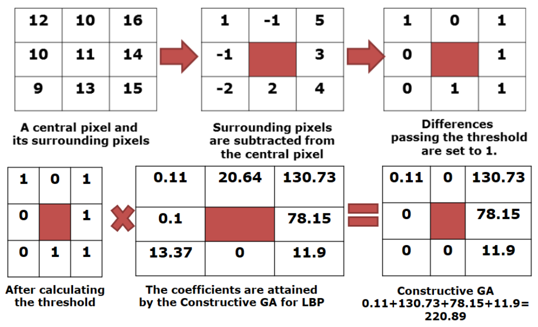
12 10 16 / 10 11 14 / 9 13 15
1 -1 5 / -1 3 / -2 2 4
1 0 1 / 0 1 / 0 1 1
A central pixel and its surrounding pixels
Surrounding pixels are subtracted from the central pixel
Differences passing the threshold are set to 1.
1 0 1 / 0 1 / 0 1 1
0.11 20.64 130.73 / 0.1 78.15 / 13.37 0 11.9
0.11 0 130.73 / 0 78.15 / 0 0 11.9
After calculating the threshold
The coefficients are attained by the Constructive GA for LBP
Constructive GA 0.11+130.73+78.15+11.9= 220.89

Fig. 3. An example of the constructive GA for LBP.

IV. APPLYING THE GENETIC ALGORITHM

The genetic algorithm [8], [9], which resembles biological evolution by the process of natural selection and principle of survival of fittest, is a global search technique. It searches among possible solutions or chromosomes to find the fittest ones. A chromosome is a sequence of bits. Each bit can be a real or discrete number. In every iteration or generation, stronger chromosomes are remained and others will be discarded based on a fitness criterion. Then, some pair candidates are randomly selected for recombination by exchanging some bits between each other. Very rare candidates are randomly chosen for mutation which is a small change in one bit or swapping two bits of a chromosome. Iterations have been repeated until an acceptable solution is acquired.

Nonetheless, there is no strict structure in the genetic method for all problems. Different problems need various definitions of genetic elements. These definitions are crucial and portray how a problem can be solved. In the next section of this paper IV-A, a novel definition of the genetic components is proposed to optimize the formula “(2)” for face recognition.

A. *Chromosome encoding and Initialization*

The first step of the genetic algorithm is to define a chromosome, a feasible solution. Every chromosome is a sequence of coefficients used in the formula “(2)”. If other variables such as B, L, p, and r are needed to be optimized, they can be concatenated at the end of chromosomes. A typical chromosome is shown in Fig. 4. The first chromosome is the powers of two according to LBP.

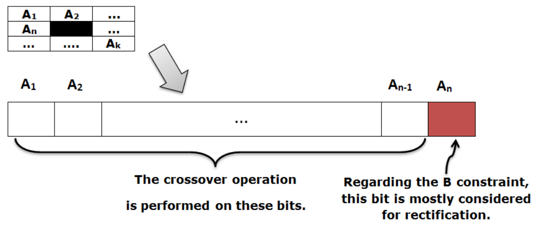
A₁ A₂ ... / Aₙ ... / ... . Aₖ
A₁ A₂ ... Aₙ₋₁ Aₙ
The crossover operation is performed on these bits.
Regarding the B constraint, this bit is mostly considered for rectification.

Fig. 4. A typical representation of a chromosome

Initialization of GA shown in Algorithm 1 also consists of the population size, the number of chromosomes in every generation. As seen in Algorithm 1, each bit of an individual is randomly chosen and assigned by the value, proportional to the largest possible value. These assignments are repeated for entire bits except the last bit.

Algorithm 1 Initialize Operation

```
1:  function INITIALIZE( $population\_size, cl$ )
2:                                         ▷ cl is chromosome length
3:      individuals← CreateZeroVectors (population_size, cl);
4:      for $i \leftarrow 1$ to $population\_size$ do
5:          [$p_1..p_{cl-1}$]=Generate_discrete_Rand_Points(cl-1);
6:                  ▷ This determines the order that bits are randomly
    selected for assignments.
7:          last_coefficient  ← $2^{cl} - 1$;
8:          for $j \leftarrow 1$ to $cl - 1$ do
9:              $individuals[i, p_j] \leftarrow Rand() * last\_coefficient$;
10:             ▷ Rand() generates a random number in the range of
11:                 [0..1]
12:             last_coefficient ← last_coefficient-individuals[i , $p_j$];
13:         end for
14:         RECTIFY(individuals[i,:],cl);
15:     end for
16:     return $individuals$
17: end function
```

Additionally, a significant problem arising in GA is that GA produces invalid chromosomes due to the destructive nature of the genetic algorithm operators such as the initialization and crossover operators.

Regarding producing valid chromosome, the value of the last bit is set after checking whether the chromosome is valid or not. A chromosome is valid if the summation of entire bits becomes B as in the formula "(2)". If it does not satisfy the summation constraint, modifications in some bits are necessary to attain a valid chromosome. The procedure of validation of a chromosome is delineated in the rectification algorithm in Subsection IV-B.

B. *Rectification*

Since chromosome bits are set randomly by GA operators and the summation of bits does not usually satisfy the constraint, it is required to correct chromosomes. Validity of chromosomes is essential owing to garnering the macro-information through the computation of histograms. Invalid chromosomes produce the values greater than histogram bins or much lower. This inconsistency leads to lose information and not to exploit the proper micro-information of some blocks. Thus, it is indispensable to guarantee the validity of chromosomes during the entire genetic process. A chromosome is valid if the summation constraint is not violated. The summation constraint is denoted by the B variable that is a power of the chromosome length minus one ($2^{cl} - 1$) because this value denotes the number of bins.

The detailed description of the rectifying algorithm (2): Sum of all bits except the last bit is calculated. The residue of difference between B and the summation is named as the residual variable. If the residual value is positive, it is assigned to the last bit. If the residual value is negative, it is depicted that the individual is invalid. It is required to alter some bit values. Due to the negative value of the residual variable, the last bit is set to 0 and the other largest number in the bit sequence will be explored as shown in Fig 5. This largest number is the victim of the modification.

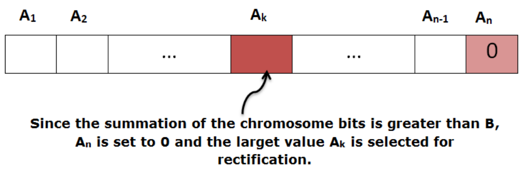
A_1
A_2
A_k
A_{n-1}
A_n
...
...
0
Since the summation of the chromosome bits is greater than B, A_n is set to 0 and the larget value A_k is selected for rectification.

Fig. 5. A typical chromosome that summation of all bits except that the last bit is greater than the B constraint.

If the value of the victim bit is greater than the absolute value of the residual variable, the victim bit is decreased to the amount that makes the individual valid. It also stops the iteration. Otherwise, the victim bit is set to zero and the residual variable is changed to the value that must decrease from another victim in the sequence. This iteration is repeated until the residual value becomes zero.

Algorithm 2 Rectify Operation

```
function RECTIFY($individual, last$)
    correctPrecision(individual)
        ▷ Since numbers are produced randomly, they may be infinitesimal. These decimal fractions are required to be adjusted.
    $residual \leftarrow 2^{last} - 1 - sum(individual[1 : last - 1])$;
    if residual $\geq$0 then
        individual[last] $\leftarrow$ residual;
    else
        individual[last] $\leftarrow$ 0;
        while ( residual !=0 ) do
            [max_value, idx]$\leftarrow$max(individual);
            if max_value $<$ absolute(residual) then
                residual$\leftarrow$ residual+individual[idx];
                individual[ idx ]$\leftarrow$ 0;
            else
                individual[ idx ]$\leftarrow$ individual[ idx ] +residual;
                residual $\leftarrow$ 0;
            end if
        end while
    end if
    return individual
end function
```

C. *Cross-over Operator*

A crossover operator exchanges some bit values between two parental chromosomes to produce two new chromosomes. A single point technique is used to swap bits between chromosomes. The number of the exchanged bits can be randomly selected. In addition, this random point is between the first bit to the one before the last bit.

The last bit is preserved for repair of chromosomes due to destructive nature of the crossover operation. In other words, some destructive alterations take place during the crossover operation on bit values. Thus, it is crucial to check whether chromosomes are still valid. An altered chromosome is repaired at the end of cross-over operation by the rectification procedure described in Subsection IV-B.

The pseudo-code of the crossover operator is depicted in the algorithm 3. The crossover ratio, a real number in the range of [0..1], determines how often two consecutive chromosomes exchange their bits or not.

D. *Mutation Operator*

As the location of every bit in the sequence is influential on account of prioritizing a gray difference to others, the mutation operator can be a shuffle of two bits. When a chromosome is randomly selected for the mutation, two discrete random numbers between 1 to length of chromosome are generated, then these two bits are exchanged. Indeed, the mutation operator changes the order of coefficients in a chromosome. In other words, it changes the priority of bits to each other. The pseudo-code of the mutation operator is shown in the algorithm 4. The mutation ratio, a real number in the range of [0..1], determines how often a chromosome to be shuffled by its bits.

Algorithm 3 Cross Over Operation

```
function CROSSOVER($individuals, crossover\_ratio$)
                    ▷ each row of individuals presents one individual
    no_individuals← FirstDimension(individuals);
    cl ← SecondDimension(individuals);
                                              ▷ Choromosome Length
    if $Odd$ ($no\_individuals$) then
        $no\_individuals \leftarrow no\_individuals - 1$;
    end if
    $counter \leftarrow 1$;
    for $i \leftarrow 1$ to $no\_individuals - 1$ do
        if $Rand() < crossover\_ratio$ then
            point=Random_Point_Generate(cl-1);
                ▷ generates a random discrete number in [1..cl-1]
            newIndividuals[counter,1:point]←individuals[i,1:point];
            newIndividuals[counter, point+1: cl-1]  ←
                                individuals[i+1, point+1: cl-1];
            RECTIFY( newIndividuals[counter,:], cl);

            newIndividuals[counter+1,1:point]←
                                        individuals [i+1,1:point];
            newIndividuals[counter+1, point+1:cl-1]←
                                    individuals[i, point+1:cl-1];
            RECTIFY( newIndividuals[counter+1,:], cl);
            counter ← counter+2;
        end if
    end for
    return $newIndividuals$
end function
```

Algorithm 4 Mutation Operation

```
function MUTATION($individuals, mutation\_ratio$)
            ▷ each row of individuals presents one individual vector
    no_individuals  ← FirstDimension(individuals);
    cl ← SecondDimension(individuals);
                                              ▷ Choromosome Length
    $counter \leftarrow 1$;
    for $i := 1$ to $no\_individuals$ do
        if $Rand() < mutation\_ratio$ then
            [p1, p2]=Random_Two_Points(cl);
                        ▷ generates two discrete numbers in [1..cl]
            newIndividuals[counter,:]←
                        Exchange(individuals[i,:],p1,p2);
            counter ← counter+1;
        end if
    end for
    return $newIndividuals$
end function
```

E. Fitness function

The ultimate purpose of this function is to select the strongest chromosomes indicating optimally feasible solutions from all chromosomes. The optimum solution is identified by a specific application or criterion. Thus, the fitness function evaluates the current and new generations based on the fitness criterion to preserve the fittest chromosomes and discard others.

Since LBP and its variants are employed to extract features to identify face images, the recognition rate is used as the criterion. If another criterion is defined by a different application or another task, that benchmark can be replaced with the recognition rate.

Every chromosome represents the specified coefficients prioritizing gray differences. These coefficients of a chromosome are also used in the formula "(2)" to reduce images or blocks to histograms. A histogram (concatenation of histograms) of a test image is compared to histograms (concatenations) of training images. An identity of a training image is assigned to a test image if the difference between the histogram of a test image and the histogram of a training image is lowest. Then, the number of correct identifications is counted and assigned as an evaluation number of that chromosome.

This evaluation method is applied to all chromosomes. Consequently, all chromosomes, belonging to the old population and a new one produced by the cross-over and mutation operators, are sorted based on their recognition rate. Half of top chromosomes are selected for a half of the new next population. The other half of chromosomes are randomly chosen from the remaining chromosomes that have not been selected yet. Finally, these new chromosomes are shuffled for the preparation of the next iteration.

V. COMPARABLE METHODS

There is a noticeable difference between this paper and other variants of LBP. LBP variants did not alter any part of the LBP formulation or structure. This paper proposes a general method for enhancing the formula of LBP.

Nevertheless, the paper [10] attempted to find effective blocks among entire blocks of a face image. It eliminated some blocks that did not have any impact in the recognition rate.

Further, the article [11] endeavored to optimize the selected feature space among LBP, LGP, and NRLBP. In addition, it made the size of block optimal in the feature space and utilized different distance measures.

In [12], low resolution and noise are the center of attention for facial expression recognition. It attempted to tackle these problems through LBP, weighing the patches, sparse representation, and fisher separation criterion.

VI. EXPERIMENT AND EVALUATION PROTOCOLS

If an experiment has only been performed on unique train and test sets, it will face danger of over-fitting. Thus, train and test sets are selected randomly.

It is also required for identification applications to comprise at least two samples of one identity, a first sample with the known label in the training dataset and a second sample in the test dataset for comparison with the first one to determine falseness or correctness of a found label. Since the goal is to elevate recognition rate, two datasets are needed for specifying coefficients in the learning phase. A first dataset is necessary for their true labels and a second dataset is used to compare with the first data set to determine their labels. These two datasets are engaged in the learning phase, the genetic algorithm, to assess a coefficient set or a chromosome. Other two datasets are necessary in the evaluation phase to report how a final coefficient set works. This evaluation is also used to report. Therefore, databases are divided into four sets: coefficient-train, coefficient-test, profile and test (probe) datasets.

This division is regular when the parameters of a model is unknown [14], [15]. Some parts of data are employed in the learning stage to pinpoint these parameters. First two datasets are similar to the model-data in the papers [14], [15]. In their experiments, they have used different data to find model parameters. Then, they used these parameters in profile and test data sets to report their accuracy.

Eventually, the learning phase of the proposed method exploits two datasets to select a chromosome or a coefficient set. After this specification, two other datasets, i.e. profile and test datasets, are used to report the recognition rate of a selected coefficient set. These profile and test datasets are also utilized to report the recognition rate of LBP.

A. *Stopping criteria*

Since the genetic algorithm is an iterative process, it is required to specify a stopping criterion. A stop can be the definite number of iterations. Another stop can be the particular criterion based on a specific application. Since the goal is to improve the recognition rate, a possible criterion is to stop the genetic algorithm when recognition rate on the probe set is decreasing after some successive increases.

To avoid over-fitting, recognition rates of two phases, namely for coefficient-test and probe sets, are checked after every epoch. If a recognition rate in the coefficient-test set is increasing and decreasing in the probe dataset, over-fitting has occurred on account of the fact that coefficients are tuned to the coefficient-test dataset and are not adjusted to the new dataset (the probe set). This event is summarized in Fig 6. Hence, further iterations do not ameliorate the recognition rate and the learning phase is terminated.

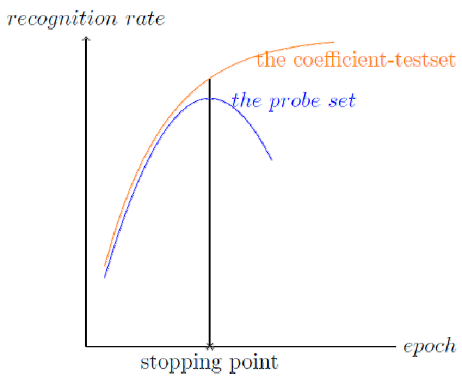

Fig. 6. The stopping point.

B. *Experiments on FERET*

FERET [13] is a common face data-bank used to evaluate face recognition algorithms. It is deserved to evaluate the proposed method using FERET since most papers and algorithms have used FERET to report their recognition rate.

The first step, analogous to [7], is the preprocessing algorithm of CSU Face Identification Evaluation System [16]. The CSU preprocessing algorithm normalizes FERET images according to geometric normalization, histogram equalization and pixel normalization. One example of this preprocessing is illustrated for an image of FERET in Fig. 2.

In the first experiment with SVM classifier, subjects containing at least four images are preserved and others are disregarded. First-images of subjects are employed for the coefficient-train set, second images for the coefficient-test set, third images for the profile set and forth images for the probe set. The result illustrated in Table I is according to Multi- class SVM classifier [17] with the degree of one polynomial kernel. Other parameters are default settings. The crossover and mutation ratios are 0.7 and 0.3 respectively. Every dataset, coefficient-train, coefficient-test, profile or probe set, consists of 240 images.

In the second experiment with the k-NN classifier, if a subject consists of less than four images, its first-image is assigned to the coefficient-train set and the second image to the coefficient-test set. If it comprises more than 4 images, the first-image is attributed to the coefficient-train set, the second to the coefficient-test, the third to the profile dataset and the forth to the probe dataset. If a subject consists of more than four images, the remaining images are attributed to the profile set. The coefficient-train and coefficient-test sets have 1200 images. The profile and probe datasets have 510 and 240 images respectively. Every image is divided into 64 (8 rows x 8 columns) sub-images or blocks. Then, the histogram of each block is acquired by the formula “(2)” with coefficients of a particular chromosome. Then, 64 histograms are concatenated to each other to form one feature vector. The high recognition rates of this experiment are depicted in Table II.

The next experiment is arranged to compare results of the proposed method with LBP results when they both use the same train and probe sets. Datasets are analogous to the second experiment. Images are divided into 16 blocks. Fa images are used in the train dataset and Fb, Fc, DUP1 and DUP2 images are used in the probe datasets such as [7]. Three coefficient sets shown in Table III have higher recognition rates than LBP has. Their corresponding recognition rates of coefficient-sets in Table III are depicted in Table IV.

C. *Experiment on Extended Yale B*

The version of Yale [18], i.e. Extended Yale B [19] comprising 2414 images in 38 subjects is used. Every subject is approximately composed of 63 or 64 images cropped and resized into 32 by 32. Twenty percent of each subject’s images are selected randomly in the learning phase. Half of these data are used in the coefficient-train set and half of the data in the coefficient-test set. Remaining 80% of each subject is selected in the profile and probe sets. Half of them are employed in the profile dataset and half of them in the test dataset. Eventually, the coefficient-train, coefficient-test, profile and probe sets encompass 114, 114, 1095 and 1091 images respectively. Multi-class SVM [17] classifier with a linear kernel is applied to distinguish the identity of each test-image. The result is shown in Table V.

As transparently seen in Table V, the various arrangements of the coefficient-sets have the varying recognition rates. Therefore, it is not only sufficient to encode the circular or square of grey differences to acquire a better result. Additionally, the arrangement is essential for the FERET experiments. Nonetheless, the better coefficient-sets are denoted due to the significance of the better recognition rates.

Table I. THE FIRST ROW DEPICTS THE COEFFICIENT-SET USED IN LBP. IN THE NEXT ROWS ARE COEFFICIENT-SETS ATTAINED BY CONSTRUCTIVE GA FOR FERET WITH THE LIB-SVM CLASSIFIER.

A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	Recognition rate
1	2	4	8	16	32	64	128	22.92
0.69	35.6	3.91	53.29	3.02	145.3	1.35	11.84	25.00
47.63	2.21	66.19	10.33	21.78	18.67	7.71	80.48	24.58
1.68	1.81	5.16	54.41	95.95	26.13	19.76	50.10	24.58
0.69	35.6	39.18	53.29	3.02	48.15	71.16	3.91	24.58
48.64	1.85	15.47	3.17	65.71	109.54	9.24	1.38	24.17

Table II. THE FIRST ROW IS THE COEFFICIENT-SET BELONGING TO LBP. THE REST ARE COEFFICIENT-SETS ATTAINED BY CONSTRUCTIVE GA FOR FERET WITH THE 1-NN CLASSIFIER. IMAGES ARE BLOCKED TO 8X8.

A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	Recognition rate
1	2	4	8	16	32	64	128	70.00
0.11	20.64	130.73	78.15	11.9	0	13.37	0.1	71.25
0	0.1	160.14	27.7	0.59	28.99	36.98	0.5	70.83
13.97	73.19	92.76	26.01	7.75	31.29	8.80	1.23	70.42

Table III. THE FIRST ROW AND THE REST ARE THE COEFFICIENT-SETS RELATED TO LBP AND CONSTRUCTIVE GA RESPECTIVELY FOR FERET WITH THE K–NN CLASSIFIER. IMAGES ARE BLOCKED TO 4X4 SUB-IMAGES.

NO.	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
1	1	2	4	8	16	32	64	128
2	1.51	0.57	2.87	27.18	36.93	185.28	0.26	0.4
3	33.69	21.29	2.20	54.69	65.4	75.24	1.09	1.40
4	24.25	0.3	5.52	38.35	135.58	7.44	42.4	1.16

Table IV. THE RECOGNITION RATES ARE RELATED TO THE COEFFICIENT-SETS MENTIONED IN TABLE III FOR FERET WITH THE K-NN CLASSIFIER.IMAGES ARE BLOCKED TO 4X4.

No	proposed probe-set	fb	fc	dup1	dup2
1	59.58	78.33	12.89	38.92	19.66
2	60.42	78.33	11.86	40.03	23.93
3	60.42	78.41	17.01	38.78	21.37
4	60.00	80.67	13.92	38.37	20.94

VII. CONCLUSION

This paper urges a general approach to obtain optimal coefficients of LBP through the genetic algorithm. It also proposes constructive chromosome generators to produce valid chromosomes.

It checks validity of chromosomes, coefficient-sets, during the entire process of the genetic algorithm. The genetic algorithm generates defective chromosomes, not satisfying the summation constraint. The proposed rectification algorithm scrutinizes new chromosomes whether they are valid. Under invalidity, the rectification algorithm repairs the invalid chromosomes.

One of the advantages of the constructive GA for LBP is to make LBP adaptable to new environments based on a specific application. That is, it employs parts of application data to excel itself.

The results on the FERET and Extended Yale B datasets indicate that the proposed method successfully leads to higher face recognition rates than LBP.

REFERENCES

[1] T. Ojala, M. Pietikainen, and T. Maeenpaa. “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” IEEE Trans. on PAMI, vol. 24, issue 7, pp. 971–987, 2002.

[2] S. Brahnam, L. C. Jain, A. Lumini, and L. Nanni. “Introduction to local binary patterns: new variants and applications,” In Local Binary Patterns: New Variants and Applications, Springer Berlin Heidelberg, pp 1–13, 2014.

[3] L. Nanni, A. Lumini, and S. Brahnam. “Survey on LBP based texture descriptors for image classification,” Int. J. Expert Systems with applications, vol. 39, issue 3, pp 3634–3641, 2012.

[4] D. Huang, C. Shan, M. Ardebilian, and L. Chen. “Facial image analysis based on local binary patterns: A survey,” Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, vol. 41, issue 6, pp 765–781, 2011.

[5] S. Marcel, Y. Rodriguez, and G. Heusch. “On the recent use of local binary patterns for face authentication,” Int. J. Image and Video Processing Special Issue on Facial Image Processing, 2007.

[6] Hadid, J. Ylioinas, M. Bengherabi, M. Ghahramani, and A. Taleb-Ahmed. “Gender and texture classification: A comparative analysis using 13 variants of local binary patterns,” Pattern Recognition Letters, 2015.

[7] T. Ahonen, A. Hadid, and M. Pietikainen. “Face description with local binary patterns: application to face recognition,” IEEE Trans. on PAMI, vol. 28, pp. 2037–2041, 2006.

[8] G. Eiben, and J. Smith. “Introduction to evolutionary computing,” Springer, ISBN: 978-3-540-40184-1, 2008.

TABLE V
THE FIRST ROW AND THE REST REPRESENT LBP AND CONSTRUCTIVE GA COEFFICIENT-SETS FOR EXTENDED YALE B. THE VARIOUS POSITIONS OF THE COEFFICIENTS LEAD TO THE VARYING RECOGNITION RATES.

A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	Recognition rate
1	2	4	8	16	32	64	128	51.15
35.6	0.69	3.91	21.18	2.82	53.29	9.55	127.96	52.52
35.6	0.69	3.91	2.82	53.29	21.18	9.55	127.96	51.79
35.6	0.69	41.11	4.15	7.87	86.01	14.44	65.13	51.97
3.79	0.69	3.91	53.29	2.82	21.18	9.55	159.76	51.79
1.71	3.91	0.69	21.18	53.29	9.55	35.6	129.07	51.33

[9] X. Yu, and M. Gen. “Introduction to evolutionary algorithms,” Springer London Ltd, ISBN: 978-1-447-12569-3, 2012.

[10] J. Shelton, et al. “Genetic based LBP feature extraction and selection for facial recognition,” Proceedings of the 49th Annual Southeast Regional Conference, ACM, pp. 197–200, 2011.

[11] M. Loderer, and J. Pavlovicova. “Optimization of LBP parameters,” In ELMAR (ELMAR), 2014 56th International Symposium, pp. 1–4, IEEE, 2014.

[12] Q. Jia, X. Gao, H. Guo, Z. Luo, and Y. Wang. “Multi-Layer sparse representation for weighted LBP-patches based facial expression recognition,” Sensors 15, no. 3, pp. 6719–6739, 2015.

[13] P. J. Phillips, H. Wechsler, J. Huang, and P. Rauss. “The FERET database and evaluation procedure for face recognition algorithms,” Image and Vision Computing, vol. 16, no. 10, pp. 295–306, Apr 1998.

[14] S.J.D. Prince, J. Warrell, J.H. Elder, and F.M. Felisberti. “Tied factor analysis for face recognition across large pose differences,” IEEE Trans. On PAMI, Vol. 30, pp. 970–984, 2008.

[15] P. Li, Y. Fu, U. Mohammed, J. Elder, and S. J. D. Prince. “Probabilistic models for inference about identity,” IEEE Trans. On PAMI, vol. 34, pp. 144–157, 2012.

[16] D. Bolme, R. Beveridge, M. Teixeira, and B. Draper. “The CSU face identification evaluation system: Its purpose, features and structure,” In ICVS, pp. 304–311, 2003.

[17] C. Chang and C. Lin. “LIBSVM: a library for support vector machines, 2001. Software available at,” http://www.csie.ntu.edu.tw/~cjlin/libsvm

[18] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. “Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection,” IEEE Trans. On PAMI, vol.19, no. 7, pp. 711-720, 1997.

[19] A.S. Georghiades, P.N. Belhumeur, and D.J Kriegman. “From few to many: Illumination cone models for face recognition under variable lighting and pose,” IEEE Trans. On PAMI, vol.23, no.6, pp. 643–660, 2001.