ANTICIPATORY VERSUS TRADITIONAL GENETIC ALGORITHM

Irina Mocanu[1]
Eugenia Kalisz[2]

ABSTRACT

This paper evaluates the performances of a new type of genetic algorithms - anticipatory genetic algorithms (AGA) versus traditional genetic algorithms (GA). The performances are evaluated based on two simple problems using different genetic operators. The evaluation included in the paper shows that AGA is superior to traditional genetic algorithm from both speed and accuracy points of view. Then we evaluate the two types of genetic algorithms for solving the problem of image annotation, which will be used in content based image retrieval systems. In this case the AGA performances are superior to GA, too.

Keywords: genetic algorithm, anticipation, image annotation, performance evaluation.

1. INTRODUCTION

Many optimization problems are very complex and quite hard to solve by conventional optimization techniques. Since the 1960s there has been an increasing interest in imitating living beings to solve such kinds of hard optimization problems. Simulating the natural evolutionary process of human beings results in stochastic optimization techniques called evolutionary algorithms. Among these techniques, the genetic algorithms are the most widely known evolutionary algorithms as in (Gen and Cheng, 1997).

Genetic algorithms are used in different problems: optimization problems, cellular automata, machine and robot learning (classification and prediction), designing neural networks to evolve rules for learning classifier systems, population genetic models, interactions between evolution and learning, models of social systems.

There may be different types of genetic algorithms. The first part of the paper (section 2 and 3) overviews the traditional genetic algorithm and an approach of the genetic algorithms based on anticipation – the Anticipatory Genetic Algorithm. Section 4 presents image annotation performed with genetic algorithms (GA and AGA). Section 5 describes a framework for evaluation the two types of genetic algorithms (traditional versus anticipatory) and Section 6 presents the results obtained from experiments. The last section is dedicated to the conclusions.

[1]Lecturer, Ph.D., University Politehnica of Bucharest, Computer Science and Engineering Dept., Splaiul Independentei 313, sector 6, 060042, Bucharest, Romania, irina.mocanu@cs.pub.ro
[2] Proffessor, Ph.D., University Politehnica of Bucharest, Computer Science and Engineering Dept., Splaiul Independentei 313, sector 6, 060042, Bucharest, Romania, eugenia.kalisz@cs.pub.ro

2. TRADITIONAL GENETIC ALGORITHMS

Genetic algorithms (GA) are adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection and genetic. The basic concept of GA is designed to simulate processes in natural system necessary for evolution, specifically those that follow the principles first laid down by Charles Darwin of survival of the fittest.

A genetic algorithm is a programming technique that mimics biological evolution as a problem-solving strategy. Given a specific problem to solve, the input to the GA is a set of potential solutions to that problem, encoded in appropriate way, and a fitness function that allows each candidate to be quantitatively evaluated.

The usual form of genetic algorithms was described in (Goldberg, 1989). The structure of the traditional genetic algorithm is presented in Figure 1.

They differ from conventional search techniques; they start with an initial set of random solutions called population. Each individual in the population is called a chromosome and represents a possible solution to the problem. A chromosome is a string of symbols, usually, not necessarily, a string of bits. The chromosome evolves through successive iterations called generations. During each generation, the chromosomes are evaluated, using some measures of fitness.

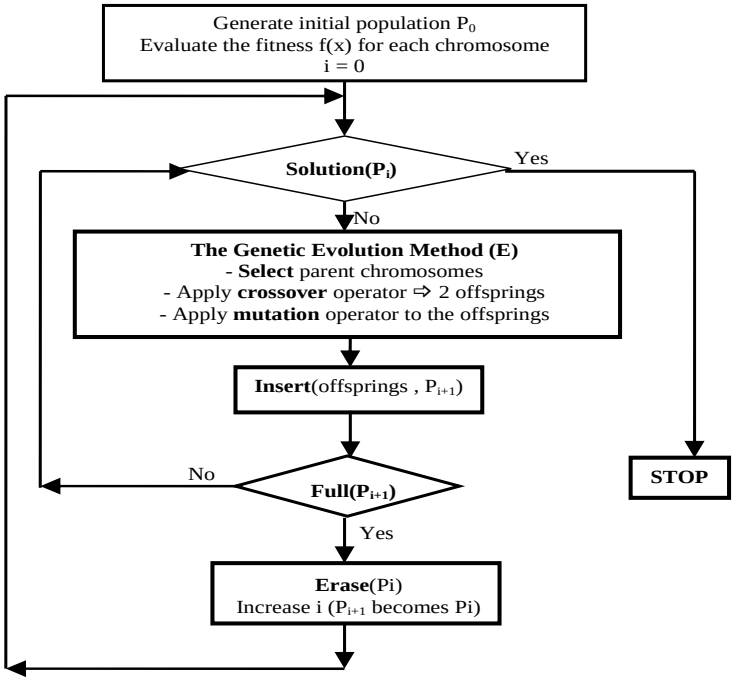

Figure 1: Traditional genetic algorithm

To create the next generation, new chromosomes, called offsprings, are formed using the genetic operators: selection, recombination (crossover) and mutation. After several

generations, the algorithm converges to the best chromosome, which hopefully represents the optimum or suboptimal solution of the problem.

The selection operator allows strings with higher fitness to appear with higher probability in the next generation. One possibility for crossover is uniform crossover. It is performed between two selected individuals (parents), by exchanging parts of their strings, starting from a randomly chosen crossover point. This operator tends to enable the evolutionary process to move towards promising regions of the search space. Mutation is used for exploring variety of data.

The selection, crossover and mutation procedures selected for a specific problem define the genetic evolution method E: $P_{i+1} = E(P_i)$.

In literature are presented many implementation versions for each component of the genetic evolution method (different types of selection, crossover and mutation as in (Mitchell, 1996), (Goldberg, 2002) or (Genetic Server/Library, 2009). Selection operator may be (i) roulette wheel selection (the chance of a chromosome getting selected is proportional to its fitness), (ii) tournament selection (it uses roulette selection N times to produce a tournament subset of chromosomes – the best chromosome from this subset will be chosen), (iii) truncated selection (randomly selects a chromosome from the top N percent of the population), etc. Crossover combines two parents (chromosomes) to produce a new offspring (chromosome). Based on crossover, the new chromosome may be better than both parents if it takes the best characteristics from each of the parents. The crossover may be: (i) one point crossover, (ii) two points crossover, (iii) uniform crossover, etc. Mutation alters one or more gene values in a chromosome. Mutation helps to prevent the population from stagnating at any local optima. Mutation may be: (i) flip bit mutation (inverts the value of the chosen gene), (ii) boundary mutation, (iii) uniform mutation, etc.

Each problem requires an appropriate combination of procedures, defining its specific method E.

3. ANTICIPATORY GENETIC ALGORITHM - AGA

In (Mocanu, 2010) we have introduced the Anticipatory Genetic Algorithm – AGA, which has an anticipation behavior (Dubois, 2000). This algorithm is based on the idea that each new generation should be closer to the solution than the previous one. In order to insure this important property, candidate chromosomes that, from the fitness point of view, are worse than the worst in the previous generation are rejected. The AGA algorithm is described in Figure 2.

Consequently, each population is characterized by its Evolution Threshold (ET), which is represented by the worst fitness value (WFV) computed for its members. According to the particular problem solved, WFV is either the lowest or the highest fitness value in the considered population.

Let us consider the current population P_i and its evolution threshold, ET_i. Our anticipatory genetic algorithm uses ET_i for insuring that the next population is closer to the solution. It follows that: $|S - ET_{i+1}| < |S - ET_i|$, where S represents the solution fitness value.

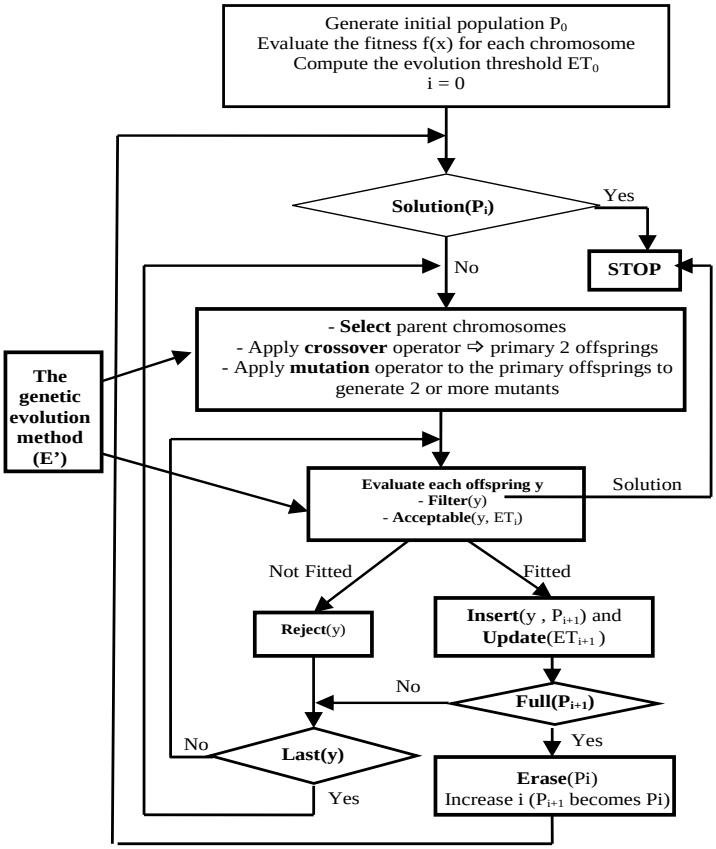

Figure 2: Anticipatory genetic algorithm

This evolution property is achieved by filtering the new generated chromosomes, which are either included in P_{i+1} or rejected. In order to compensate the rejection effect, each pair of selected parent chromosomes generates 4 or more offsprings: the first two by crossover and the next as mutates clones of the first ones.

In this case the genetic evolution method E' consists of selection, crossover, mutation and filter procedures selected for a specific problem: $P_{i+1} = E'(P_i, ET_i)$.

4. IMAGE ANNOTATION WITH GENETIC ALGORITHMS

The term of content based image retrieval has been widely used to describe the process of retrieving desired images from a large collection on the basis of features (such as color, texture and shape) that can be automatically extracted from the images themselves. A query image can be described by three levels of abstraction: (i) Primitive features: color, shape or

texture; (ii) Logical features: the identity of objects; (iii) Abstract attributes: the significance of the scenes depicted.

Based on the above characterization of queries, they may be classified into three levels as in (Eakins, 1996), (Eakins, 1998): (i) level 1 comprises retrieval by primitive features, (ii) level 2 comprises retrieval by logical features (logical inference about the identity of the objects from an image), (iii) Level 3 comprises retrieval by abstract attributes (high-level reasoning about the meaning and purpose of the objects or scenes from an image).

Many authors as (Gudivada & Raghavan, 1995) see level 2 and level 3 as semantic image retrieval. At present the most significant gap is between level 1 and level 2. For example, a query for "sky" will return scenes of both daytime (where sky is mostly blue) and sunsets (where sky tends to be orange) with equal ease. It is usually quite difficult to design a set of visual features that captures all the relevant dimensions of image variability (e.g. that sky can be either blue or orange). There is evidence that combining primitive image features with text keywords can overcome some of these problems. For this purpose methods which associate semantic information to the visual content have already been developed.

In (Mocanu, 2009) a genetic algorithm for image annotation is presented. The proposed algorithm is a classical genetic algorithm – based on the structure of GA - IA_GA (Image Annotation Genetic Algorithm).

The vocabulary used for image annotation consists of a set of words and each word is associated with a representative image. Image annotation is based on the color and shape similarities. The histogram color method is used for color similarity and centroid radii and turning angle method is used for shape similarity.

Before applying the genetic algorithm for image annotation a preprocessing is needed: each image is segmented into component regions R_1, R_2, …, R_N. Then a similarity degree S_{ij} is assigned to each region R_i, S_{ij} is based on the similarity of the low level features (color and shape) between R_i and the image associated with the word W_j from the vocabulary. The color feature and the shape feature are considered in equal ratios – and normalized in interval [0, 1].

The vocabulary images are classified based on shapes (using the k-means clustering algorithm), thus a region R_i will be compared only with a part of the vocabulary images – only with the images from the cluster to which R_i is closest (the cluster which has the minimum distance between R_i and its centroid.

A chromosome in IA_GA will be represented by N values $\{V_1, V_2, \ldots, V_N\}$, where V_i represents the word W_{Ni} associated with region R_i. At the end of the algorithm W_{Ni} will annotate the region R_i. The initial population is generated randomly, based on the vocabulary used for annotation.

The fitness function for IA_GA must express the similarity degree between an object and the image associated with its current associated keyword (from the vocabulary). The fitness function is based on the color distance and on the shape distance.

Based on the tests performed, better performance was obtained with tournament selection operator and uniform crossover. For mutation it is used something like bit flip mutation – the corresponding value of a gene is modified with a value which is associated with an image from the same cluster.

Based on the idea of AGA we can modify the IA_GA and note this modified algorithm IA_AGA. As in AGA, for each new generation four children (chromosomes) are created: two obtained by crossover, while other two will be mutated clones of the first ones. The evolution threshold for a new generation **i+1** (ET_{i+1}) is represented by the maximum fitness value of the individuals from generation **i**. A newly generated chromosome will be considered appropriate for the next generation and will be included in the P_{i+1} population only if its fitness value is less than ET_i.

5. A FRAMEWORK FOR PERFORMANCE EVALUATION

We propose a genetic algorithm framework based on two main modules: GA, which implements the skeleton of the traditional genetic algorithm, and AGA, implementing the anticipatory one. These two modules include calls to the generic modules Chromosome Structure, Selection, Crossover, Mutation, Fitness and Filter. The structure of the developed framework is shown in Figure 3.

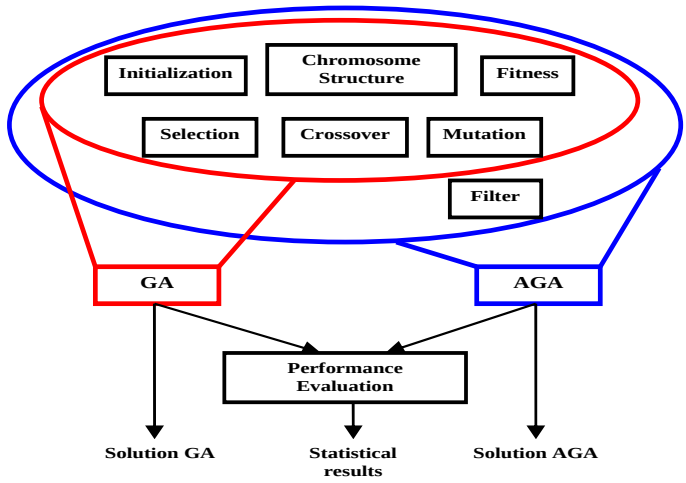

Figure 3: The framework for algorithms evaluation

The framework user tailors these modules for his/hers needs by selecting an appropriate set of parameters such as maximum number of generations, number of chromosomes in

population (Initialization module), structure and length of chromosome representation (Chromosome Structure module), selection method (Selection module), number of crossover points and crossover type (Crossover module), fitness function (Fitness module), type of filter (up / down) (Filter module). The results of this parameterization process are a specific evolution method, applicable in both GA and AGA context.

The user can choose between 3 operation modes: GA, AGA or performance evaluation. The GA and AGA modules are used to solve a problem using a traditional genetic algorithm, or an anticipatory genetic algorithm. The third mode is used to obtain comparative statistical information concerning GA and AGA, by repeatedly running both genetic algorithms, each run using the same initial population for GA and AGA. The results are compared by the number of population generated until the result is obtained and by the number of generated chromosomes (including the rejected ones in the case of AGA). The provided comparative results are on the average number of generations and the average number of generated chromosomes.

6. PERFORMANCE EVALUATION

For AGA versus GA performance evaluation we have considered two types of problems:

simple problems:

- MAX: finding the maximum value of the function $f(x) = x^2$, (x is a signed number represented by 6 bits);
- DEC: finding a decomposition for a natural number N in $x^2+y^2+z^2+u^2+w^2$, where x, y, z, u, w are digits. If such decomposition is not found, the result must represent the decomposition of a number close to N.

using the genetic algorithms for solving the problem of image annotation.

I. Simple Problems Evaluation

We compare the results obtained by both GA and AGA operating under the same conditions as follows: selection - roulette wheel selection, tournament selection and truncated selection; crossover – one point crossover; mutation - affects one bit / all bits (with a probability) of a chromosome (flip bit mutation); the same initial population;

The performances of GA versus AGA are evaluated by: (i) the number of generations until solution is found; (ii) the total number of chromosomes until solution is found (including the rejected chromosomes in case of AGA); (iii) the obtained solution value; (iv) using different types of selections, and (v) the average of the above criteria.

A. Max Evaluation

In this case we use: fitness function - x^2, 6 individuals in each population, maximum number of generations: 100, 40 test runs.

The results for roulette wheel selection are presented in Figure 4a, b and c (GA versus AGA): Figure 4.a presents the number of generations until solution; Figure 4.b presents the number of chromosomes until solution; Figure 4.c presents the solution value obtain.

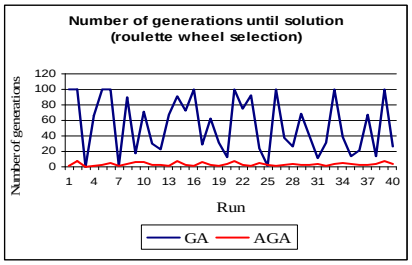

Figure 4.a: Number of generations until solution (GA versus AGA – roulette wheel selection)

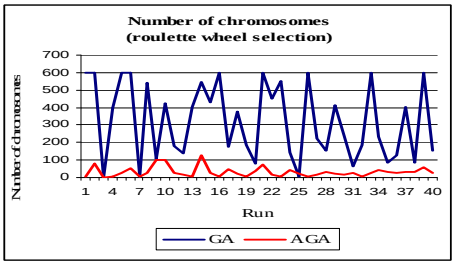

Figure 4.b: Number of total chromosomes until solution (GA versus AGA – roulette wheel selection)

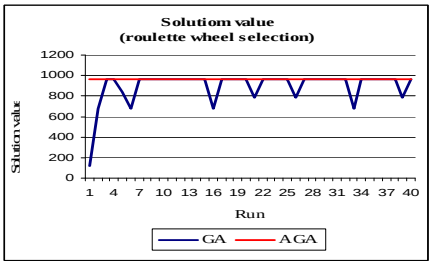

Figure 4.c: Solution value (GA versus AGA – roulette wheel selection)

Based on these, using the roulette wheel selection method, AGA performs better from all points of view: (i) the average number of generations until solution is 15.8 times higher in case of GA compared to the case of AGA; (ii) the average number of chromosomes until solution is 10 times higher in case of GA compared to the case of AGA; (iii) AGA obtains the correct solution in all cases, but GA obtains the correct solution only in 93% cases.

We made the same performance tests using another selection method – the tournament selection (with tour = 2). The obtained results are presented in Figure 5a, b and c (GA versus AGA): Figure 5.a presents the number of generations until solution; Figure 5.b presents the number of chromosomes until solution; Figure 5.c presents the solution value obtain.

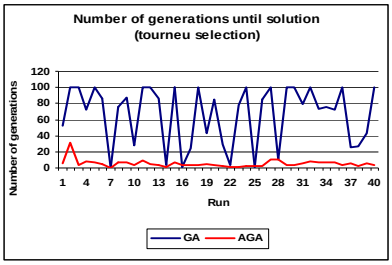

Figure 5.a: Number of generations until solution (GA versus AGA – tournament selection)

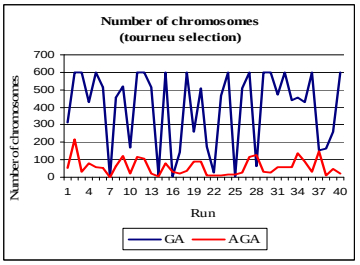

Figure 5.b: Number of total chromosomes until solution (GA versus AGA – tournament selection)

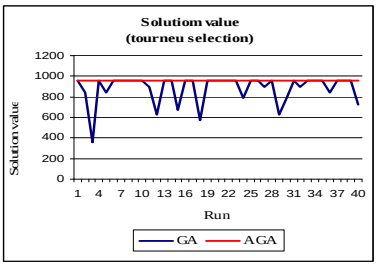

Figure 5.c: Solution value (GA versus AGA – tournament selection)

In this case, using the tournament selection method, AGA performs better from all points of view, too: (i) the average number of generations until solution is 12.2 times higher in case of GA compared to the case of AGA; (ii) the average number of chromosomes until solution is 6.76 times higher in case of AGA compared to the case of GA; (iii) AGA obtains the correct solution in all cases, but GA obtains the correct solution only in 92% cases.

Other tests were made using the truncated selection method (with threshold = 50%). The obtained results are presented in Figure 6a, b and c (GA versus AGA): Figure 6.a presents the number of generations until solution; Figure 6.b presents the number of chromosomes until solution; Figure 6.c presents the solution value obtain.

In this case, using the tournament selection method, AGA performs better from all points of view, too: (i) the average number of generations until solution is 14.1 times higher in case of GA compared to the case of AGA; (ii) the average number of chromosomes until solution is 7.76 times higher in case of AGA compared to the case of GA; (iii) AGA obtains the correct solution in all cases, but GA obtains the correct solution only in 98% cases.

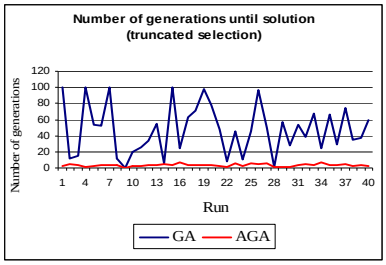

Figure 6.a: Number of generations until solution (GA versus AGA – truncated selection)

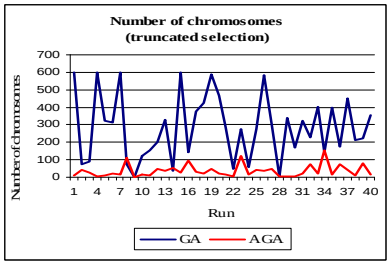

Figure 6.b: Number of total chromosomes until solution (GA versus AGA – truncated selection)

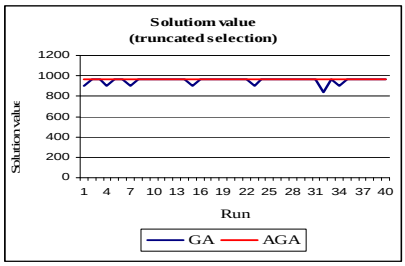

Figure 6.c: Solution value (GA versus AGA – truncated selection)

According to the results presented in Figure 4, 5 and 6 AGA performs better than GA, but the results values depend by the used selection method. Further we compare the average performances of both AGA and GA for all the three used selection methods (roulette wheel, tournament and truncated selection methods). The obtained results are presents in Figure 7a, b and c (GA versus AGA): Figure 7.a presents the average number of generations until solution; Figure 7.b presents the average number of chromosomes until solution; Figure 7.c presents the solution value obtain.

The average results are better in case of AGA than GA. The performance of AGA depends by the used selection method: (i) the average number of generations until solution is approximately the same in case of roulette wheel selection method and truncated selection. In case of tournament selection the average number of generations is 1.59 times higher than in case of roulette wheel selection; (ii) the lowest average number of chromosomes until solution is reached in the case of roulette wheel selection, and the highest average number of chromosomes is in case of tournament selection (which is 1.82 times higher than in case of roulette wheel selection). This value is only 1.14 times higher in case of truncated selection compared to the roulette wheel selection method; (iii) AGA obtains the correct solution in all cases – this is not influenced by the selection method.

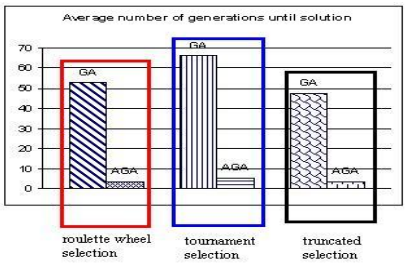

Figure 7.a: Average number of generations until solution (GA versus AGA)

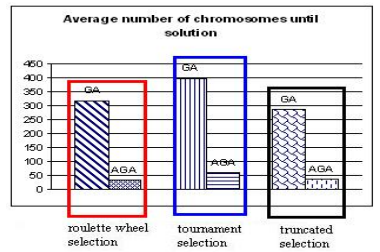

Figure 7.b: Average number of chromosomes until solution (GA versus AGA)

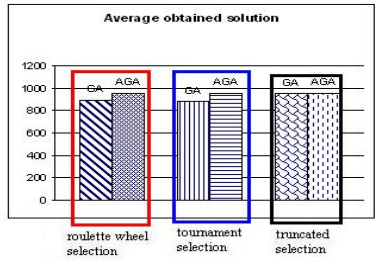

Figure 7.c: Average solution value (GA versus AGA)

B. DEC Evaluation

For this problem we use the following test conditions:

- N is the natural number to be decomposed
- fitness function - N - $x^2+y^2+z^2+u^2+w^2$, where x, y, z, u, w is the current decomposition
- 120 individuals in each population
- maximum number of generations: 500
- 40 test runs.

We evaluate the performance of GA versus AGA for obtaining the solution of this problem. The average number of generations until solution for the three types of selection operator is presented in Table 1. Also Table 1 presents the average number of chromosomes until solution and the average value of the solution.

Table 1: Evaluation of the two algorithms.

Selection	The average number of generations until solution		The average number of chromosomes until solution		The average value of the solution	
	GA	AGA	GA	AGA	GA	AGA
Roulette	475.02	384.075	57003	90204	201	86
Tournament	25.95	5.55	3114	1195.55	22	22
Truncated	3	2.175	360	474.175	22	22

Based on the results from Table 3, the roulette wheel selection operator is not good for solving the DEC problem – the average value of the solution is bigger than N. Though the average obtained solution is lower in the case of AGA compared to the case of GA. The others two operators produce good results. In case of tournament and truncated operators, solution is obtained in a few generations in AGA case compared to the GA case, as in the MAX problem. In the case of tournament operator the average number of generations in 4.67 times higher in case of GA compared to the case of AGA. For truncated selection operator the average number of generations is 1.37 times higher in case of GA compared to the case of AGA. The average number of chromosomes until solution for GA is 2.6 times higher in case of tournament selection operator and 0.75 times higher in case of truncated selection operator compared to the case of AGA with the same selection operator.

I. Image Annotation Evaluation

Based on the tests performed in (Mocanu, 2009) we consider the following test conditions: tournament selection operator; uniform crossover; bit flip mutation from IA_GA (as in section 4); fitness function – the distance between low level features shape and color; 6 individuals in each population; maximum number of generations: 100; 40 test runs for each image.

In the case of image annotation with genetic algorithms we want to associate with each object from an image a representative keyword. Thus we will run the genetic algorithm (IA_GA or IA_AGA) for the same initial image. Finally we will obtain a keyword (image) which will be associated with the initial object/image. In case of image annotation the genetic algorithm IA_GA or IA_AGA will always run a fixed number of generations (they must minimize the fitness function and they cannot stop after some number of generations). Thus we will not compare the number of generations until solution. For testing we use a set of images with objects from the domain of flowers. For each image we run the IA_GA and IA_AGA and we compare the obtained keyword with the component object. The comparison is realised based on the fitness of the obtained solution (which represents the distance of the low level features of the object to be annotated and the annotation). On average the results obtained with the IA_AGA are 5% better than in case of IA_AGA. Thus the annotation with IA_AGA is made more accurately. But from the number of chromosomes until solution, the IA_GA is 2 times higher than IA_AGA. So, the anticipatory genetic algorithm is superior to traditional genetic algorithm only from the obtained solution value.

7. CONCLUSIONS

From the functional point of view the main differences between the traditional and the anticipatory genetic algorithm are represented by the offsprings generation strategy (both crossover offsprings and their mutated clones) and the rejection of not fitted offsprings. Thus the difference between the solution and the evolution threshold is reduced at each new generation.

The performance evaluation performed shows that the anticipatory genetic algorithm is in most cases superior to traditional genetic algorithm from all points of view: number of generations until solutions, number of chromosomes until solution and accuracy points of view. In the case of image annotation, the anticipatory genetic algorithm is superior to traditional genetic algorithm from the obtained solution value. But in this case the genetic algorithm is superior to the anticipatory case from number of chromosomes until solution.

Based on the superior performance of the anticipatory genetic algorithm in case of image annotation (from the point of view of the obtained solution value), in future we want to implement a retrieval system based on the semantic annotations obtained by applying the IA_AGA algorithm.

ACKNOWLEDGMENTS

The work has been co-founded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/89/1.5/S/62557.

REFERENCES

1. Dubois, D.M., 2000. Review of Incursive, Hyperincursive and Anticipatory Systems - Foundation of Anticipation in Electromagnetism. Computing Anticipatory Systems: *CASYS'99. Edited by Daniel M. Dubois, Published by The American Institute of Physics, AIP Conference Proceedings 517*, pp. 3-30.
2. Eakins, J. P., 1996. *Automatic image content retrieval – are we getting anywhere?*, Proceedings of Third International Conference on Electronic Library and Visual Research, (May 1996), *pp. 123-135*
3. Eakins, J. P., 1998. *Techniques for image retrieval,* Library and Information Briefings, Vol. 85, South Bank University, Library Information Technology Centre, London
4. Gen, M., Cheng, R., 1997. *Genetic Algorithms and Engineering Design*, John Wiley & Sons Inc., New York.
5. Genetic Server/Library, 2009. Accessed August 2009. http://www.nd.com/products/genetic/selection.htm
6. Goldberg, D. E., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning, Kluwer Academic Publisher, Boston.
7. Goldberg, D. E., 2002. The Design of Innovation: Lessons from and for Competent Genetic Algorithms, Addison-Wesley, Reading, MA
8. Gudivada, V. N. & Raghavan, V. V., 1995. *Content-based image retrieval systems*, IEEE Computer, Vol. 28, No. 9, pp. 18-22
9. Mitchell, M., 1996, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA
10. Mocanu, I., Negreanu, L. 2009. Semantic Image Analysis, 17th International Conference on Control Systems and Computer Science, (26-29 May, 2009), Bucharest, Romania, ISSN: 2066-4451, *pp. 141-146*
11. Mocanu, I., Kalisz, E., Negreanu, L. 2010. Genetic Algorithms Viewed as Anticipatory Systems, CASYS 2009, Liege, Belgium , AIP Conference Proceedings, Vol. 1303, DOI: 10.1063/1.3527157, *pp. 207-215*