A Novel Evolutionary Approach for Optimizing Content-Based Image Indexing Algorithms

M. Saadatmand-Tarzjan
Electrical Engineering Department, Tarbiat Modares University,
Nasr Bridge P.O. Box 14155-111, Tehran, Iran
saadatmand@kiaeee.org

H. Abrishami Moghaddam*
Electrical Engineering Department, K.N. Toosi University of Technology,
Seyed Khandan P.O. Box 16315-1355, Tehran, Iran
moghadam@saba.kntu.ac.ir

*Corresponding Author

Corresponding Author

Hamid Abrishami Moghaddam (Ph.D.)

Electrical Engineering Department

K. N. Toosi University of Technology

Seyed Khandan

P.O. Box 16315-1355

Tehran, Iran

Tel: +98 21 88461025

Fax: +98 21 88462066

E-mail: moghadam@saba.kntu.ac.ir

A Novel Evolutionary Approach for Optimizing Content-Based Image Indexing Algorithms

Abstract

Optimization of content-based image indexing and retrieval (CBIR) algorithms is a complicated and time-consuming task since each time a parameter of the indexing algorithm is changed, all images in the database should be indexed again. In this paper, a novel evolutionary method called *evolutionary group algorithm* (EGA) is proposed for complicated time-consuming optimization problems such as finding optimal parameters of content-based image indexing algorithms. In the new evolutionary algorithm, the image database is partitioned into several smaller subsets and each subset is used by an *updating process* as training patterns for each chromosome during evolution. This is in contrast to genetic algorithms (GA) which use the whole database as training patterns for evolution. Additionally, for each chromosome, a parameter called *age* is defined that implies the progress of the updating process. Similarly, the genes of the proposed *chromosomes* are divided into two categories: *evolutionary genes* that participate to evolution and *history genes* that save previous states of the updating process. Furthermore, a new *fitness function* is defined which evaluates the fitness of the chromosomes of the current population with different ages in each generation. We used EGA to optimize the quantization thresholds of the wavelet correlogram algorithm for CBIR. The optimal quantization thresholds computed by EGA, improved significantly all the evaluation measures including average precision, average weighted precision, average recall, and average rank for the wavelet correlogram method.

Index Terms

Evolutionary Group Algorithm, Evolutionary Algorithms, Genetic Algorithms, Content-Based Image Indexing and Retrieval, Wavelet Correlogram, Global Optimization.

I. Introduction

Digital image libraries and other multimedia databases have been dramatically expanded in recent years. Storage and retrieval of images in such libraries become a real demand in industrial, medical, and other applications [1]. Content-based image indexing and retrieval (CBIR) is considered as a solution. In such systems, in the *indexing algorithm*, some features are extracted from every picture and stored as an index vector [2]. Then, in the *retrieval algorithm*, every index is compared (using a similarity criterion) to find some similar pictures to the query image [3].

Various indexing algorithms based on different image features such as color [4-5], texture [6], and shape [7] have been developed. Among all these features, color is the most frequently used signature for indexing [8]. Color histogram [4] and its variations [9-11] were the first algorithms introduced in the pixel domain. Despite its efficiency and insensitivity to little changes of the view point, color histogram is unable to carry local spatial information of pixels. Therefore, in such systems, retrieved images may have many inaccuracies, especially in large image databases. For these reasons, two variations called image partitioning and regional color histogram were proposed to improve the effectiveness of such systems [6, 12-13]. Color correlogram is a different pixel domain approach which incorporates spatial information with color [5].

Shape-based indexing algorithms may be divided into two categories. In the first category, features such as edges [7] that reflect the shape of the objects in the image are used [14]. In the second category, each image is initially segmented to several regions (according to a similarity criterion) and then, the features of these regions are used to construct the image index [15].

In the above algorithms, the feature vectors are constructed using spatial domain information. Another possibility is the use of transformed domain data to extract some higher-level features [16]. Wavelet based methods, which provide space-frequency decomposition of the image have been used [17-19]. Daubechies' wavelets are the most frequently used in CBIR for their fast computational and regularity. In [16], Daubechies' wavelets in three scales have been used to obtain the transformed data. Then, histograms of the wavelet coefficients in each sub-band have been computed and stored to construct the feature vector. In SIMPLIcity [17], the image is first classified into different semantic classes using a kind of texture classification algorithm. Then, Daubechies' wavelets are used to extract feature vectors. In a different indexing algorithm, Wang *et al*. [20] used wavelet domain information to index images. Recently, a wavelet-based CBIR system called *wavelet correlogram* has been introduced by Abrishami Moghaddam *et al.* [18]. This system will be briefly reviewed in Section IV-A.

A. CBIR Systems Enhancement

CBIR systems differ from pattern classification methods such as face recognition (FR) and optical character recognition (OCR) since they are more user-dependent in retrieval phase [21]. In other words, CBIR systems should simulate the user idea about the similarity between images. From this view point, these algorithms may be considered as contextual methods whose operation may depend on the user. Therefore, the aim of enhancing the performance of a CBIR system is usually increasing the similarity between the CBIR and user retrieved images from an image database (imagebase), for all query

images. There are generally two ways to enhance the performance of CBIR systems: enhancing the *i)* retrieval and *ii)* indexing algorithms.

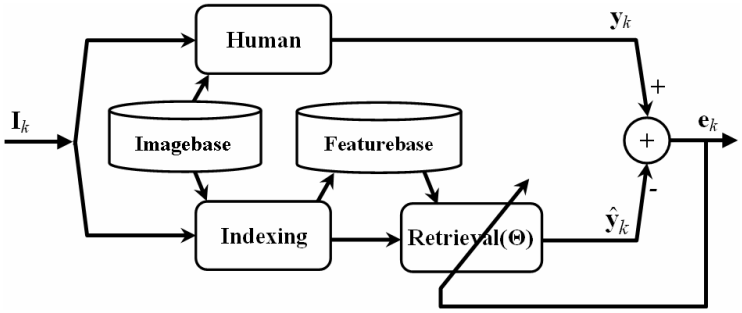

Fig. 1. Block diagram of the retrieval algorithm enhancement process in a CBIR system.

i) Retrieval Algorithm Enhancement: General block diagram of a retrieval algorithm enhancement process is illustrated in Fig. 1. Initially, all images of an imagebase are indexed and their indices are stored in a feature database (featurebase). Then, at each step of the enhancement process, similar images to each query are retrieved using the current retrieval algorithm parameters. Finally, the retrieval algorithm parameters are modified such that the similarity between the CBIR and user retrieved images ($\mathbf{y}_k$ and $\hat{\mathbf{y}}_k$, respectively) for all queries is maximized.

If only one query image ($\mathbf{I}_k = \mathbf{I}$) is used in the enhancement process, the retrieval algorithm parameters will be optimized to evaluate the similarity between that query image and all the images in the database. In this case, the block diagram of Fig. 1 corresponds to *relevance feedback* (RF) approach. RF is a general method to enhance retrieval results using user's feedbacks [21-22]. For example, in the approach presented in [21], the user specifies the matched images among the CBIR-retrieved images for each query. The information provided by the user is used for training a classifier such as support vector

machines (SVM). The classification boundaries obtained by SVM are then used to distinguish between matched and non-matched images.

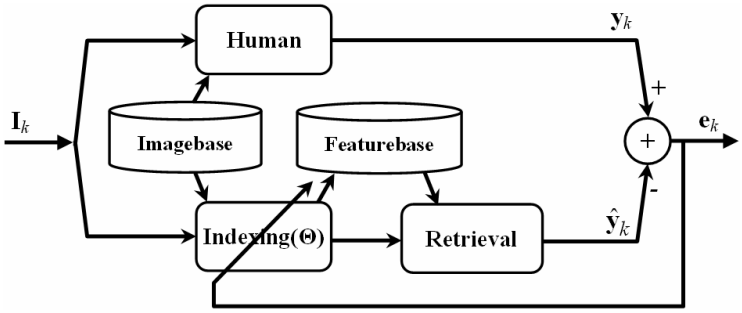

Fig. 2. Block diagram of the indexing algorithm enhancement process in a CBIR system.

ii) Indexing Algorithm Enhancement: Here, enhancing the indexing parameters only for one query image is not meaningful (Fig. 2). Accordingly, the aim of enhancement is global optimization of the indexing algorithm parameters.

Optimization of the indexing algorithm is a more difficult task compared to the retrieval algorithm enhancement. Because each time the indexing algorithm parameters are modified, all images of the reference imagebase should be indexed again. Therefore, the above optimization process has a large computational cost, particularly for large imagebases. Actually, there is no published work focused on optimization of the indexing algorithm parameters.

A number of researchers used a contextual texture classifier to overcome the above drawback [17, 23]. This classifier categorizes images into several contextual categories such as picture, painting, portrait, etc. In the indexing algorithm, considering the attachment category of each image, an appropriate index is constructed. Then, in the retrieval algorithm, this index is compared to the indices of images in the same category.

Although the above classifier may improve the retrieval algorithm performance, it has the following drawbacks: *i)* the contextual classifier should be trained, *ii)* any miss-classification of the query image causes important inaccuracies in the retrieval algorithm results, *iii)* the indexing algorithm parameters for each category should be optimized to provide better performance.

B. Optimization Methods

Optimization methods may be divided into two categories: *i) variational approaches* such as least squares error, recursive least squares error, steepest descent, quasi-Newton approach, and Levenberg-Marquardt method [24-25]; and *ii) global optimizing approaches* such as simulated annealing [26], evolutionary algorithms (EA) [27-28], and ant colony optimization [29].

The variational approaches may converge to a local minimum of a cost function. In such approaches, in contrast to the global optimizers, the cost function usually should be convex and differentiable. Therefore, these approaches can not be used for parameter optimization of the indexing algorithms.

Among global optimizers, evolutionary approaches such as genetic algorithms (GA) [27] seem to be more adapted to the indexing algorithm optimization problem. However, optimizing the indexing algorithm parameters by GA as well as other evolutionary optimizers is not a trivial task, because of their huge computational cost. For example, when GA are used, for evaluation of each chromosome, all images of the reference imagebase should be indexed again (see Section I-A). In other words, if indexing of all

images of the reference imagebase takes 20 minutes long and GA generate only 2000 chromosomes during evolution, its entire computational time will be 667 hours.

Researchers proposed a wide variety of approaches to improve the efficiency of evolutionary algorithms [27-28, 30-36]. Most of these algorithms try to reduce the number of generations, while preserving the optimization performance [30]. For example, Smith [31] proposed an algorithm in which the population size was adjusted as a function of selection error probability. Another example is GAVaPS [32], a genetic algorithm with variable population size. The age concept for a chromosome was introduced for the first time in this algorithm. Srinivas and Patnaik [33] added mutation and crossover probabilities to the bitstring of each chromosome. They adjusted these probabilities using the maximum and mean fitness in the population, and obtained more efficiency. Recently, elitism has been used to speed up the GA performance [34-35]. In elitism-based GA, a number of individuals in the current population survive to the next generation. GENITOR algorithm [36] proposed by Whitley generates only two offspring in each generation and replaces the worst two chromosomes of the population. Recently, Xu *et al* [37] proposed six efficient speed-up strategies to improve the convergence speed of GA.

Obviously, for optimizing the indexing algorithm parameters, GA should obtain an acceptable enhanced solution by generating a limited number of chromosomes during evolution; otherwise the computational cost increases impractically. Among the above fast evolutionary algorithms, the elitism-based GA and GENITOR are more adapted to the above requirements. However, their computational cost is still a major problem.

In this paper, a novel evolutionary method called *evolutionary group algorithm* (EGA) for time-consuming problems such as optimizing the parameters of image indexing

algorithms is proposed. Compared to the conventional GA, EGA has an important advantage: the evolutionary process in EGA is several times faster than GA. EGA uses dynamic chromosomes that can experience during evolution; while in GA, they are unchangeable members of a population. Therefore, EGA corresponds better to the new evolution theory proposed by Williams [38-39].

C. Evolutionary Group Algorithm

In EGA, the imagebase is partitioned into several subsets and each subset is used by an *updating process* as training patterns for each chromosome during evolution. This is in contrast to GA which use the whole database as training patterns for evolution. Additionally, for each chromosome, a parameter called *age* is defined that implies the progress of the updating process. Similarly, the genes of the proposed *chromosomes* are divided into two categories: *evolutionary genes* that participate to evolution and *history genes* that save the previous states of the updating process. Furthermore, in each generation, a new *fitness function* evaluates the fitness of the chromosomes with different ages in the current population.

We applied EGA for optimizing quantization thresholds of the wavelet correlogram CBIR algorithm [18, 40]. The optimal quantization thresholds computed by EGA, improved significantly all the evaluation measures including average precision, average weighted precision, average recall, and average rank for the wavelet correlogram method.

D. Paper Outline

The remainder of the paper is organized as follows: Section II presents the application of GA to optimize the indexing algorithm parameters. The proposed EGA is described in

Section III. In Section IV, EGA is used to optimize the quantization thresholds of the wavelet correlogram CBIR algorithm and the simulation results are given. Finally, we conclude and suggest future research in Section V.

II. CBIR Optimization Using GA

In order to illustrate the difference between the proposed EGA and typical GA, the application of GA to optimize the indexing algorithm parameters is described in this section.

A. Reference Imagebase

As stated in Subsection I-B, the aim of indexing parameters optimization (in a CBIR system) is maximizing the similarity between the CBIR and user retrieved images in a reference imagebase for all query images. In this paper, a subset of COREL database [17] is used as the reference imagebase (**D**). This imagebase consists of 10 image categories ($A = 10$) as listed in Table I, and each category includes 100 images ($C = 100$), i.e. $|\mathbf{D}| = 1000$ where |.| returns the cardinality of a set.

In Fig. 3, an image from each category is shown. For the above reference imagebase, the user-retrieved images ($\mathbf{y}_k$) for the query image $\mathbf{I}_k$ are defined as follows:

$$\mathbf{y}_k = \text{Human}(\mathbf{I}_k) = \left\{ i = 1,2,...,|\mathbf{D}| \,\middle|\, \mathbf{I}_i \in \mathbf{D};\ \Psi(\mathbf{I}_i) = \Psi(\mathbf{I}_k) \right\} \tag{1}$$

$$\Psi(\mathbf{I}_k) \in \{1,2,..., A\}, \qquad k = 1,2,...,|\mathbf{D}| \tag{2}$$

where, $\Psi(\mathbf{I})$ returns the similarity category of image $\mathbf{I}$, and $\mathbf{y}_k$ represents a set of the images $\mathbf{I}_k$ that belong to the same similarity category. Obviously, we will have, $|\mathbf{y}_k| = C$.

Fig. 3. Some images from the reference imagebase (one image per category).

B. Indexing Algorithm Optimization by GA

In a typical GA, to optimize the indexing parameters $\boldsymbol{\Theta}=[\theta_0,\theta_1,...,\theta_{N-1}]$, chromosomes may be defined as follows:

$$\boldsymbol{\Theta}_j=[\theta_{j,0},\theta_{j,1},...,\theta_{j,N-1}],\qquad j=1,2,...,M \tag{3}$$

where N and M are the number of genes (indexing parameters) and population size, respectively. Then, for each chromosome, the query results for all images in the reference imagebase are computed according to the following equations:

$$\hat{\mathbf{y}}_k^j=\mathrm{CBIR}(\mathbf{I}_k;\boldsymbol{\Theta}_j)=\mathrm{Retrieval}(\mathbf{F}_k^j),\qquad k=1,2,...,|\mathbf{D}| \tag{4}$$

$$\mathrm{Retrieval}(\mathbf{F}_k^j)=\left\{\, p=1,2,...,|\mathbf{D}| \;\middle|\; \mathbf{I}_p\in\mathbf{D};\ Rank(\left\|\mathbf{F}_p^j-\mathbf{F}_k^j\right\|^m)<C \right\} \tag{5}$$

$$\mathbf{F}_k^j=\mathrm{Indexing}(\mathbf{I}_k;\boldsymbol{\Theta}_j),\qquad k=1,2,...,|\mathbf{D}| \tag{6}$$

where, $\mathbf{F}_k^j$ and $\hat{\mathbf{y}}_k^j$ represent the feature vector of the query image $\mathbf{I}_k$ and the query results obtained by the indexing parameters of the j-th chromosome, respectively, and $\|\cdot\|^m$ is the Minkowski distance of rank m [41]. Obviously, better chromosomes give more similarity between the CBIR and user-retrieved images. Therefore, the evaluation function of GA may be defined as follows:

$$J(\boldsymbol{\Theta}_j) = \frac{1}{2C \mid \mathbf{D} \mid} \sum_{k=1}^{|\mathbf{D}|} \sum_{p=1}^{C} \sum_{q=1}^{C} \delta(y_{k,p}^{j}, \hat{y}_{k,p}^{j}) \tag{7}$$

$$0 \le J(\boldsymbol{\Theta}_j) \le 1 \tag{8}$$

where, $\mathbf{y}_{k,p}^{j}$ indicates the p-th member from $\mathbf{y}_{k}^{j}$ and $\delta(\cdot)$ is the Kronecker delta function:

$$\delta(a,b) = \begin{cases} 1 & a = b \\ 0 & a \neq b \end{cases} \tag{9}$$

This evaluation function is equivalent to the normalized recall measure [42]. Larger values of J mean more similarity between the CBIR and user-retrieved images. In other words, if the CBIR solutions completely match the user solutions, the evaluation function will result in 1 and if they are totally mismatched it will be 0. From (4)-(7), it is clear that any modification of the indexing algorithm parameters, requires reindexing of all images in the reference imagebase for each chromosome which means a very large computational cost. We propose EGA as a solution to overcome the above drawback.

III. EGA Principles

In EGA, the reference imagebase is partitioned into several subsets and the whole database is used only once. In each evolution step, only one subset is indexed using each chromosome in the population. Therefore, in EGA, the evolution proceeds during indexing the image subsets of the reference imagebase. This will reduce significantly the computational cost of EGA compared to GA.

A. Reference Imagebase Partitioning

All images of the reference imagebase are partitioned to L different subsets according to (10) and (11) such that each subset includes $\eta = C/L$ images from each similarity category:

$$\mathbf{I}_{(i,k)} = \mathbf{I}_{|\mathbf{D}|_L \times i+k}, \quad i = 0,1,\ldots,L-1, \quad k = 1,2,\ldots,|\mathbf{D}|_L \tag{10}$$

$$\mathbf{D}_i = \{\mathbf{I}_{(i,k)} \big| k = 1,2,\ldots,|\mathbf{D}|_L\}, \qquad i = 0,1,\ldots,L-1 \tag{11}$$

$$|\mathbf{D}|_L = |\mathbf{D}|/L \tag{12}$$

where, $\mathbf{I}_{(i,k)}$ indicates the *k*-th image in the subset $\mathbf{D}_i$ and $|\mathbf{D}|_L$ is the total number of images in each subset. The following equations are simply concluded:

$$|\mathbf{D}_i| = |\mathbf{D}|_L = A\eta \tag{13}$$

B. Proposed Chromosomes

In EGA, the chromosomes are defined as follows:

$$\mathbf{\Theta}'_j = \{g_j, \mathbf{\Theta}_j^{\text{evn}}, \mathbf{\Theta}_j^{\text{his}}, \hat{\mathbf{J}}_j\}, \qquad j = 1,2,\ldots,M \tag{14}$$

where, $g_j, \mathbf{\Theta}_j^{\text{evn}}$, $\mathbf{\Theta}_j^{\text{his}}$ and $\hat{\mathbf{J}}_j$ represent the *age gene*, *evolutionary genes*, *history genes*, and *evaluation genes* of the *j*-th chromosome, respectively. Evolutionary genes supply inheritance characteristics like the chromosomes in a simple GA. The age is a new gene which enables the chromosome to treat time during evolution. The history and evaluation genes are complementary to the age since they enable the chromosome to treat changes of the environmental conditions as well as time during evolution. In other words, the proposed chromosome can obtain new experiences during evolution; and logically, it should be evaluated not only based on its evolutionary genes but also based on its experiences.

i) Evolutionary Genes: As implied by their name, the evolutionary genes participate to evolution whose goal is their optimization. Indeed, evolutionary genes are the indexing parameters that should be optimized during evolution:

$$\mathbf{\Theta}_j^{\text{evn}} = [\theta_{j,0}, \theta_{j,1}, \ldots, \theta_{j,N-1}], \qquad j = 1,2,\ldots,M \tag{15}$$

ii) Age Gene: For each chromosome, the age gene indicates the number of the image subsets that have already been indexed during evolution. Using the age gene g_j, we may write:

$$\mathbf{F}^{j}_{(i,k)} = \text{Indexing}(\mathbf{I}_{(i,k)}; \mathbf{\Theta}^{\text{evn}}_{j}), \quad i = 0,1,...,g_j - 1, \;\; k = 1,2,...,|\mathbf{D}|_L, \;\; j = 1,2,...,M \tag{16}$$

where, $\mathbf{F}^{j}_{(i,k)}$ indicates the index vector of the *k*-th image from the *i*-th image subset that is computed by the indexing parameters of the *j*-th chromosome. The age of each chromosome is initially set to zero and obviously, it is always between 0 and *L*.

iii) History Genes: For each chromosome, the history genes are the index vectors of the previously indexed images. Indeed, the history genes develop a local featurebase for each chromosome as follows (Fig. 4):

$$\mathbf{\Theta}^{\text{his}}_{j} = [\mathbf{F}^{j}_{(0,1)},\ldots,\mathbf{F}^{j}_{(0,|\mathbf{D}|_L)},\mathbf{F}^{j}_{(1,1)},\ldots,\mathbf{F}^{j}_{(1,|\mathbf{D}|_L)},\ldots,\mathbf{F}^{j}_{(g_j-1,1)},\ldots,\mathbf{F}^{j}_{(g_j-1,|\mathbf{D}|_L)}], \qquad j = 1,2,...,M \tag{17}$$

During evolution, for each chromosome, the CBIR system retrieves the matched images using the above local featurebase. Therefore, for query image $\mathbf{I}_{(i,k)}$, we have:

$$\hat{\mathbf{y}}^{j}_{(i,k)} = \text{CBIR}(\mathbf{I}_{(i,k)}; \mathbf{\Theta}'_{j}) = \text{Retrieval}(\mathbf{F}^{j}_{(i,k)}, \text{g}_j) \tag{18}$$

$$\text{Retrieval}(\mathbf{F}^{j}_{(i,k)}, g_j) = \left\{ (p,q) \,\middle|\, p = 0,...,g_j - 1, q = 1,...,|\mathbf{D}|_L; Rank(\left\| \mathbf{F}^{j}_{(p,q)} - \mathbf{F}^{j}_{(i,k)} \right\|^{m}) < g_j \times \eta \right\}, \quad i = 0,1,...,g_j - 1,\ k = 1,2,...,|\mathbf{D}|_L,\ j = 1,2,...,M \tag{19}$$

where, $g_j \times \eta$ is the cardinality of each similarity category in the local featurebase (history genes) of the *j*-th chromosome. Similarly, for each chromosome, the user retrieves the matched images in the local imagebase as follows (Fig. 4):

$$\mathbf{y}^{j}_{(i,k)} = \text{Human}(\mathbf{I}_{(i,k)}; g_j) = \left\{ (p,q) \,\middle|\, p = 0,...,g_j - 1, q = 1,\ldots,|\mathbf{D}|_L; \Psi(\mathbf{I}_{(p,q)}) = \Psi(\mathbf{I}_{(i,k)}) \right\}, \quad i = 0,1,...,g_j - 1,\ k = 1,2,...,|\mathbf{D}|_L,\ j = 1,2,...,M \tag{20}$$

The evolutionary genes participate directly to the evolution process, while the history genes provide only the required memory information (index vectors of the previously

indexed images) of the CBIR system. As will be seen in the next subsection, this memory is used to compute the evaluation function for each chromosome. In other words, the evolutionary and history genes provide the necessary information respectively for indexing and retrieval algorithms.

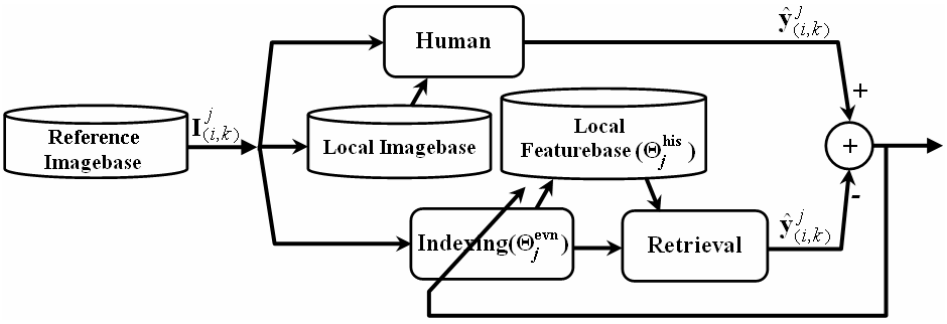

Fig. 4. Block diagram of the indexing and retrieval processes for each chromosome during evolution.

iv) Evaluation Genes: For the *j*-th chromosome, the evaluation function is defined in the same way as in Section II-B:

$$\hat{J}(\Theta'_j, g_j) = \frac{1}{2C\,|\,\mathbf{D}\,|}\left(\frac{L}{g_j}\right)^2 \times \sum_{i=0}^{g_j-1} \sum_{k=1}^{|\mathbf{D}|_L} \sum_{p=1}^{g_j\eta} \sum_{q=1}^{g_j\eta} \delta(\mathbf{y}^j_{(i,k),p}, \hat{\mathbf{y}}^j_{(i,k),q}), \qquad j = 1,2,...,M \tag{21}$$

where, $\mathbf{y}^j_{(i,k),p}$ indicates the *p*-th member of $\mathbf{y}^j_{(i,k)}$. During evolution of each chromosome, the evaluation function values are saved in the evaluation genes defined as follows:

$$\hat{\mathbf{J}}_j = [\hat{J}_{j,g}], \qquad \hat{J}_{j,g} = \hat{J}(\Theta'_j, g), \quad j = 1,2,...,M, \quad g = 0,1,...,g_j - 1 \tag{22}$$

where, $\hat{J}_{j,g}$ indicates the evaluation function value for the *j*-th chromosome at the age *g*.

C. Mature and Immature Chromosomes

Obviously, (21) evaluates the performance of the CBIR system, provided that the cardinalities of the local featurebases and imagebases are sufficiently large. For each

chromosome, the cardinality of the local featurebase is related to its age. Therefore, the evaluation computed by (22) will be valid if the chromosome's age is larger than a threshold λ. We call this chromosome as *mature* and the chromosome whose age is smaller than λ as *immature*.

D. Updating Chromosomes

A chromosome updating process (CUP) is defined in EGA in order to update the genes by indexing the new images of the next image subsets. CUP proceeds as follows:

1. S mature chromosomes, whose ages are smaller than L, are randomly selected from the current population as well as all immature chromosomes to make a set $\mathbf{P}$.
2. The history genes of these chromosomes are extended as follows:

$$\forall j \in \mathbf{P}: \qquad \mathbf{\Theta}_j^{\text{his}} = [\mathbf{\Theta}_j^{\text{his}}, \mathbf{F}^j_{(g_j,1)}, \ldots, \mathbf{F}^j_{(g_j,|\mathbf{D}|_L)}] \tag{23}$$

3. The evaluation genes of there also extended as follows:

$$\forall j \in \mathbf{P}: \qquad \hat{\mathbf{J}}_j = [\hat{\mathbf{J}}_j, \hat{J}(\Theta'_j, g_j + 1)] \tag{24}$$

4. The age genes of these chromosomes are increased by one:

$$\forall j \in \mathbf{P}: \qquad g_j = g_j + 1 \tag{25}$$

Therefore, the age gene indicates the progress of the updating process. In each stage of CUP, all immature chromosomes are kept in the population and are updated until they become mature and generate offspring.

E. Fitness Function

In typical GA, the fitness function is defined as a function of evaluation values for all chromosomes [43]. This simple definition is no more useful in EGA, since the

chromosomes in the population have different ages and are not in the same conditions for being comparable. Generally, comparing the evaluation values of the chromosomes with various ages is not valid for the following reasons:

1. Chromosomes in different ages have local feature and image-bases with different sizes. Consequently, their retrieval results are not comparable.
2. According to (18)-(20), the retrieval results of the chromosomes with various ages have different dimensions.
3. Older chromosomes are more valuable than younger ones, since they have been kept in the group for a longer duration and competed more with other chromosomes during evolution.

In the proposed fitness function, the chromosomes are first classified into same-age classes. Then, the probability of selecting each class is computed. Finally, the probability of selecting each chromosome as parent within each same-age class is computed.

Mathematical Discussion: The same-age class $\mathbf{Q}_j$ containing the chromosomes with age j is defined as follows:

$$\mathbf{Q}_j = \left\{ \mathbf{\Theta}'_k \middle| k = 1,2,...,M; g_k = j \right\}, \quad j = 1,2,\ldots,\beta \tag{26}$$

We also have:

$$\mathbf{Q} = \bigcup_{j=1}^{\beta} \mathbf{Q}_j \tag{27}$$

$$\mathbf{Q}_i \cap \mathbf{Q}_j = \phi \tag{28}$$

where**,** $\mathbf{Q}$ contains all chromosomes and β is the age of the oldest chromosome in the current population. In other words, β represents the number of same-age classes.

Furthermore, a set $\mathbf{\Gamma}_j$ including the chromosomes in the current group with the ages equal to or smaller than j is defined as:

$$\mathbf{\Gamma}_j = \begin{cases} \bigcup_{i=1}^{j} \mathbf{Q}_i, & j = 1,2,\ldots,\beta \\ \phi, & j = 0 \end{cases} \tag{29}$$

Using (28) and (29), the probability of selecting a chromosome from the set $\mathbf{\Gamma}_j$ is computed as:

$$P(\mathbf{\Gamma}_j) = \sum_{i=1}^{j} P(\mathbf{Q}_i), \qquad j = 1,2,\ldots,\beta\,. \tag{30}$$

where $P(\mathbf{Q}_i)$ is the probability of selecting a chromosome from the set $\mathbf{Q}_i$. Then, we have:

$$P(\overline{\mathbf{\Gamma}}_j) = 1 - \sum_{i=1}^{j} P(\mathbf{Q}_i), \qquad j = 1,2,\ldots,\beta \tag{31}$$

where, $\overline{\mathbf{\Gamma}}$ indicates the complement set of $\mathbf{\Gamma}$. Suppose that j is the age of the k-th chromosome. Then, its probability for selection is defined using Bayes law as follows:

$$\mathbf{\Theta}'_k \in \mathbf{Q}_j \Rightarrow P(\mathbf{\Theta}'_k) = P(\mathbf{\Theta}'_k, \mathbf{Q}_j) = P(\mathbf{\Theta}'_k \big| \mathbf{Q}_j) P(\mathbf{Q}_j) = \frac{\hat{J}_{k,j}}{\sum_{\mathbf{\Theta}'_i \in \mathbf{Q}_j} \hat{J}_{i,j}} P(\mathbf{Q}_j) \tag{32}$$

Then,

$$P(\mathbf{Q}_j) = \sum_{\mathbf{\Theta}'_k \in \mathbf{Q}_j} P(\mathbf{\Theta}'_k, \mathbf{Q}_j) \tag{33}$$

Moreover, the conditional probability of selecting the same-age class $\mathbf{Q}_j$, knowing that the selected chromosome's age is equal to or greater than j, is defined as follows:

$$P(\mathbf{Q}_j \big| \overline{\mathbf{\Gamma}}_{j-1}) = \frac{\sum_{\mathbf{\Theta}'_i \in \mathbf{Q}_j} \hat{J}_{i,j}}{\sum_{\mathbf{\Theta}'_i \notin \mathbf{\Gamma}_{j-1}} \hat{J}_{i,j}} \tag{34}$$

The numerator of the above equation is the total evaluation value of the chromosomes with the age j, while its denominator is the total evaluation value of the chromosomes with the age equal to or greater than j. Using (31), (34) and Bayes law we may write:

$$\mathbf{Q}_j \subset \overline{\mathbf{\Gamma}}_{j-1} \;\Rightarrow\; P(\mathbf{Q}_j) = P(\mathbf{Q}_j, \overline{\mathbf{\Gamma}}_{j-1}) = P(\overline{\mathbf{\Gamma}}_{j-1}) P(\mathbf{Q}_j \left| \overline{\mathbf{\Gamma}}_{j-1}\right.) \Rightarrow$$

$$P(\mathbf{Q}_j) = \left(1 - \sum_{i=1}^{j-1} P(\mathbf{Q}_i)\right) \frac{\sum_{\mathbf{\Theta}'_i \in \mathbf{Q}_j} \hat{J}_{i,j}}{\sum_{\mathbf{\Theta}'_i \notin \mathbf{\Gamma}_{j-1}} \hat{J}_{i,j}} \tag{35}$$

Finally, using (32) and (35), we have:

$$P(\mathbf{\Theta}'_k) = \left(1 - \sum_{i=1}^{j-1} P(\mathbf{Q}_i)\right) \frac{\hat{J}_{k,j}}{\sum_{\mathbf{\Theta}'_i \notin \mathbf{\Gamma}_{j-1}} \hat{J}_{i,j}} \tag{36}$$

The last equation computes the fitness (selection probability) of the *k*-th chromosome with age *j*.

Fitness Computation Algorithm: The immature chromosomes can not be used as parents to generate new offspring during evolution for their imprecise evaluations. Therefore, their fitness should always set to zero until they become mature. The proposed algorithm for computing the chromosomes' fitness (the fitness algorithm) is summarized as follows:

1. k is set equal to the threshold of puberty λ. Also, the fitness of all immature chromosomes is set to zero:

$$\forall \mathbf{\Theta}'_j \in \mathbf{Q}_k : \qquad P(\mathbf{\Theta}'_j) = 0, \qquad k = 0,1,\ldots,\lambda - 1. \tag{37}$$

2. The fitness of all chromosomes that belong to the same-age class $\mathbf{Q}_k$ is computed using (36).
3. The total fitness of the same-age class $\mathbf{Q}_k$ is computed by (33).
4. k=k+1,
5. Steps 2-4 are repeated until the fitness of all chromosomes is computed ($k = \beta$).

Example: We give an example to illustrate how the proposed fitness computation algorithm works. Suppose that the current population includes 6 chromosomes with the following evaluation values:

$\hat{\mathbf{J}}_1$: 0.80

$\hat{\mathbf{J}}_2$: 0.50

$\hat{\mathbf{J}}_3$: 0.75 0.70

$\hat{\mathbf{J}}_4$: 0.90 0.65

$\hat{\mathbf{J}}_5$: 0.65 0.60 0.55

$\hat{\mathbf{J}}_6$: 0.75 0.65 0.58

Therefore, the same-age classes are defined as $\mathbf{Q}_1 = \{\boldsymbol{\Theta}'_1, \boldsymbol{\Theta}'_2\}$, $\mathbf{Q}_2 = \{\boldsymbol{\Theta}'_3, \boldsymbol{\Theta}'_4\}$, $\mathbf{Q}_3 = \{\boldsymbol{\Theta}'_5, \boldsymbol{\Theta}'_6\}$. The fitness of the above classes and their chromosomes are computed by the fitness algorithm as follows:

Step 1:

$$\begin{cases} P(\boldsymbol{\Theta}'_1) = \dfrac{0.80}{0.80+0.50+0.75+0.90+0.65+0.75} = 0.1839 \\ P(\boldsymbol{\Theta}'_2) = \dfrac{0.50}{0.80+0.50+0.75+0.90+0.65+0.75} = 0.1149 \\ P(\mathbf{Q}_1) = P(\boldsymbol{\Theta}'_1) + P(\boldsymbol{\Theta}'_2) = 0.2988 \end{cases}$$

Step 2:

$$\begin{cases} P(\boldsymbol{\Theta}'_3) = (1-0.2988) \times \dfrac{0.70}{0.70+0.65+0.60+0.65} = 0.1888 \\ P(\boldsymbol{\Theta}'_4) = (1-0.2988) \times \dfrac{0.65}{0.70+0.65+0.60+0.65} = 0.1735 \\ P(\mathbf{Q}_2) = P(\boldsymbol{\Theta}'_3) + P(\boldsymbol{\Theta}'_4) = 0.3623 \end{cases}$$

Step 3:

$$\begin{cases} P(\boldsymbol{\Theta}'_5) = (1-0.2988-0.3623) \times \dfrac{0.55}{0.55+0.58} = 0.1650 \\ P(\boldsymbol{\Theta}'_6) = (1-0.2988-0.3623) \times \dfrac{0.58}{0.55+0.58} = 0.1739 \\ P(\mathbf{Q}_3) = P(\boldsymbol{\Theta}'_5) + P(\boldsymbol{\Theta}'_6) = 0.3389 \end{cases}$$

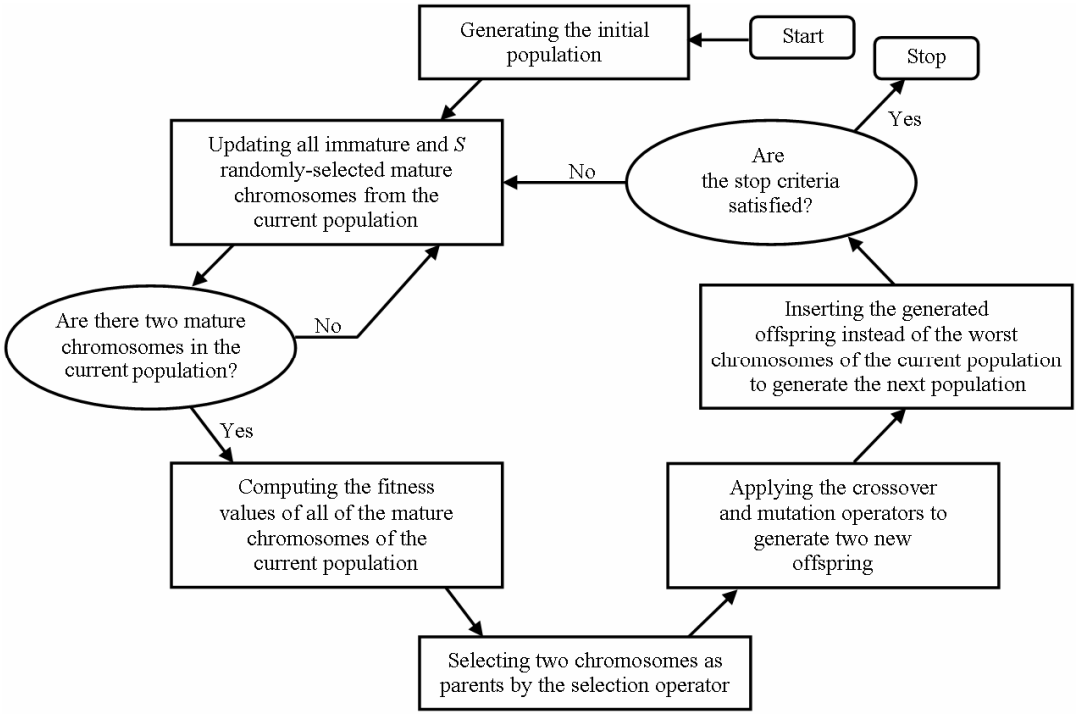

Fig. 5. EGA flowchart.

F. Generation Model

Since chromosomes in EGA can experience during evolution, the conventional generation models are no more useful. Instead, one may use the GENITOR method [36] as generation model for EGA.

G. EGA Flowchart

The EGA flowchart is shown in Fig. 5. This evolutionary algorithm initially generates a random population. The chromosomes of the current population are then updated by CUP and the process is repeated until there are at least two mature chromosomes. In the next stage, based on the fitness of chromosomes, two mature individuals are selected from the current population as parents by a selection operator. Two offspring are then generated by the parents using the crossover and mutation operators. Finally, the new population is

generated by replacing two chromosomes that have the smallest fitness by the new offspring. The above procedure is repeated until a stop criterion is satisfied.

Since in each generation, the elite chromosomes are kept in the population, according to the Rudolph theory [44], the proposed evolutionary group converges eventually to the global optimum after a sufficient number of generations. In EGA, the selection, crossover, and mutation operators and the stop criterion can be determined according to the application requirements.

H. EGA Computational Volume

Suppose that u and v are the computational volumes (the number of basic operations) for indexing and computing the recall measure for all images in the reference imagebase, respectively:

$$\begin{cases} T[\text{Indexing}(\mathbf{D};\boldsymbol{\Theta}_j)] = u \\ T[J(\boldsymbol{\Theta}_j)] = v \end{cases} \tag{38}$$

where, $T(\cdot)$ returns the computational volume. In CBIR systems, when the reference imagebase is sufficiently large, we have $u \succ\succ v$. Hence, we may write:

$$u = \alpha v,\ \alpha \succ\succ 1 \tag{39}$$

The computational volumes of the selection, crossover, and mutation operators are negligible compared to the indexing and retrieval algorithms.

Suppose that in elitism-base GA, m individuals ($m < M$) from the current population survive to the next population in each generation. Therefore, the computational volume of elitism-based GA, during k generations, is given by:

$$T[GA_{Elitism}(k, M, m)] = M(u+v) + (M-m)(k-1)(u+v) = \big(k(M-m)+m\big)(u+v) \ \Rightarrow$$

$$T\left[GA_{Elitism}(k,M,m)\right]=u\left(1+\frac{1}{\alpha}\right)\left[k(M-m)+m\right] \tag{40}$$

In particular, GENITOR can be actually considered as an elitism-based genetic algorithm with m=M-2. Thus, we have:

$$T\left[GA_{Genitor}(k,M)\right]=u\left(1+\frac{1}{\alpha}\right)\left[2k+M-2\right] \tag{41}$$

Similarly, simple GA are elitism-base GA with m=0. Thus, their computational volume is given by:

$$T[GA_{Simple}(k,M)]=u\left(1+\frac{1}{\alpha}\right)M\times k \tag{42}$$

On the other hand, the computational volume of the indexing algorithm in EGA is proportional to the number of images to be indexed. Hence, we have:

$$T\left[\text{Indexing}(\mathbf{D}_k;\boldsymbol{\Theta}_j^{\text{evn}})\right]=\frac{u}{L},\qquad k=0,1,...,L-1,\quad j=1,2,\ldots,M \tag{43}$$

The following equation can be easily derived from (7), (21), (38) and (39):

$$T\left[\hat{J}(\Theta'_j,g)\right]=\begin{cases}\dfrac{1}{\alpha}\left(\dfrac{g}{L}\right)^3 u, & g=0,1,...,L\\ 0, & \text{Otherwise}\end{cases} \tag{44}$$

$$0\le T[\hat{J}(\Theta'_j,g)]\le\frac{u}{\alpha} \tag{45}$$

Therefore, the computational volume of the evaluation function for a chromosome varies during evolution according to the following sequence:

$$0,\ \frac{1}{\alpha}\left(\frac{1}{L}\right)^3 u,\ \frac{1}{\alpha}\left(\frac{2}{L}\right)^3 u,\ \cdots,\ \frac{1}{\alpha}u$$

The mathematical expectation of the above sequence may be considered as an upper band for the expectation of the computational volume of the evaluation function:

$$0 \le E\left(T\left[\hat{J}(\Theta'_j, g)\right]\right) \le \frac{u}{\alpha} E\left(\frac{g^3}{L^3}\right)_{g=0}^{L} \tag{46}$$

If L is large enough (L>20), the above equation may be simplified as:

$$L > 20 \Rightarrow 0 \le E\left(T\left[\hat{J}(\Theta'_j, g)\right]\right) \le \frac{1}{4\alpha} u \tag{47}$$

For large number of generations ($k \to \infty$), the above expectation can be used as an upper band of the average computational volume of the evaluation function for each chromosome in each step of CUP during evolution.

In EGA, the immature chromosomes should be kept in the population during λ generations until they become mature. Therefore, the EGA computational volume after k generations ($k \to \infty$) is computed using (47), as follows:

$$\lambda M \frac{u}{L} + (k-\lambda)(S+2\lambda)\frac{u}{L} \le T\left[EG(k; M, S, \lambda)\right] \le \lambda M\left(\frac{1}{L} + \frac{1}{4\alpha}\right)u + (k-\lambda)(S+2\lambda)\left(\frac{1}{L} + \frac{1}{4\alpha}\right)u \tag{48}$$

Therefore:

$$\frac{u}{L}\left[k(S+2\lambda) + \lambda(M-S-2\lambda)\right] \le T\left[EG(k; M, S, \lambda)\right] \le u\left(\frac{1}{L} + \frac{1}{4\alpha}\right)\left[k(S+2\lambda) + \lambda(M-S-2\lambda)\right] \tag{49}$$

Using (40) and (49), the relative computational volume of elitism-based GA with respect to EGA is obtained by:

$$\lim_{k\to\infty} \frac{u\left(1+\frac{1}{\alpha}\right)\left[k(M-m)+m\right]}{u\left(\frac{1}{L}+\frac{1}{4\alpha}\right)\left[k(S+2\lambda)+\lambda(M-S-2\lambda)\right]} \le \xi_{Elitism}(m) = \lim_{k\to\infty} \frac{T\left[GA_{Elitism}(k; M, m)\right]}{T\left[EG(k; M, S, \lambda)\right]} \le \lim_{k\to\infty} \frac{u\left(1+\frac{1}{\alpha}\right)\left[k(M-m)+m\right]}{\frac{u}{L}\left[k(S+2\lambda)+\lambda(M-S-2\lambda)\right]} \Rightarrow$$

$$\frac{L(M-m)}{S+2\lambda}\left(\frac{1+1/\alpha}{1+L/4\alpha}\right) \le \xi_{Elitism}(m) \le \frac{L(M-m)}{S+2\lambda}\left(1+\frac{1}{\alpha}\right) \tag{50}$$

For $\alpha \succ\succ L$ we have:

$$\xi_{Elitism}(m) \approx \frac{L(M-m)}{S+2\lambda} \tag{51}$$

In other words, in EGA, the evolution proceeds $\xi_{Elitism}(m)=\frac{L(M-m)}{S+2\lambda}$ times faster than in elitism-based GA with $\alpha \succ\succ L$. Similarly, the relative computational volume of GENITOR and simple GA with respect to EGA are obtained by:

$$\frac{2L}{S+2\lambda}\left(\frac{1+1/\alpha}{1+L/4\alpha}\right) \le \xi_{Genitor}=\xi_{Elitism}(M-2) \le \frac{2L}{S+2\lambda}\left(1+\frac{1}{\alpha}\right) \tag{52}$$

$$\frac{ML}{S+2\lambda}\left(\frac{1+1/\alpha}{1+L/4\alpha}\right) \le \xi_{Simple}=\xi_{Elitism}(0) \le \frac{ML}{S+2\lambda}\left(1+\frac{1}{\alpha}\right) \tag{53}$$

GENITOR is considered as a fast generation model; because its computational volume is $(M-m)/2$ and $M/2$ times smaller than the elitism-based and simple GA, respectively. According to (52), EGA computational volume is $\frac{2L}{S+2\lambda}\left(\frac{1+1/\alpha}{1+L/4\alpha}\right)$ times smaller than GENITOR.

I. EGA and Evolution Theory

According to the new evolution theory, called *adaptation and natural selection* proposed by G. Williams in 1966 [38], everything in the nature is only caused by natural selection in the population. EGA corresponds better to this theory since in EGA; the chromosomes can experience new conditions during evolution. Moreover, in each generation, there is a competition among the chromosomes of the current population in which the best chromosomes win according to their current and previous performances. Elimination of a chromosome (death) in EGA is the result of harmful effects of the genes which might previously have dominant advantages. On the other hand, a chromosome, that previously had poor performance, may have more chance for survival in the current group due to its

current advantages. This behavior of EGA is in accordance with the pleiotropy [39], while it is not included in conventional GA.

IV. Experimental Results

Although EGA has been presented in the context of CBIR, its development is not dependent on this context. In other words, CBIR is a good example for indicating the limitations of the evolutionary algorithms like GA for solving optimization problems involved with large databases. Therefore, EGA is an efficient algorithm potentially applicable to problems such as feature extraction in FR and handwritten recognition [45-46], CBIR [18], data mining [47], and medical applications [48] where large databases should be processed. Wavelet correlogram is a state-of-the-art wavelet-based approach for CBIR [18, 49]. This image indexing retrieval algorithm and its variations [40] demonstrated high performance compared to other CBIR algorithms such as SIMPLIcity [17]. This section illustrates the application of EGA to optimize the quantization thresholds of the wavelet correlogram algorithm as an essential and time-consuming task.

EGA can be used to optimize other conventional CBIR systems in the same manner. For example, it can be used to optimize the color quantization thresholds in color correlogram [5], coherency and centrality thresholds in CCV [50], and the quantization thresholds of wavelet coefficients in the wavelet histogram algorithm [51].

A. Wavelet Correlogram

The block diagram of the wavelet correlogram indexing algorithm is shown in Fig. 6. As illustrated, the wavelet coefficients of each image are first computed in three consecutive

scales and then quantized by quantization thresholds shown in Fig. 7. In the next stage, the autocorrelogram [5] of the wavelet coefficients in each scale is computed using only LH and HL matrices. The resultant autocorrelogram is finally used for the construction of the image feature vector. As illustrated in Fig. 7, a simple procedure has been used for determining the quantization thresholds in the original work [40]. Here, EGA was used to optimize these thresholds.

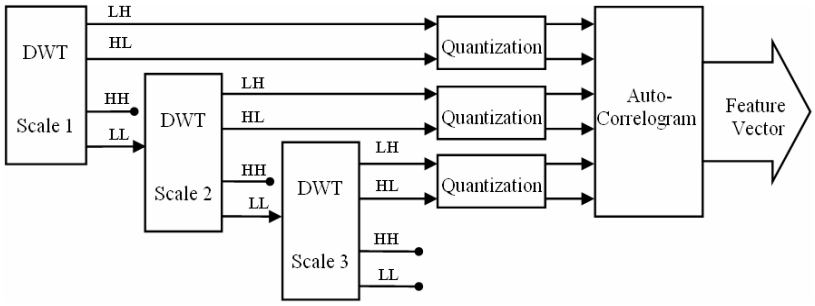

Fig. 6. Block diagram of the wavelet correlogram indexing algorithm [49].

B. Optimizing Quantization Thresholds Using EGA

To optimize the quantization thresholds of the wavelet correlogram indexing algorithm, the reference imagebase should be partitioned into L subsets. On the one hand, a large L reduces the computational volume according to (50)-(53). On the other hand, for a normal evolution progress, each subset should contain the same number of images from each category. Consequently, the possible values of L are between 1 and C=100. Obviously, selecting L=100 provides the fastest evolution progress. In contrary, the best optimization performance may be resulted with L=1 in which EGA solution and computational volume are the same as GENITOR. In these experiments, L=100 was selected for obtaining the minimum evolution time. Evolutionary genes of each chromosome are corresponded to the quantization thresholds as indicated in Fig. 8.

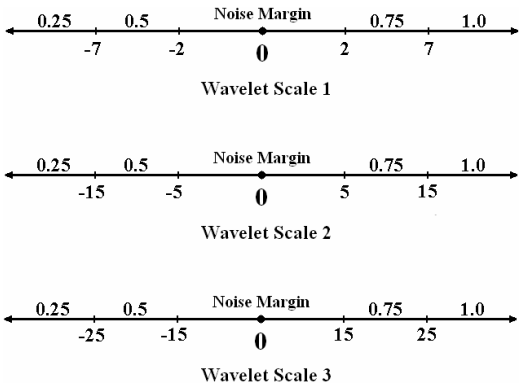

Fig. 7. Original quantization thresholds of the wavelet correlogram algorithm [18].

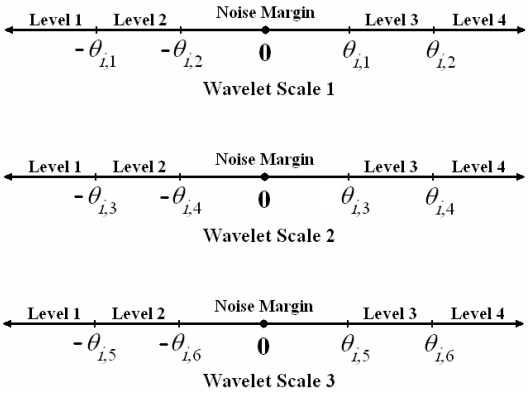

Fig. 8. Quantization thresholds introduced by chromosome *i*.

According to (50)-(53), the value of λ should be selected as small as possible to obtain the smallest computational cost. However, as illustrated in Fig. 9, the evaluation values of chromosomes decrease very fast during the first 5 generations and its variations become slower thereafter. Consequently, in order to suppress the harmful effects of the initial variations caused by small cardinality of the local featurebase (see Subsection III-C), only the chromosomes older than $\lambda = 5$ are considered as mature and participate to generation.

In this application of EGA, the evolutionary genes of only one chromosome was corresponded to the quantization thresholds appeared in Fig. 7; and for other chromosomes of the initial population, they were randomly generated (*seeding* method [52]). The population size and the number of mature chromosomes in CUP were set to M=150 and

S=0.2×M=30, respectively. Moreover, we used the *tournament selection operator* [53] in which the tournament size was set to $M/7$, *one-point mathematical crossover operator* [54], and *typical mutation operator* in which the mutation probability was set to $P_m = 0.01$ [43]. The parameter α was experimentally estimated as α=180. Consequently, using (50), (52) and (53), the relative computational volumes of Elitism-based GA, GENITOR, and simple GA with respect to EGA were:

$$330 - 2.2m \le \xi_{Elitism}(m) \le 378 - 2.52m \tag{54}$$

$$4.4 \le \xi_{Genitor} = \xi_{Elitism}(148) \le 5.0 \tag{55}$$

$$330 \le \xi_{Simple} = \xi_{Elitism}(0) \le 378 \tag{56}$$

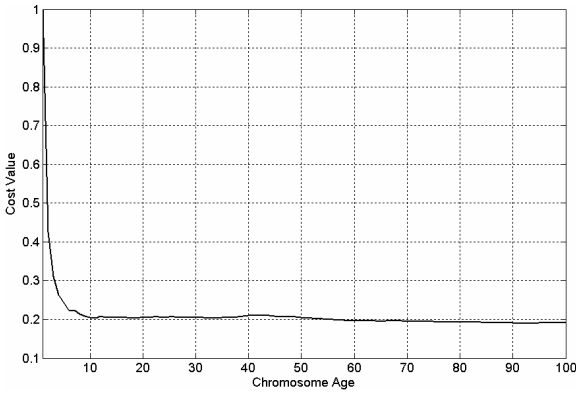

Fig. 9. Variations of the evaluation function for an EGA chromosome in different ages.

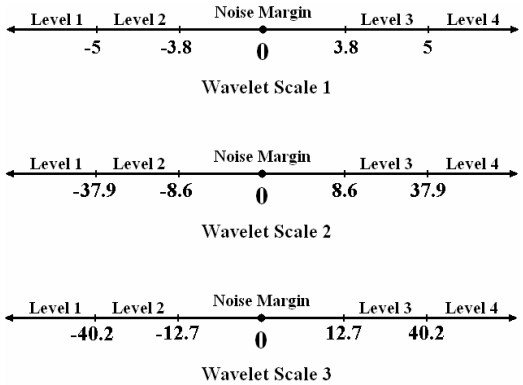

Fig. 10. Optimal quantization thresholds computed by EGA.

That means, the EGA computational volume was at least 4.4, 330-2.2m, and 330 times smaller than GENITOR, the elitism-based, and simple GA, respectively.

C. Simulation Results

All the simulations in this section were performed on a *Pentium IV 2.8 GHz* system under Matlab environment. We used EGA in order to optimize the wavelet correlogram quantization thresholds as stated in the previous subsection. Significant improvements were resulted by the algorithm evolution during 1000 generations.

Fig. 9 shows the variations of the evaluation function for the optimal chromosome obtained by EGA. As illustrated, the best performance was achieved by the oldest chromosomes with the age 100 in the final population. The optimal quantization thresholds corresponding to the best chromosome are indicated in Fig. 10.

Fig. 11 shows the histogram of chromosomes' ages in the final EGA population. The distribution of chromosomes in terms of their age is as follows: 4% immature (younger than 5), 20% young (with the age between 5 and 35), 70% middle-aged (with the age between 35 and 80), and 6% old (older than 80). Therefore, a reasonable age distribution was observed in the final population. The middle-aged chromosomes, with majority in each population, are well-behaved ones which have obtained sufficient experience and can compete with the older chromosomes. The young chromosomes are the offspring of the older ones and are considered as the investment of the population for next generations.

The above optimization algorithm took about 107 hours (equivalent to 4.5 days) long. According to Equations (50), (52), and (53), a similar optimization process with the same number of generations (k=1000 generations) takes 471, 28248, and 35310 hours

(equivalent to 19.6 days, 3.22 years and 4.03 years) for GENITOR, elitism-based GA (with m=0.20×M), and simple GA, respectively.

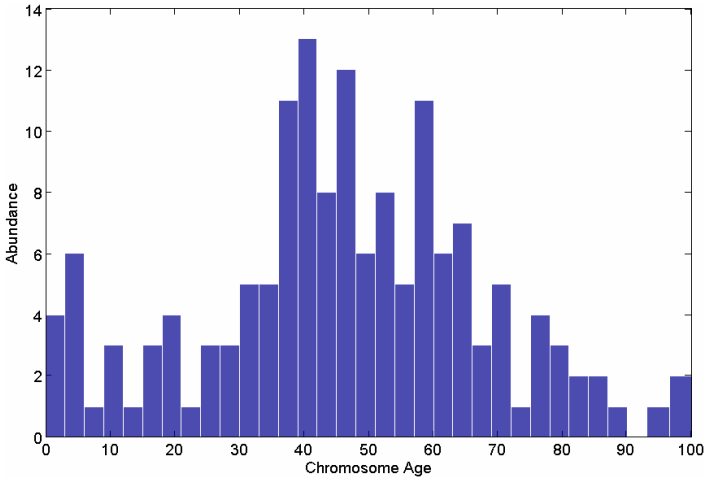

Fig. 11. The histogram of chromosomes' ages in the final population.

The fitness probability variations during evolution for the best and worst chromosomes in the population are shown in Fig. 12. As illustrated, the fitness probabilities of the chromosomes were considerably different in the first generations. However, during evolution, the poor chromosomes were replaced by better offspring and consequently, the difference between the above fitness probabilities decreased.

Tables I and II compares the performance of the wavelet correlogram CBIR with optimal and original quantization thresholds (Figs. 10 and 7, respectively) in terms of different evaluation measures including average and standard deviation (STD) of precision (P), weighted precision (Q), recall (R) and rank (C) [8, 17]. The above comparison is further illustrated in Fig. 13. For the query image $\mathbf{I}_k$, the precision is defined as follows:

$$P(\mathbf{I}_k, n; \boldsymbol{\Theta}) = \frac{1}{n}\left|\left\{ j = 1,2,\ldots,|\,\mathbf{D}\,| \,\middle|\, Rank(\mathbf{I}_j, \mathbf{I}_k; \boldsymbol{\Theta}) < n,\ \Psi(\mathbf{I}_j) = \Psi(\mathbf{I}_k) \right\}\right| \tag{57}$$

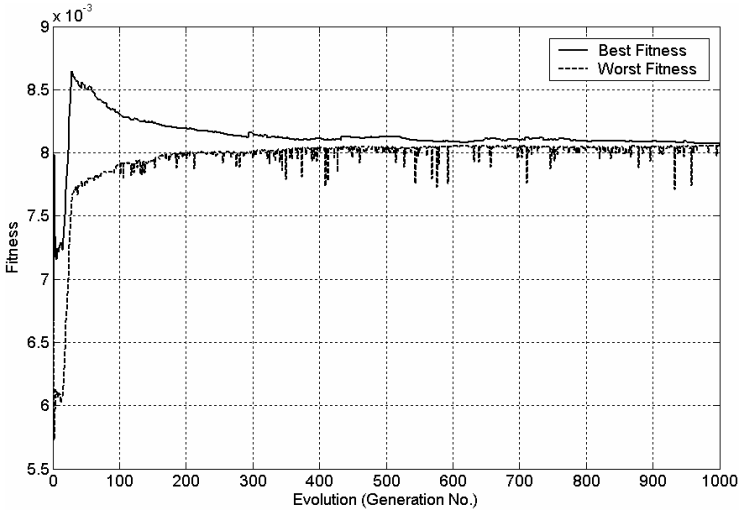

Fig. 12. Fitness variations of the best and worst chrosmosmes during EGA evolution.

where, n indicates the number of retrieved images and the function $Rank(\mathbf{I}_j;\mathbf{I}_k,\boldsymbol{\Theta})$ returns the rank of image $\mathbf{I}_j$ (for the query image $\mathbf{I}_k$) among all images of $\mathbf{D}$ when the indexing algorithm parameters are initialized by $\boldsymbol{\Theta}$. In the same manner, the weighted precision, recall, and rank are defined according to (58-60), respectively:

$$\overline{P}(\mathbf{I}_k,n;\boldsymbol{\Theta})=\frac{1}{n}\sum_{k=1}^{n}\frac{1}{k}\left|\left\{j=1,2,\ldots,|\,\mathbf{D}\,|\,\middle|\,Rank(\mathbf{I}_j,\mathbf{I}_k;\boldsymbol{\Theta})<k,\ \Psi(\mathbf{I}_j)=\Psi(\mathbf{I}_k)\right\}\right| \tag{58}$$

$$R(\mathbf{I}_k;\boldsymbol{\Theta})=P(\mathbf{I}_k,C;\boldsymbol{\Theta}) \tag{59}$$

$$R(\mathbf{I}_k;\boldsymbol{\Theta})=\frac{1}{C}\sum\nolimits_{\Psi(\mathbf{I}_j)=\Psi(\mathbf{I}_k)}Rank(\mathbf{I}_j,\mathbf{I}_k;\boldsymbol{\Theta}) \tag{60}$$

The average and STD of precision for the i-th similarity category of the reference imagebase are given by (61) and (62), respectively:

$$P_{\text{ave}}^{i}(n;\boldsymbol{\Theta})=\frac{1}{C}\sum\nolimits_{\Psi(\mathbf{I}_k)=i}P(\mathbf{I}_k,n;\boldsymbol{\Theta}) \tag{61}$$

$$P_{\text{std}}^{\text{i}}(i,n)=\sqrt{\frac{1}{C-1}\sum\nolimits_{\Psi(\mathbf{I}_k)=i}\left(P(\mathbf{I}_k,n;\boldsymbol{\Theta})-P_{\text{ave}}^{i}(n;\boldsymbol{\Theta})\right)^2} \tag{62}$$

Finally, the total average and STD of precision for the whole reference imagebase are computed by (63) and (64), respectively:

Table I. Evaluation of the wavelet correlogram with original quantization thresholds, setting n=10.

	Category	P_{ave} (%)	P_{std} (%)	$\overline{P}_{ave}$ (%)	$\overline{P}_{std}$ (%)	R_{ave} (%)	R_{std} (%)	C_{ave}	C_{std}
1	Africans	56.7	27.7	68.0	23.8	28.9	12.3	304	83
2	Beaches	49.9	28.5	63.0	24.0	27.9	16.7	345	131
3	Buildings	46.9	22.4	60.1	19.5	29.4	12.9	314	142
4	Buses	83.8	22.7	88.2	18.4	63.4	16.6	110	71
5	Dinosaurs	74.9	26.6	84.5	19.8	25.2	10.2	438	69
6	Elephants	48.5	19.3	63.4	18.1	28.3	8.2	256	48
7	Flowers	84.8	23.2	89.5	18.4	66.0	19.9	116	81
8	Horses	68.9	27.0	81.1	21.4	29.9	10.6	268	63
9	Mountains	39.9	19.8	54.9	20.3	22.1	8.3	345	72
10	Food	48.8	26.6	60.9	23.0	31.0	10.9	249	51
	Total	60.3	29.0	71.4	24.1	35.2	19.9	275	129

Table II. Evaluation of the wavelet correlogram with EGA-optimized quantization thresholds, setting n=10.

	Category	P_{ave} (%)	P_{std} (%)	$\overline{P}_{ave}$ (%)	$\overline{P}_{std}$ (%)	R_{ave} (%)	R_{std} (%)	C_{ave}	C_{std}
1	Africans	57.7	29.2	68.2	25.0	31.1	12.7	282	78
2	Beaches	49.3	28.2	61.9	23.4	28.6	16.5	335	131
3	Buildings	50.9	23.7	63.2	20.6	30.5	12.1	308	141
4	Buses	87.1	20.5	91.2	15.3	64.0	16.4	108	79
5	Dinosaurs	74.6	28.5	82.8	22.0	28.8	10.6	410	91
6	Elephants	55.7	20.9	70.7	17.5	30.7	8.4	235	44
7	Flowers	84.3	24.3	88.3	19.4	65.3	19.9	125	82
8	Horses	78.9	23.1	85.9	19.5	39.9	13.9	264	99
9	Mountains	47.2	23.7	60.0	21.2	25.1	9.7	324	79
10	Food	57.1	31.8	67.5	26.0	36.4	14.3	236	57
	Total	64.3	29.4	74.0	23.9	38.0	19.6	263	127

$$P_{\text{ave}}(n;\boldsymbol{\Theta}) = \frac{1}{|\mathbf{D}|}\sum_{k=1}^{|\mathbf{D}|} P(\mathbf{I}_k, n;\boldsymbol{\Theta}) \tag{63}$$

$$P_{\text{std}}(i,n) = \sqrt{\frac{1}{|\mathbf{D}|-1}\sum_{k=1}^{|\mathbf{D}|}\left(P(\mathbf{I}_k, n;\boldsymbol{\Theta}) - P_{\text{ave}}(n;\boldsymbol{\Theta})\right)^2} \tag{64}$$

The average and STD of the other evaluation measures are also defined in the same manner.

As can be observed, the optimal quantization thresholds improved all the evaluation measures of the wavelet correlogram algorithm. In order to verify the significance of the improvements provided by EGA optimized quantization thresholds, we applied the hypothesis testing. This test confirmed that the average precision and recall were improved

with statistically significance level ρ=0.01. It also demonstrated that the average weighted precision and rank were improved with statistically significance level ρ=0.05.

Therefore, EGA was successful to optimize effectively the indexing parameters (the quantization thresholds) of the wavelet correlogram in a shorter computational time compared to the conventional evolutionary optimizers such as GA.

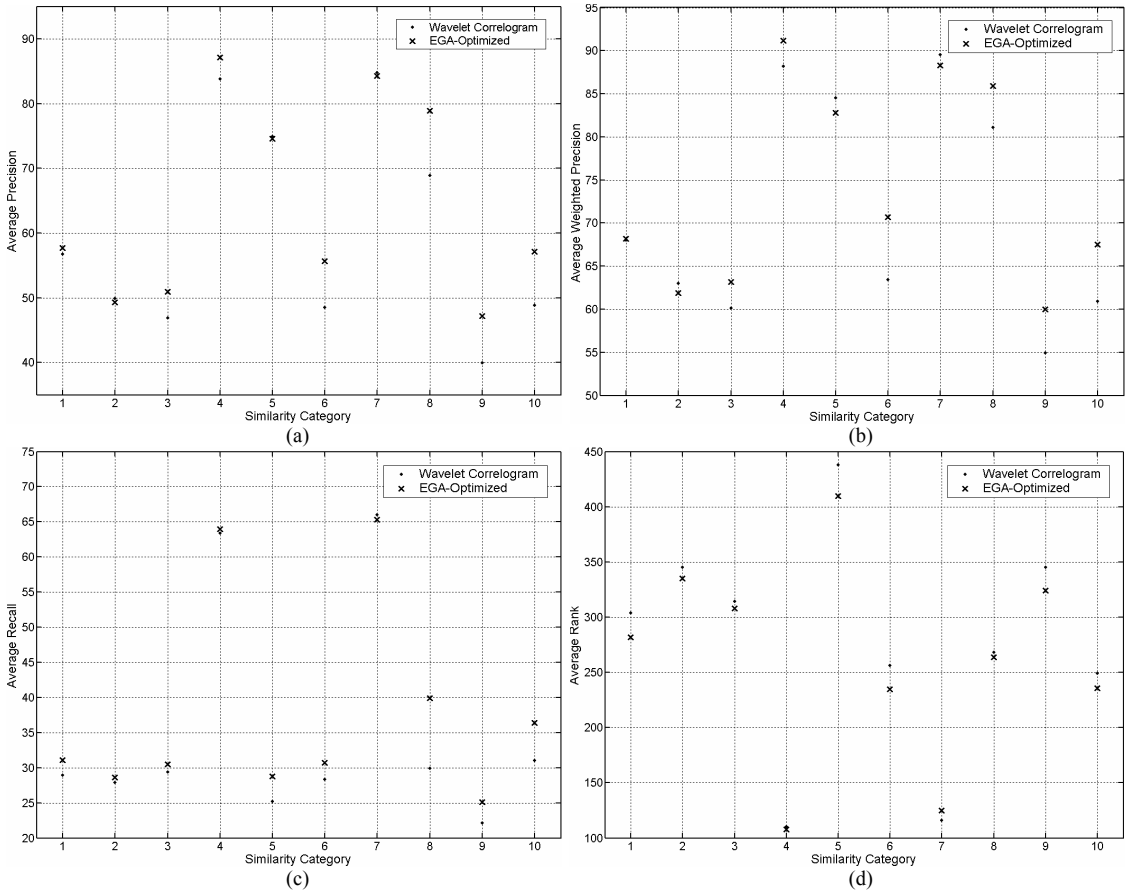

Fig. 13. Evaluation results of the wavelet correlogram algorithm with EGA-optimized quantization thresholds compared to the original quantization thresholds in terms of different evaluation measures including (a) average precision, (b) average weighted precision, (c) average recall, and (d) average rank.

In Fig. 14, the retrieval results of the wavelet correlogram algorithm with the optimal quantization thresholds (computed by EGA) for 4 query images are shown. The precision

of the algorithm, for each query image, is appeared at the right-hand side of the graphical user interface.

V. Conclusion

In this paper, a novel evolutionary method called evolutionary group algorithm was proposed for solving time-consuming optimization problems involved with large databases.

The database was partitioned into several subsets and each subset was used by an updating process as training patterns for each chromosome during evolution by EGA. Additionally, for each chromosome, an age parameter was defined for indicating the progress of the updating process. Two types of genes including evolutionary and history genes were defined for the proposed chromosomes. Evolutionary genes were able to participate to evolution, and history genes were aimed to save the previous states of the updating process. Furthermore, in each generation, a new fitness function was defined to evaluate the fitness of the chromosomes with different ages in the population.

The novel algorithm was used for optimizing the quantization thresholds of the wavelet correlogram method for CBIR. The optimal parameters, obtained by EGA in a reasonable computational time, improved significantly the performance of the wavelet correlogram algorithm for CBIR.

Although the proposed evolutionary algorithm is developed in the context of CBIR, it can be used in other applications in which large databases should be processed. We expect our developments to have application in other evolutionary algorithms as well. How this can be achieved is an open problem and further research is necessary to this end.

Acknowledgment

This work was partially supported by Iran's Telecommunication Research Center under grant No. 500/1451.

References

[1] "Special issue on digital libraries," *IEEE Trans. Pattern Anal. Mach. Intell.,* 18(8), 1996.

[2] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of early years," *IEEE Trans. Pattern Anal. Mach. Intell.,* 22(12):1340-1380, 2000.

[3] V.N. Gudivada, V. Raghavan, "Content based image retrieval systems," *IEEE Comp.*, 28(9):18-22, 1995.

[4] M.J. Swain and D.H. Ballard, "Color indexing," *Int'l J. Computer Vision*, 7(1):11-32, 1991.

[5] J. Huang, S.R. Kumar, M. Mitra, W.-J. Zhu, and R. Zabih, "Image indexing using color correlograms," in *Proc. IEEE Computer Soc. Conf. Computer Vision and Pattern Recognit.*, 1997, pp. 762-768.

[6] M. Stricker and A. Dimai, "Spectral covariance and fuzzy regions for image indexing," *Machine Vision Applications*, 10(2):66-73, 1997.

[7] F. Mahmoudi, J. Shanbehzadeh, A.M. Eftekhari-Moghadam, and H. Soltanian-Zadeh, "Image retrieval based on shape similarity by edge orientation autocorrelogram," *Pattern Recognit.*, 36(8):1725-36, 2003.

[8] R. Schettini, G. Ciocca, and S. Zuffi, "A survey on methods for color image indexing and retrieval in image databases," *Color Imaging Science: Exploiting Digital Media,* R. Luo and L. MacDonald eds., J. Wiley, 2001.

[9] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by image and video content: the QBIC system," *IEEE Computer*, 28(9):23-32, 1995.

[10] V.E. Ogle and M. Stonebraker, "Chabot: retrieval from a relational database of images," *IEEE Computer*, 28(9):40-48, 1995.

[11] A. Pentland, R. Picard, and S. Sclaroff, "Photobook: content-based manipulation of image databases," *Int'l J. Computer Vision*, 18(3):233-254, 1996.

[12] C. Carson, M. Thomas, S. Blongie, J.M. Hellerstein, and J. Malik, "Blobworld: a system for region-based image indexing and retrieval", in *Proc. 3rd Int'l Conf. Visual Information Syst.,* 1999, pp. 509-516.

[13] A.D. Bimbo, M. Mugnaini, P. Pala, and F. Turco, "Visual querying by color perceptive regions," *Pattern Recognit.*, 31(9):1241-1253, 1998.

[14] A.K. Jain and A. Vailaya, "Shape-based retrieval: a case study with trademark image database," *Pattern Recognit.*, 31(9):1369-1390, 1998.

[15] S. Abbasi, F. Mokhtarian, and J. Kittler, "Curvature scale space in shape similarity retrieval," *Multimedia Syst.*, 7(6):467-476, 1999.

[16] L. Balmelli and A. Mojsilovic, "Wavelet domain features for texture description, classification and replicability analysis", in *Wavelets in Signal and Image Analysis: from Theory to Practice* (edited book), Netherland: Kluwer Academic Publishers, 2001.

[17] J.Z. Wang, J. Li, and G. Wiederhold, "SIMPLIcity: semantics-sensitive integrated matching for picture libraries," *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(9):947-963, 2001.

[18] H. Abrishami Moghadam, T. Taghizadeh Khajoie, A.H. Rouhi, and M. Saadatmand-T., "Wavelet correlogram: a new approach for image indexing and retrieval," *Pattern Recognition*, 38(12):2506-18, 2005.

[19] M.K. Mandal, T. Aboulnasr, and S. Panchanathan, "Fast wavelet histogram techniques for image indexing," *Comput. Vis. Image Understand.*, 75(1-2):99-110, 1999.

[20] J.Z. Wang, G. Wiederhold, O. Firschein, and S.X. Wei, "Content-based image indexing and searching using Daubechies' wavelets," *Int'l J. Digital Libraries*, 1(4):311-328, 1997.

[21] G.-D. Guo, A. K. Jain, and W.-Y. Ma, and H.-J. Zhang, "Learning similarity measure for natural image retrieval with relevance feedback," *IEEE Trans. Neural Networks*, 13(4):811-820, 2002.

[22] Y. Rui, T.S. Huang, M. Ortega, and S. Mehrotra, "Relevance feedback: a powerful tool for interactive content-based image retrieval," *IEEE Trans. Circuits Syst. Video Technol.*, 8(5):644-655, 1998.

[23] J.R. Smith and C.S. Li, "Image-classification and querying using composite region templates," *Comput. Vis. Image Understand.*, 75(1-2):165-174, 1999.

[24] O. Nelles, *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*, Berline: Springer-Verlag, 2001.

[25] S. Haykin, *Neural Networks: A Comprehensive Foundation*, New Jersey: Prentice-Hall, 1999.

[26] F. Tian and L. Wang, "Chaotic simulated annealing with augmented Lagrange for solving combinatorial optimization problems," in *Proc. 26th Annual Conf. IEEE Indust. Electron. Soc.*, vol. 4, 2000, pp. 2722-5.

[27] T. Bäck, H.P. Schwefel, "An overview of evolutionary algorithm for parameter optimization," *IEEE Trans. Evolutionary Computational*, 1(1):1-23, 1993.

[28] B.-T. Zhang, J.-J. Kim, "Comparison of selection methods for evolutionary optimization," *Evolutionary Optimization*, 2(1):55-70, 2000.

[29] T. Stützle and M. Dorigo, "ACO algorithms for the traveling salesman problem," In K. Miettinen, M. Makela, P. Neittaanmaki, J. Periaux, editors, *Evolutionary Algorithms in Engineering and Computer Science*, Wiley, 1999.

[30] Á.E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Trans. Evolutionary Computation*, 3(2):124-141, 1998.

[31] R. Smith, "Adaptively resizing populations: an algorithm and analysis," in *Proc. 5th Int. Conf. Genetic Algorithms*, S. Forrest, Ed. San Mateo, CA: Morgan Kaufmann, 1993, p. 653.

[32] J. Arabas, Z. Michalewicz, and J. Mulawka, "GAVaPS –A genetic algorithm with varying population size," in *Proc. 2nd IEEE Conf. Evolutionary Computation*, 1994, pp. 73-78.

[33] M. Srinivas and L.M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Trans. Syst., Man, and Cybern.*, 24(4):17-26, 1994.

[34] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: empirical results," *Evol. Comput.*, 8(2):173-195, 2000.

[35] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitism multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evolutionary Computation*, 6(2):182-197, 2002.

[36] D. Whitley, "The GENITOR algorithm and selective pressure", in *Proc. 3rd Int. Conf. Genetic Algorithms*, Morgan-Kaufmann Ed., 1989, pp. 116-121.

[37] Z.-B. Xu, K.-S. Leung, Y. Liang, and Y. Leung, "Efficiency speed-up strategies for evolutionary computational: fundamentals and fast-GA," *Applied Mathematics and Computational*, vol. 142, pp. 341-388, 2003.

[38] G.C. Williams, *Adaptation and Natural Selection: A Critique of Some Current Evolutionary Thought*, Princeton, New Jersey: Princeton University Press, 1966.

[39] G.C. Williams, "Pleiotropy, natural selection, and the evolution of senescence," *Evolution*, 11(4):398-411, 1957.

[40] H. Abrishami Moghadam, T. Taghizadeh Khajoie, A.H. Rouhi, "A new algorithm for image indexing and retrieval using wavelet correlogram," in *Proc. IEEE Conf. Image Process.*, vol. 2, 2003, pp. 497-500.

[41] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. San Diego: Academic Press, 2003.

[42] A.D. Narasimhalu, M.S. Kankanhalli, and J. Wu, "Benchmarking multimedia databases," *Multimedia Tools and Applications*, 4(3):333-356, 1997.

[43] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, 4(2):65-85, 1994.

[44] G. Rudolph, "Convergence analysis of canonical genetic algorithms," *IEEE Trans. Neural Networks*, 5(1):96-101, 1994.

[45] Z. Suna, G. Bebisa, and R. Millerb, "Object detection using feature subset selection," *Pattern Recognition*, 37(11):2165-2176, 2004.

[46] L. S. OLIVEIRA and R. SABOURIN, "A methodology for feature selection using multiobjective genetic algorithms for handwritten digit string recognition," *Int'l J. Pattern Recognition and Artificial Intelligence*, 17(6):903-929, 2003.

[47] A.A. Freitas, "A survey of evolutionary algorithms for data mining and knowledge discovery," In A. Ghosh and S. Tsutsui, editors, *Advances in Evolution. Comput.*, pp. 819-845. Springer-Verlag, Aug. 2002.

[48] P. Zhang, B. Verma, and K. Kumar, "Neural vs. statistical classifier in conjunction with genetic algorithm based feature selection," *Pattern Recognition Letters*, 26:909-919, 2005.

[49] M. Saadatmand-T., H. Abrishami Moghadam, "Enhanced wavelet correlogram methods for image indexing and retrieval," in *Proc. IEEE Int'l Conf. Image Processing*, vol. 1, Sep. 2005, pp. 541-544.

[50] G. Pass, R. Zabih, J. Miller, "Comparing images using color coherence vectors," *Proc. 4th ACM Multimedia Conf.*, 1996.

[51] M.K. Mandal, T. Aboulnasr, S. Panchanathan, "Image indexing using moments and wavelets," *IEEE Trans. Consumer Electronics*, 42(3):557-565, 1996.

[52] M. A. Lee and H. Takagi, "Embedding a priori knowledge into an integrated fuzzy system design method based on genetic algorithms," in *Proc. 5th IFSA World Congress*, vol. II, 1993, pp. 1293-1296.

[53] D. E. Goldberg, "Genetic and evolutionary algorithms in the real world," Illinois Genetic Algorithms Laboratory *(IlliGAL) Technical Report No. 99013*, 1999.

[54] A. P. Engelbrecht, *Computational Intelligence*, England: John Wiley & Sons, 2002.

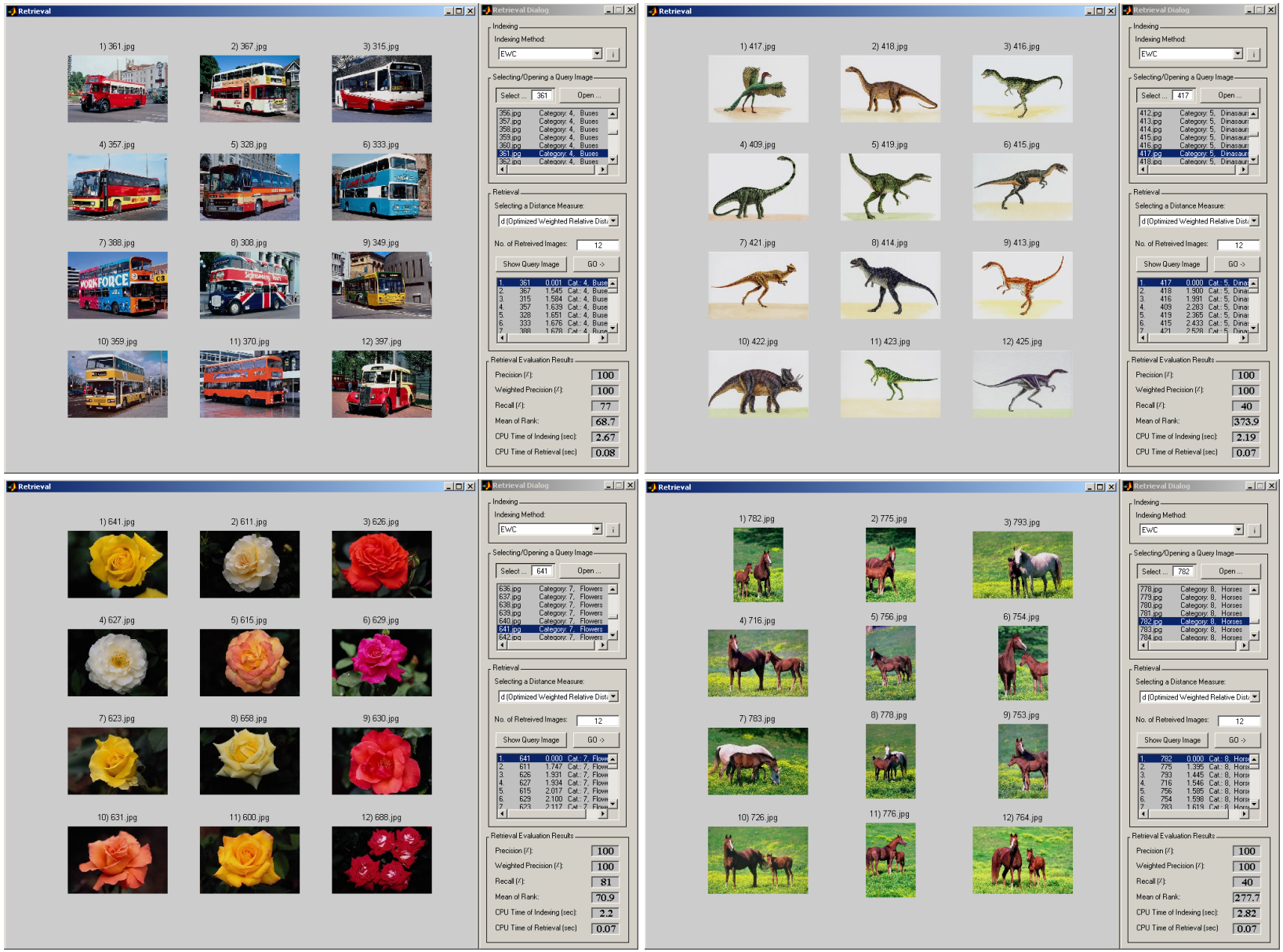

Fig. 14. Retrieval results of the wavelet correlogram image indexing and retrieval algorithm with the EGA-optimized quantization thresholds for four different query images. The precision of the algorithm is appeared at the right-hand side of the graphical user interface.