MODELLING THE WORLD WITH STATISTICALLY NEUTRAL PBGAs. ENHANCEMENT AND REAL APPLICATIONS

F. Bellas and R. J. Duro

Grupo de Sistemas Autónomos, Universidade da Coruña, Spain
fran@cdf.udc.es, richard@udc.es

ABSTRACT

This paper is concerned with extending and enhancing the use of promoter genes and introns in the encoding of variable length artificial neural network structures for their evolution. These structures have become necessary due to the requirements imposed by the problem we are tackling, that is, the real time evolution of world and internal models for robots operating in changing environments. Promoter Based Genetic Algorithms (PBGA) contemplate the evolution of the architecture and weight values of artificial neural networks regulating the expression of the different genes in the chromosome in a statistically neutral manner. A non direct genotype-phenotype transformation is thus obtained which becomes very efficient in dynamic environments. We study new features in the algorithm that permit achieving very good solutions in modelling the world for real robot applications without predetermining number of neural nets that will collaborate in order to achieve the world model.

1. INTRODUCTION

Several approaches may be considered in order to obtain world or internal models for artificial organisms operating in complex environments such reinforcement learning, ANNs or evolution. However, certain requirements must be met to make the process worthwhile and these have to do with the capacity of acquiring the models through on-line interaction with the world and of modifying them as necessary when circumstances change without becoming so chaotic that the model only represents the current instant of time. Additionally, if the models are constructed in such a way that they can be taken as aggregations of submodels, they could be reused in gradual learning processes and this would improve efficiency when adapting to new environments. Parts of models that were used in other environments may be useful in the current one. This implies, however that we must constantly maintain populations of possible submodels and/or modules out of which we make up new models through aggregation or slight modification. This is the objective of the Multilevel Darwinist Brain [1],[2] which does this through the parallel evolution of populations of world models, internal models and strategies to fulfill the organisms objectives.

In this paper we consider a very serious problem of standard Genetic/Evolutionary algorithms when applied to these tasks: They tend to converge towards homogeneous populations, that is, populations where all of the individuals are basically the same. In static problems this would not be a drawback if this convergence took place after reaching the optimum. Unfortunately, there is no way to guarantee this, and diversity may be severely reduced long before the global optimum is achieved. Obviously, when the results required would be made easier through the collaboration of different modules, if the population converges to a single type of structure this collaboration becomes impossible.

Three approaches may be considered to try to solve this problem. Two of them are based on exogenous actions over the population and the third one acts over the transcription from genotype to phenotype. In the first group, some authors have resorted to methods where individuals were introduced in the population when necessary in order to generate diversity. These individuals were generated randomly, as in the case of [3], or through some heuristic implying that at certain points of the algorithm a secondary memory would preserve interesting individuals that would be injected in the population as needed [4]. In both cases it is necessary to somehow monitor diversity.

Other researchers have considered restrictive selection mechanisms within the algorithms so as to enforce a given level of genetic diversity in the populations using concepts like similarity between individuals in a given metric and which, as in the previous approach, require the definition of a consistent distance measure in the population [5]. Finally, a third more biologically inspired way of addressing the problem has been to tweak with the representation of the individuals and their genotype-phenotype transformation by selectively expressing genes.

It is interesting to note that despite the fact that nature makes extensive use of gene expression techniques, they have not often been used in the realm of evolutionary algorithms. The situation is changing and we are seeing an increase in the study of these techniques in the last few years [6-8].

There exist two main biologically based approaches to gene expression, diploid representations and promoter based mechanisms. Diploid genotypes are made up of a double chromosome structure where each strand contains information for the same functions. Whenever a phenotype is constructed from the genotype, one of the two possible alleles for each gen is chosen following a dominance mechanism which may change with time. As not all of the genes making up the chromosomes are expressed, and as the fitness of an individual is determined by its phenotype, the recessive genes are shielded from selective pressure thus providing a memory within the encoding of the genotype. These techniques were introduced in computational evolution by Goldberg [9] who claimed that a diploid representation combined with a dominance map can outperform a standard evolutionary algorithm in dynamic problems. Several studies and applications of these mechanisms such as those by Ng and Wong [7] and Ryan [6] may be found in the literature. In this work, however, we will concentrate on the other gene shielding/expression mechanism, that is, the use gene promoters and, consequently, loosely speaking, of introns as unexpressed pieces of genetic code. Before we go into how we implement this approach, let us provide an overview of the biological facts that underpin it.

2. SOME BIOLOGICAL FACTS

In prokaryotes (bacteria and other simple cells) all the DNA coding for a protein is continuous. In more complex, eukaryotic, cells, however, the encoding DNA is generally discontinuous: sequences of encoding DNA (exons) are interspersed with long sequences of non-encoding DNA. These non-encoding DNA sequences, usually about 10-fold longer than the exons, are called introns and even though for a long time they have been considered "junk", the fact that they are so common and have been preserved during evolution leads many researchers to believe that they serve some function.

To control where a protein is encoded, the chromosome contains protein-begin and protein-end signals called codons. A codon is a group of three bases - A, T, C, or G - and codes for a single amino acid. When the machinery of the cells sees that first start codon, it knows that the instructions for making a protein begin at this point. The code is always read in groups of three, so the start codon also gives the cell's machinery it's so-called reading frame. Each set of three letters thereafter corresponds to a single amino acid. A stop codon tells the cell's machinery that it has reached the end of the protein and should stop translating the code.

It must also be considered that almost every cell in an organism has a copy of every single gene the whole organism needs. Different genes are expressed in cells corresponding to different organs. Obviously, one would not want a gene coding for toes to be expressed in the lungs. Gene promoters are in charge of controlling these effects. Gene promoters are important regulatory structures that control the initiation and level of transcription of a gene. They are located before the gene and indicate whether, or to what extent, that gene is turned on or off.

3. SUMMARY OF PREVIOUS RESULTS

In the very brief introduction presented above we have provided an indication of the elements that should go into genotype encoding in order to allow for gen expression. These elements are exons and introns, gene promoters and codons. Using these elements the "cell machinery" that is, the part of the GA in charge of constructing the phenotype will know how to make the final organism from the genotype. If these elements are an intrinsic part of the encoding, they will provide for an unobtrusive way of evolving what is expressed and how, thus making the operation of the algorithm much smoother.

To achieve this objective and taking inspiration from the work on structured genetic algorithms [10] we have considered a GA we have called Promoter Based Genetic Algorithm (PBGA) that evolves variable size feedforward artificial neural networks (ANNs) and was presented in [11]. These neural networks, in our case, are encoded into sequences of genes for constructing a basic ANN unit. Each of these blocks is preceded by a gene promoter acting as an on/off switch that determines if that

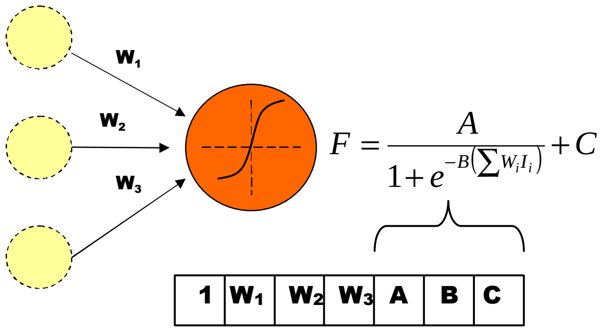

Figure 1. Sigmoid type neuron with inbound connections which constitutes the basic unit. The first field corresponds to the gene promoter value, in this case it is 1 and the neuron will be included in the phenotype.

particular unit will be expressed or not. In order to simplify the algorithm, we have also used the gene promoters as start and end codons due to their position in the chromosome.

As basic unit we have taken a neuron with all of its inbound connections (figure 1). Consequently, the genotype of a basic unit is a set of real valued weights followed by the parameters of the neuron (in this case a traditional sigmoid) and preceded by an integer valued field containing the value for the promoter gene and, consequently, determining the expression of the unit. By concatenating units of this type we can construct the whole network.

The main features of the PBGA could be summarized as follows [14]:

- The genotypes are encodings of ANNs (weights, activations and bias), with codons and promoter genes.
- We evolve the whole structure (genotypic representation) but in order to test the fitness of the ANNs we translate it to a phenotypic representation including only the weights and neurons that must be shown according to the activation value of the promoters.
- We use panmictic reproduction: two parent individuals lead to a single offspring.
- Crossover and mutation must be different depending on the gene they affect. There are different crossover and mutation probabilities for promoters and other types of genes as changing promoters means activating or deactivating all the genes associated to it, and this could be very destructive.

4. MULTI-MODULE MODELS

When seeking models for processes in the world, one would like to obtain them in the form of the most reusable components. This implies that it would be of great interest to be able to autonomously obtain the models by aggregation of individual components that are also obtained automatically. This is the final objective of the PBGA in our application. In the case of real robots, we have very complex models of the world perceived by the robot and we need to simplify them to use the world model in an efficient way. The PBGA should provide the simplest set of models into which the complex one can be decomposed adjusting the number of neurons automatically. In the case of having *n* ouput neurons, we could have a single ANN model with *n* outputs or *n* ANN models each with just one output or

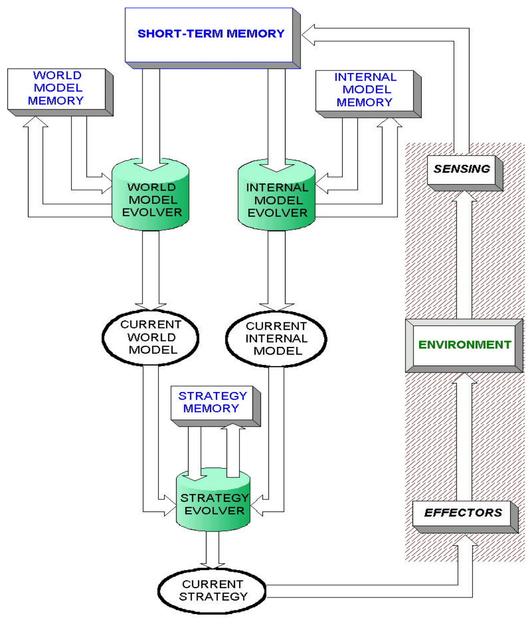

Figure 2. Block diagram of the MDB

any intermediate case. This decision must be made by the PBGA autonomously according to the fitness.

This seems like a problem for multiobjective optimization, but there is a large difference: we don't want *n* solutions that are globally good according to the Pareto front, rather, we require independent solutions living together in different groups that working together produce optimal results for every output independently. The selection schema presented in VEGA [12] works with subpopulations created in the selection process, one for each objective. The problem the authors had in multiobjective applications was the "speciation", where parts of the population develop different features, but this is exactly our goal. So we use a selection procedure that is similar to the one used in VEGA: if we have *n* output neurons, the selection process becomes a set of *n* selection processes each one of them according to the fitness value provided by one of the *n* outputs. We obviously need to carry separate values for the fitness of each output. After each one of these selection processes we perform crossover and mutation. For example, if we use an ANN with 2 output neurons, we run a selection process using the fitness of the first output and apply crossover and mutation. After this, we run a second selection process using the fitness of the second output and repeat the crossover and mutation. Finally, we will have a new population that tends to separate two different groups in evolution but if we are careful with the statistics involved in selection, and they are neutral, that is, they do not induce any bias, the different populations will grow or shrink according to the fitness function and, in the end, we will have a certain group of networks that will cover all of the outputs together, without predetermining how many outputs a network may cover. In the two output problem it would be possible for the best solution to correspond to a single ANN with the two outputs enabled.

5. REAL ROBOT RESULTS

The PBGA was designed to work in a complex cognitive mechanism for autonomous robots presented in [1] and [2] called

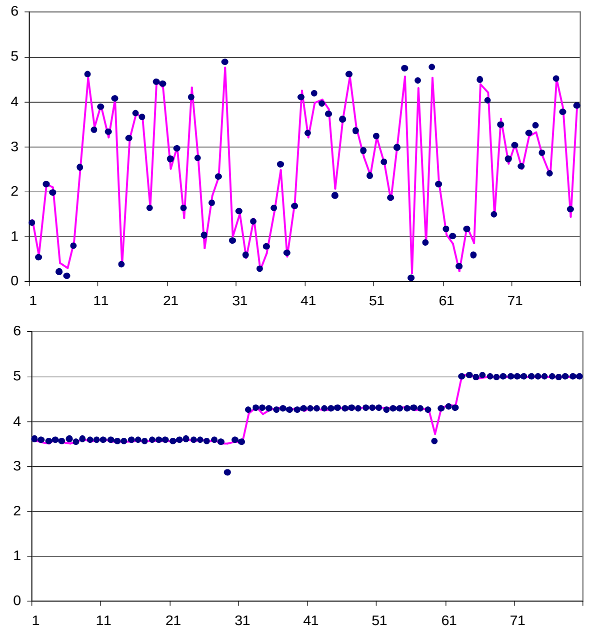

Figure 3. Real data obtained from the Hermes problem (points) and the prediction made by the neural networks resulting from the PBGA (solid line) for the first output (top) and for the second output (bottom).

multilevel Darwinist Brain (MDB). Starting from the knowledge of the environment a robot has through its sensors, we try to obtain a model of the world in which the robots lives. The world model can be very complex and using the PBGA we will try to decompose it into simple submodels. In order to test the PBGA with real functions, we have used two sets of values, one obtained from the IR sensors in a Hermes robot and the other one obtained from the sonar sensors of a Pioner II robot.

A Multilevel Darwinist Brain (MDB), which is extensively explained in [1] and [2] is a mechanism that allows an organism

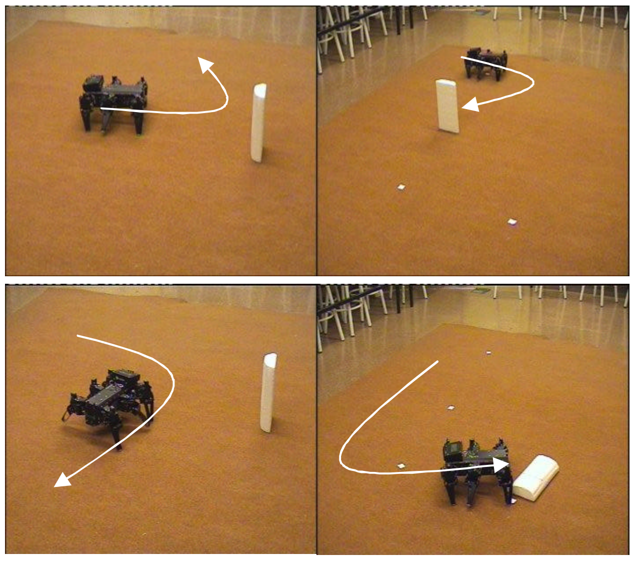

Figure 4. Hermes II robot turning to reach an objective. Left column: robot in the first iterations with the world. Right column: robot after two hundred iterations with the world.

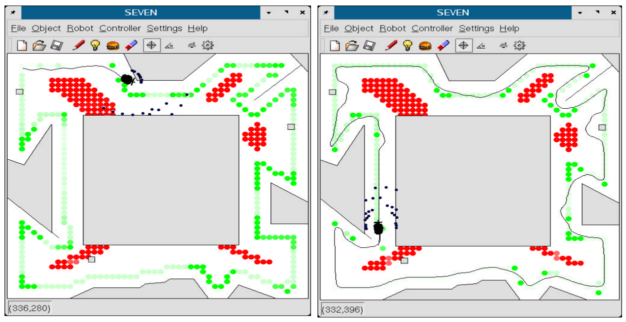
SEVEN
File Object Robot Controller Settings Help
(336,280)
SEVEN
File Object Robot Controller Settings Help
(332,396)

Figure 5. Two shots of the simulated environment in which the Pioneer 2 robot is following walls.

to interact with the environment and learn from it so that it can improve its performance in time without the designer predetermining future behaviours. To this end we have considered several elements [1] like *Strategies*, *World Models*, *Internal Models and* A*ction-Perception Pairs* connected as shown in the block diagram of figure 2, which represents the whole mechanism. The final objective of the mechanism is to obtain the S*trategy* the agent must execute in the real world to satisfy its motivations. This selected S*trategy* is represented in the block diagram of figure 2 by the block labeled *Current Strategy* and is applied to the *Environment* through the *Effectors* providing new *Sensing* values for the agent. Thus, after each interaction with the world, we have an A*ction-Perception Pair* obtained from the real world. These action-perception pairs are stored in the *Short-Term Memory* and are used as the fitness function for the evolution of the *World* and *Internal Models*. These concurrent evolutionary processes provide the current *World* and *Internal Models* which are used to evaluate possible strategies in a third evolutionary process that generates ever-improving strategies until the best one obtained within the time constraints of the robot in its environment becomes the current strategy and is applied on the world, generating new information. This basic cycle is repeated and, as time progresses, the models become better adapted to the real world and the selected strategies improve in their efficiency to fulfil the agent's motivations.

It is evident from the way the mechanism operates that it is necessary to be able to obtain very good W*orld Models* for it to work at its optimum. This is where PBGAs come in. Using the implementation of PBGAs mentioned above, we are able to obtain W*orld Models* made up of autonomously selected submodels working together and, consequently, the MDB can be efficient in its operation. In what follows we present some results of the application to two real robots, a Hermes II and a Pioneer 2. The idea is to obtain, on line, the best possible W*orld Models* each instant of time so that the actions the robot selects during its thinking phase lead it to the desired objective when in the execution phase.

The Hermes II robot is a hexapod robot provided with six infrared sensors placed on the top of each leg. We applied the MDB trying to teach the robot to turn right and left avoiding obstacles. Some results using a traditional genetic algorithm were presented in [2] and were very promising. The robot learned to avoid the obstacles by itself using the information obtained from its environment. The problem was that the world model obtained was a single complex neural network, which was not divided into sub-modules. Using the strategy in that case it would be very difficult to scale up to really complex worlds. We

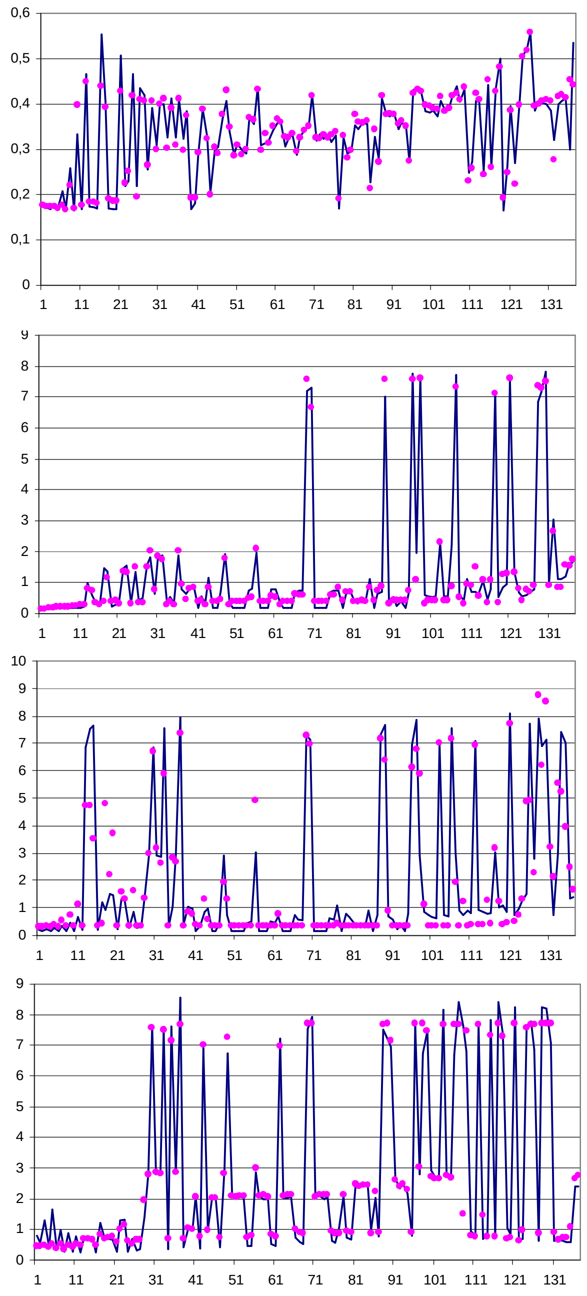

Figure 6. Comparison between data obtained from the real world (solid line) and the prediction made by the networks obtained by the PBGA (dots) for each output of the pioneer example.

are going to consider the same example, but now using the PBGA. We take 3 inputs (distance, angle and the amount of turn) and 2 outputs (distance and angle). As we are talking about a world model, it should predict a new state from the previous state and the action performed (in this case a turn at a given speed). In theory we could obtain with the same probability two different sub-models one for each output or just one that contemplates both outputs, like the one in [2]. We start the PBGA imposing the maximum size of the ANN (2 layers and 8 neurons per layer) and we leave the algorithm to find the appropriate number of active neurons for each layer.

In figure 3 we display a comparison between the real data obtained from the Hermes problem and the prediction made by the best ANN obtained through PBGAs for the first output (top image), and the second output (bottom image). The results are very good and the PBGA obtains two different ANNs that make up the model, one for each output. The first one (distance) is a 2 layer ANN with 5 neurons per layer and the second one is a 2 layer ANN with 6 neurons in the first layer and 7 in the second.
In the bottom image we can see that point 30 in the distance prediction fails but all the rest are good, and the angle prediction (top image) is good everywhere. This means that with minimum intervention of the designer, the PBGA provides the world model divided into two simple parts, in this case two ANNs which were automatically chosen with the appropriate sizes.

Figure 4 shows some images of the robot acting on a world model of this type and performing turns that lead it to the objective. On the left side we display some results corresponding to the first iterations of the mechanism, where very little evolution has taken place. The results are not good, obviously. On the right side we display results after a longer period of interactions with the world. The world model now is a lot better and the robot reaches its objective every time.

The second example we considered corresponds to the Pioneer 2 robot, which is a robust wheeled robot. It has 16 sonar sensors all around its body. In this example we use the robot performing a wall-following task (figure 5). The ANNs have 6 inputs (4 virtual sensor values obtained from the 8 frontal sensors, the linear speed and the angular speed of the wheels) and 4 outputs (the predicted virtual sensor values).This is not an easy set of sensors to model as can be seen in figure 6. Every time the robot moves following a wall, depending on the features of the environment (corners, straight segments, etc.), very brusque changes in sensor values can take place, and this makes them very hard to predict, especially if one uses a single ANN as a model.

In figure 6 we display the results obtained by the networks resulting from the PBGA in this example. As we can see, the outputs are well predicted again using, in this case, 4 different ANNs: a 6-2-2-1 ANN for the first output, a 6-3-2-1 ANN for the second output, a 6-5-2-1 ANN for the third output and finally for the fourth output a 6-4-1-1 ANN. This means that in the PBGA four types of individuals coexist and evolve together.

6. CONCLUSIONS

In this work we have presented the advantages of using a statistically neutral promoter based genetic algorithm (PBGA) when we work in complex environments with real robots. This algorithm is characterized by the fact that there is a transcription between genotype and phenotype regulated by a set of promoter genes which determine if a given gene is expressed or not. In addition, we use a subpopulation based selection mechanism in order to autonomously obtain simple submodels from complex models. This is vey useful in real environments and real robots where the world models are complex and changing functions. The PBGA has been successfully tested using a cognitive mechanism (MDB) in two real robots.

7. ACKNOWLEDGEMENTS

This work was supported by the MCYT of Spain (Pr. N. TIC2000-0739C0404 and REN2000-0204-P4-0).

8. REFERENCES

[1] F. Bellas, J.A. Becerra, R. J. Duro, *Using evolution for thinking and deciding*, Proceedings WSES2001, pp 6161-6166, ISBN 960-8052-25-4, Tenerife, Spain.

[2] F. Bellas, A. Lamas, R.J. Duro, *Multilevel Darwinist Brain and Autonomously Learning to Walk*, Proceedings CIRAS2001, pp 392-398, ISSN 0219-6131, Singapur.

[3] J. Grefenstette, *Genetic algorithms for changing environments*, In Mnner, R. and Manderick, B. (editors), Parallel Problem Solving from Nature 2, pp 137-144, Amsterdam: North Holland. 1992.

[4] P. Hartono and S. Hashimoto, *Migrational GA that preserves Solutions in Non-Static Optimization Problems*, Proceedings of IEEE-SMC 2001, pp 255-260.

[5] J. Kubalik, L. J. M. Rothkrantz, *Genetic Algorithm with Limited Convergence*, Proceedings 6th Joint Conference on Information Sciences 2002, pp 610-614.

[6] C. Ryan, J. J. Collins, *Polygenic Inheritance - A Haploid Scheme that Can Outperform Diploidy*. PPSN 1998, pp 178-187.

[7] K. P. Ng and K. C. Wong, *A new diploid sceme and dominance change mechanism for nonstationary function optimisation*, Proceedings of the Sixth International Conference on Genetic Algorithms 1995, pp 159-166.

[8] H. Kargupta and K. Sarkar, Function Induction, *Gene Expression, and Evolutionary Representation Construction*, Proceedings of the Genetic and Evolutionary Computation Conference. vol 1, pp 313-320, 1999.

[9] D. E. Goldberg and R. E. Smith, *Nonstationary function optimization using genetic algorithms with dominance and diploidy*, J. J. Grefenstette, editor, Second International Conference on Genetic Algorithms, pp 59-68, 1987.

[10] D. Dasgupta and D. R. McGregor. *Designing application-specific neural networks using the structured genetic algorithm.* In COGANN-92 Proceedings of the International Workshop on Combinations of Genetic Algorithms and Neural Networks, Baltimore, MD, pp 87-96, 1992.

[11] F. Bellas, R.J. Duro, *Statistically neutral promoter based GA for evolution with dynamic fitness functions*, Accepted for publication IASTED-AIA 2002.

[12] J.D. Schaffer *Multiple objective optimization with vector evaluated genetic algorithms*, Genetic Algorithms and their Applications: Proceedings on the First International Conference on Genetic Algorithms, pp 93-100, Lawrence Erlbaum, 1985.