Using Assortative Mating in Genetic Algorithms for Vector Quantization Problems

Carlos Fernandes
Laseeb-ISR
Av. Rovisco Pais,1, Torre Norte,6.21,1049, Portugal
351 21 8418273
cfernandes@lovelace.isr.ist.utl.pt

Rui Tavares
Universidade Évora
R. Romão Ramalho, 59, 7000, Portugal
351 66 21744616
rt@di.uevora.pt

Cristian Munteanu
Laseeb-ISR
Av. Rovisco Pais, 1, Torre Norte, 6.21,1049, Portugal
351 96 4423118
cmunteanu@pop.isr.ist.utl.pt

Agostinho Rosa
Laseeb-ISR
Av. Rovisco Pais, 1, Torre Norte, 6.21,1049, Portugal
351 21 8418277
acrosa@isr.ist.utl.pt

Keywords

Genetic Algorithm, Assortative Mating, Genetic Diversity.

ABSTRACT

In nature, some species mate according to their phenotype similarity. The Assortative Mating Genetic Algorithm (AMGA) mimics some mechanisms of reproduction in natural environments. The main difference between AMGA and the Standard GA (SGA) is the selection of the parents in the crossover operators. We develop a similarity measure for the Vector Quantization problem and we show that the application of AMGA to some instances of this problem reduces the number of times that the algorithm becomes trapped in local optima. We also present results that show that AMGA keeps a higher level of genetic diversity than the SGA.

1. INTRODUCTION

Genetic Algorithms (GAs) [12] [14] are adaptive systems inspired by natural evolution. The Standard GA (SGA) randomly creates an initial population of solutions, also called chromosomes. Crossover operator recombines these solutions over a certain number of generations until a stop criterion is reached. The chromosomes to recombine – parents - are selected according to their fitness values: better solutions have larger probability to be chosen to cross with other solutions and generate offspring - children - that share the genetic material from both parents. Mutations may occur with very low rate. This reproduction method is called *random matin*g [19][20]. In random mating, an individual mates with any other regardless of its parenthood or likeness.

Non-random mating is common in nature and it can be divided in two general categories: inbreeding in the broad sense and assortative mating. Inbreeding in the broad sense is (as described in [19] and [20]) the occurrence of mating between relatives more or less often than expected by chance. More mating between relatives is called inbreeding in the strict sense or, more commonly, incest. Outbreeding refers to less mating between relatives than occurs in random mating.

Inbreeding in the strict sense decreases the genetic diversity in a population and outbreeding increases that same diversity [19].

Assortative mating is the occurrence of mating between individuals of similar phenotype more or less often than expected by chance. Mating between individuals with similar phenotype more often is called positive assortative mating and less often is called negative assortative mating.

In the present paper we develop a GA with assortative mating (AMGA) and apply this new algorithm to the Vector Quantization problem [10]. In general, the results are better, or at least equivalent, to those obtained by the GA with random mating. Assortative mating increases computational time, so it is best suited for problems without time restrictions.

The paper is organized as follows. Section 2 gives an overview of related work in the area of non-random mating GAs. Section 3 briefly describes the Vector Quantization problem, which will be used to test the new GA approach, which in turn is described in Section 4. Results are given in Section 5, and Section 6 concludes the paper and suggests future research.

2. RELATED WORK

Some previous research has already been done on non-random mating GAs. In [3], recombination between individuals with a certain degree of parenthood is not allowed. The authors define an incest prevention degree, which defines how far back in the family tree of an individual the GA must look to prevent a recombination between related individuals. The niGAVaPS (non-incest Genetic Algorithm with Varying Population Size) is described in [8]. The incest prevention technique is similar to [3] and the population size varies from generation to generation [18]. Each individual has a lifetime, measured in generations, defined by the quality of the solution it represents [18]. The best individuals have longer lifetimes, therefore remaining in the population for more generations, and thus increasing the probability of being chosen to recombination events. In [7] and [21], authors present a GA in which recombination events are only allowed between individuals with Hamming distance above a certain threshold.

3. VECTOR QUANTIZATION

Vector Quantization (VQ) is a generalization of Scalar Quantization to the quantization of a vector [10]. It is an NP-hard problem [2]. VQ is usually used in signal processing in order to output a signal that is a compressed version of the original one. It is applied to problems where it is not necessary that the decompressed signal is an exact copy of the original: VQ is a lossy technique. VQ recodes input sequences – vectors – by mapping each sequence onto smaller sequences - codewords. The finite, indexed set of all codewords, is the *codebook*. Compressing a signal with VQ consists of identifying the codeword for each

vector. Then, the signal is represented by the indexes of the codewords. Decompressing it consists of writing the corresponding codewords.
The dimension of the codewords and size of the codebook determine the compression ratio. If N is the codebook size then an index needs $log_2\ N$ bits for its binary representation. If n is the number of vectors and p the dimension, in bits, of each vector, then the compression ratio is given by:

(1) $$\frac{np}{n \log_2 N} = p / \log_2 N$$

The similarity between the original and the decompressed signal depends strongly on the codebook. If the vectors are close to the codewords then the two signals are similar. The method used to measure the distance between vectors and codewords is a very important criterion in VQ. It is common to use the Squared Euclidean Distance:

(2) $$d(v_1, v_2) = \sum_{i=1}^{k} \left[(v_{1i} - v_{2i})^2 \right]$$

where v_1 and v_2 are k-dimensional vectors.
The codebook generation is usually done using a finite set of training vectors, which are samples of the original signal. This is done due to the fact that, in some cases, the signal is not completely known or it is too large to allow acceptable computational time.
VQ creates a codebook to minimize the distance between signal vectors (or training vectors) and codewords so that the difference, or distortion, between the original and the decompressed signal is also minimized. With a finite set of training vectors we can define the average distortion of a codebook as:

(3) $$av\_distortion = \frac{\sum_{i=1}^{\Omega} d(v_i, cw(v_i))}{\Omega}$$

where $cw(v_i)$ is the codeword that corresponds to v_i and Ω is the training set size. The function d is the Squared Euclidean Distance.
There are two conditions that must be satisfied so that a VQ is optimal in the sense of minimizing the distortion [17]:

- The Nearest Neighbor Condition defines the optimal VQ encoding. Each vector must be mapped to the nearest codeword.
- The Centroid Condition requires that each codeword must be the centroid of the vectors that map to it.

The most commonly used algorithm for VQ is driven by these two conditions and it's called the Lloyd Algorithm.

3.1 Lloyd Algorithm

The Lloyd Algorithm, also known as *K-means* or *LBG algorithm*, is an iterative method based on the optimal codebook conditions. Figure 1 depicts the Lloyd algorithm structure, while Figure 2 describes the Lloyd itereation, on which the algorithm is based. It is proven that the average distortion of a codebook is reduced or remains unaltered after each Lloyd iteration. It has also been proved that the Lloyd algorithm converges in a finite number of iterations, although it is not certain that it converges to the optimal solution; its convergence depends on the initial codebook [17].
There are several extension and variations of the Lloyd algorithm. One example is *the local k-means algorithm* [22], which is combination of the *self-organizing map* [6] and the Lloyd algorithm. The algorithms described are named *postclustering methods* [9].

1. Creates initial codebook C_1.
2. Given codebook C_m perform the Lloyd iteration to generate the improved codebook C_{m+1}.
3. Compute the average distortion for C_{m+1}. If it has changed by a small amount since the last iteration stop. Otherwise set m+1 → m and go to step 2.

Figure 1. Lloyd Algorithm

1. Given a codebook, C_m, partition the training set into clusters using Nearest Neighbor Condition.
2. Using the Centroid Condition, compute the centroids for the clusters to obtain the new codebook C_{m+1}. If an empty cell was generated in step 1, replace the centroid computation for that cell for an alternate codeword.

Figure 2. Lloyd iteration for empirical data

Starting with an initial codebook a postclustering algorithm iteratively tries to improve the solution. Other set of VQ algorithms is called *preclustering methods*. These algorithms divide the space into clusters of similar vectors and then determine the representative vector (codeword) for each cluster. The *median cut algorithm* [13], the *octree quantization algorithm* [11] or the *bisplit algorithm* [23] are examples of preclustering methods.
Evolutionary Algorithms [9] [24] (as well as other stochastic techniques, like for instance *Simulated Annealing* [4]), were also successfully applied to the VQ problem. In [9] it is shown that a combination of the Lloyd algorithm and the GA outperforms other methods in the color image Quantization problem.

4. NEW GA APPROACH

The purpose of this GA is to generate a codebook for a given training sequence of vectors. Genomes are sequences of vectors representing possible codebooks for the problem, as Figure 3 illustrates.
Based on [5], we developed a GA with real number alleles and operators to deal with this representation.

(0.2000, 0.2100), (0.1510, 0.8005), (0.5010, 0.4017)

Figure 3. Example of a genome representing a codebook of size 3 with two-dimensional codewords.

We developed two crossover operators, when implementing the GA, to deal with the VQ problem. The first one is based on Davis' *average crossover* [5]. Our crossover takes two parents and randomly chooses a cross gene. It behaves exactly like one point crossover except in the case of the cross gene. The corresponding allele of the children is the result of averaging the cross gene alleles of the parents – see Figure 4.

Parents
(0.2000, 0.2100), (*0.2000*, 0.8005), (0.5010, 0.4017)
(0.3000, 0.1000), (*0.3000*, 0.7505), (0.1000, 0.3013)
Children
(0.2000, 0.2100), (*0.2500*, 0.7505), (0.1000, 0.3013)
(0.3000, 0.1000), (*0.2500*, **0.8005), (0.5010, 0.4017)**

Figure 4. Example of the average crossover used in this GA implementation

In this problem and with this genetic representation it is possible that two different genomes codify the same codebook. That happens if the same codebook is codified in the genomes but the

codewords are in a different order. Each one of these individuals, if recombined with other, generates different children even if the crossover point is the same. To take advantage of this characteristic and increase genetic diversity we developed a crossover based on Messy Genetic Algorithm [15]. This operator randomly exchanges the position of the vectors in the parents' genome before the crossover. An example of Messy Crossover can be seen in Figure 5.

Parents
(0.1000, 0.1000), (0.2000, 0.3000), (0.0000, 0.0000)
(0.0000, 0.5000), (0.1000, 0.2000), (0.5000, 0.5000)
Parents after changing the codewords positions.
(0.2000, 0.3000), (0.0000, *0.0000*), (0.1000, 0.1000)
(0.5000, 0.5000), (0.0000, *0.5000*), (0.1000, 0.2000)

Children
(0.2000, 0.3000), (0.0000, *0.2500*), (0.1000, 0.2000)
(0.5000, 0.5000), (0.0000, *0.2500*), **(0.1000, 0.1000)**

Figure 5. Messy Crossover.

Tests made showed that a small amount of messy crossover in each generation improved the quality of the results. The mutation operator used was the Gaussian mutation [1].
To evaluate the solution we used the function (3) that measures the average distortion of the codebook.
The fitness scaling function used in this algorithm was the following [16]:

$$f = k^{\frac{distortion\_best\_distortion}{median\_distortion\_best\_distortion}}, k \in]0,1[\quad (4)$$

where the constant k define the selection pressure of the function: the lower the value of k the higher is the selectivity of the function.
Using the values returned by the fitness scaling function, selection of the parents for crossover is done using the *Roulette wheel Parent Selection Method* [12]. In the following text we will refer to the algorithm just described as GA.

4.1 Similarity Measure

In order to develop an assortative mating GA a similarity measure is needed to compare the characteristics of the individuals. For the VQ problem the genomes are sequences of vectors representing possible codebooks. Two codebooks are similar if they are closely distributed in space. We used a statistical measure of similarity of two codebooks based on the magnitude of the signal and the mean, variance and correlation of the codewords' coordinates.

$$S(c_1, c_2) = 1 / \left(\frac{d(centroids)}{magnitude} + \frac{d(\text{var})}{magnitude^2} + \frac{1}{CR} \right) \quad (5)$$

where c_1 and c_2 are codebooks, and

$$CR = \frac{\sum_{i=1}^{\kappa} correl(c_{1\kappa}, c_{2\kappa})}{\kappa} \quad (6)$$

and d(centroids) and d(var) are the Squared Euclidean Distances of the codebooks centroids (mean) and variances. The CR function measures the correlation between coordinates of the two codebooks which are κ-dimensional vectors. For two codebooks equally distributed in space the function S returns the value 1.
It is known that mutation increases genetic diversity in the population and that same diversity depends on the selectivity of selection method. To test the efficiency of this measure we ran the GA with different mutation rates and fitness scaling function selectivity to generate codebooks of size 3 with the training vectors represented in Figure 6. This set of 300 two-dimensional vectors has a Gaussian distribution with zero mean and unit variance.

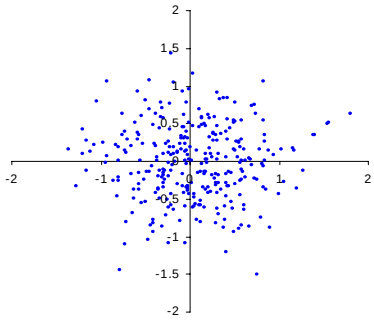

Figure 6. Set of training vectors with zero mean and unit variance (300 samples).

The results are shown in Figure 7 and Figure 8. For each configuration 100 runs were made each evaluating 720 genomes corresponding to 60 generations. Population size was 60 and crossover probability 0.2. The *average crossover* and *messy crossover* probability values were, respectively, 0.8 and 0.2. Genetic diversity was measured by randomly selecting 30 pairs (population_size/2) of genomes from the population and computing the median value of each pair's similarity.

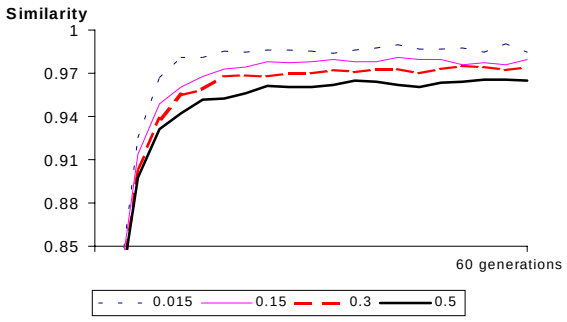

Figure 7. Diversity with different mutation rates (k=0.2).

Figure 7 illustrates that increasing the mutation rate causes the genetic diversity to increase as expected.

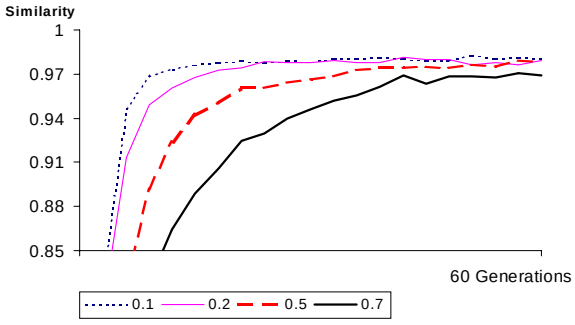

Figure 8. Diversity with different values for k (mut_rate=0.15).

It is also known that increasing the selection pressure of the selection method forces the diversity in the population to decrease faster. The results in Figure 8 show that the average similarity in the population grows slower with k values that lead to lower selection pressure. These results suggest that the similarity measure is correct for this problem. The results expected by theory were observed in the experiments.

4.2 Assortative Mating

The difference between AMGA and GA is the method used for parent selection in the crossover operators. With assortative mating the first parent is selected by the Roulette Wheel method and N genomes are selected by the same method. Then, the similarity between each of the N genomes and the first parent is computed. If assortative mating is negative (nAMGA) then the one with less similarity is chosen. If it is positive (pAMGA) the genome more similar to the first parent is chosen to be the second parent. We will see in the next section that the higher the N the better AMGA performs, but with increasing computational time.

5. RESULTS

We tested the Lloyd Algorithm, the GA and the AMGA with several sets of artificial training vectors. With some training sets the Lloyd algorithm reached the optimal solution in all the runs. For those same sets all the GAs returned solutions very close to the optimal. Applying some iterations of the Lloyd Algorithm to those solutions led to the optimal codebook.

The effects of assortative mating are best noticed with instances of the VQ problem that could not be solved in 100% of the runs either by the Lloyd algorithm or by the GA. We will see that in those cases the negative assortative mating GA becomes trapped in local optima less often than Lloyd and GA.

With the training vectors of Figure 6 and generating codebooks of size 3 we obtained the results of Figure 9. The values represent the percentage of runs in which the optimal solution was obtained. In the case of the GA the Lloyd algorithm was applied to the best genome obtained [9]. For all the GAs, we made 100 runs for each configuration each evaluating 1200 genomes in 100 generations. The mutation rate is 0.3 and k = 0.5. The values of the other parameters are the same we used in Section 4.1. Tests revealed that these values maximize the GA performance with this instance of the problem.

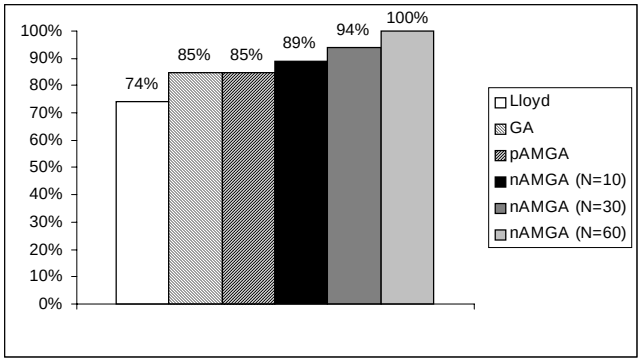

Figure 9. Comparative results *(success rates)* of Lloyd algorithm, GA, positive assortative mating GA (pAMGA) and negative assortative mating GA (nAMGA) with different values for N.

We can see that nAMGA clearly outperforms GA. The Lloyd algorithm doesn't converge to global optima with an initial local optima codebook. The success rate of 100% in AMGA with N=60 means that the algorithm converged to global optima in all runs. The GA converged to local optima in 15% of the runs. The performance increases with the size of genome set (N) from which the second parent is chosen.

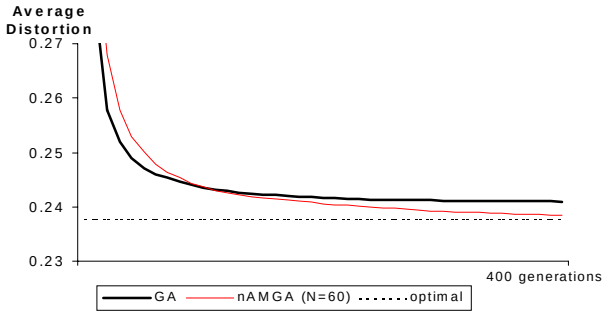

Figure 10. Compartive results of best genomes cost.

Figure 10 shows the evolution of the average fitness of the best genome in the GA and nAMGA with N=60. We see that the nAMGA convergence is slower in the beginning but after less than 100 generations it outperforms the GA. This is probably due to the fact that populations in nAMGA have more genetic diversity than in GA as we can see in Figure 11.

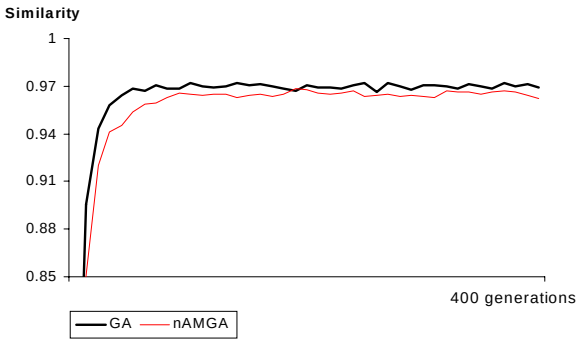

Figure 11 – Genetic diversity in GA an nAMGA with N=60.

The nAMGA explores the search space avoiding premature convergence to local optima. Increasing the mutation rate, a similar genetic diversity can be achieved, but with poorer results because of the destructive effect of high mutation rates.

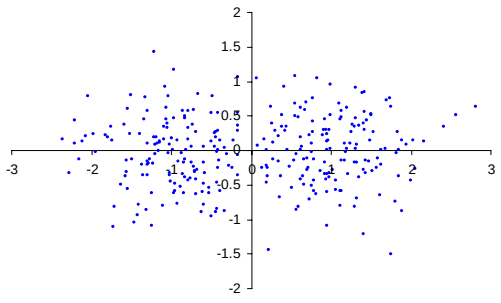

Figure 12. Training set of two-dimensional vectors.

The next example uses the training vectors of Figure 12. With this training set the nAMGA outperformed GA but only slightly. But as we can see in Figure 13 the success rate of nAMGA is clearly better. GA becomes trapped in local optima in 38% of the runs while nAMGA only becomes trapped in 25%.

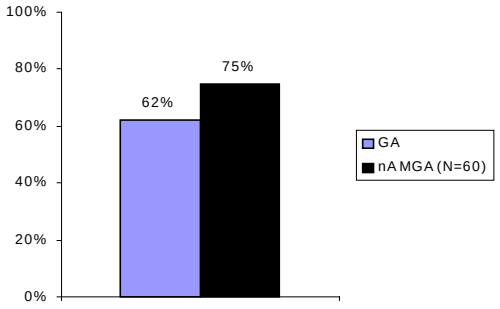

Figure 13. Success rates.

6. CONCLUSIONS AND FUTURE WORK

The negative assortative mating GA, by a broader exploration of the search space, due to higher diversity, becomes trapped in local optima less often in some problems. It increases genetic diversity in the population by mating dissimilar genomes with higher probability. nAMGA performs the similarity calculations at the expense of an increased computational time. This algorithm may be useful in problems for which finding the global optimum is required without severe time restrictions. With the VQ problem the use of positive assortative mating didn't increase GA performance although it is possible that it may be useful with other problems. Future tests may confirm this hypothesis.

We aim at testing AMGA in a range of NP-complete problems, as far as convergence and quality of solutions are concerned. The problems in which standard GA often becomes trapped in local optima are probably the best choice to apply nAMGA.

7. REFERENCES

[1] Back T. "Evolutionary Algorithms in Theory and Practice". Oxford University Press, 1996.

[2] Cormen T. H., Leiserson C. E., Rivest R. L. "Introduction to Algorithms". MIT Press, 1990.

[3] Craighurst R., Martin W. "Enhancing GA Performance through Crossover Prohibitions Based on Ancestry". *Proceedings of the Sixth International Conference on Genetic Algorithms*. Morgan Kaufmann, 1995.

[4] Davis, L.. "Genetic Algorithms and Simulated Annealing". Morgan Kaufmann Publishers, Inc., 1987.

[5] Davis, L. "Handbook of Genetic Algorithms", Van Nostrand Reinhold, 1991.

[6] Dekker A. "Kohonen Neural Networks for Optimal Colour Quantization". Network: Computation in Neural Systems, 5:351-367, 1994.

[7] Eschelman L.J., Schaffer J.D. "Preventing Premature Convergence in Genetic Algorithms by Preventing Incest". *Proceedings of the Fourth International Conference on Genetic Algorithms*. Morgan Kaufmann, 1991.

[8] Fernandes C., Tavares R., Rosa A. "niGAVaPS – Outbreeding in Genetic Algorithms". Proceedings of the 2000 ACM Symposium on Applied Computing. Villa Olmo, Como, Italy, 2000.

[9] Freisleben B., Schrader A. "An Evolutionary Approach to Color Image Quantization". *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC 97)*. Indianapolis, IN, USA, 1997.

[10] Gersho A., Gray R. M.. "Vector Quantization and Signal Compression". Kluwer Academic Publishers, 1992.

[11] Gervautz M., Purgathofer. "A Simple Method for Color Quantization: Octree Quantization". Graphics Gems, Academic Press, New York, 1990.

[12] Goldberg, David E. "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley Publishing Company, Inc., 1989.

[13] Heckbert P. "Color Image Quantization for Frame Buffer Display". *ACM Computer Graphics*, Vol. 16, No. 3, 7, pp297-307, 1982.

[14] Holland J.H. "Adaptation in Natural and Artificial Systems", MIT Press, Cambridge, Massachusetts, 1975.

[15] Kargupta, H. "The Gene Expression Messy Genetic Algorithm", *Proceedings of the IEEE International Conference on Evolutionary Computation*, Nagoya, Japan, 1996.

[16] Lima J.A., Gracias N., Pereira H., Rosa A.C.. "Fitness Function Design for Genetic Algorithms in Cost Evaluation Based Problems". Proc. IEEE - Int. Conf. Evolutionary Computation, ICEC'96 pp 207-212, 1996.

[17] Linde Y., Buzo A., Gray R. "An Algorithm for Vector Quantization Design". IEEE Transactions on Communications, COM-28(4):84-95, 1980.

[18] Michalewicz Z. "Genetic Algorithms + Data Structures = Evolution Programs" (second, extended edition). Springer-Verlag, 1994.

[19] Roughgarden J. "Theory of Population Genetics and Evolutionary Ecology". Prentice-Hall, 1979.

[20] Russel P.J. "Genetics". Benjamin/Cummings, 1998.

[21] Schaffer D., Mani M., Eshelman L., Mathias K. "The Effect of Incest Prevention on Genetic Drift". Foundation of Genetic Algorithms 5, Morgan Kauffman, 1999.

[22] Verevka O., Prunsinkiewicz, Wong S. "Variance-based Color Image Quantization for Frame Buffer Display". COLOR research and application, 15(1), 1988.

[23] Wu X., Witten I. "A Fast K-means Type Clustering Algorithm". Research Report No.85/197/10, Dept. of Computer Science, Univ. of Calgary, 1985.

[24] Zhen X., Julstrom B., Cheng W. "Design for Vector Quantization Codebooks Using a Genetic Algorithm". *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC 97)*. Indianapolis, IN, USA, 1997.