Chapter 6. Neuro-Genetic Information Processing for Optimisation and Adaptation in Intelligent Systems

Michael Watts and Nikola Kasabov
Department of Information Science
University of Otago
P.O. Box 56, Dunedin, New Zealand
mjwatts@sol.otago.ac.nz, nkasabov@comerce.otago.ac.nz

Abstract. This chapter describes the intersection of two areas of artificial intelligence research, genetic algorithms and neural networks. The chapter has six main sections. The first describes the motivation for this research, which is followed by a gentle introduction to the basic principles of genetic algorithms. The third section deals with the application of genetic algorithms to conventional neural networks, while the fourth continues this into neurofuzzy systems. The fifth section describes some advanced neurogenetic systems and suggests a new model for this. Finally, the conclusion reviews the preceding sections.

1. Introduction

Genetic algorithms (GAs) have long been known for their ability to find near optimal solutions in large search spaces. Neural networks are also well established, due to their ability to learn by example and generalise their behaviour to new data. However, the performance of a neural network is often heavily influenced by factors such as its architecture and training parameters, which are usually set by an expert when the network is created. Much work in recent years has focussed on combining the advantages of GAs and neural networks, to create a system where the problems of neural networks can be addressed by the application of GAs. This chapter presents some examples of this work, and suggests some methods to be investigated now and in the future.

2. Genetic Algorithms - A Brief Introduction

Genetic algorithms (GAs) were first extensively described by John Holland in (Holland, 1975). They were later developed by several people, including David Goldberg (Goldberg, 1989), and have since been extensively investigated for a variety of application areas, including engineering, art, and artificial intelligence. GAs are biologically inspired search algorithms characterised by the following qualities:

- They consist of Populations of Chromosomes, with each chromosome being an encoded attempt at a problem solution.

- Each chromosome consists of a string of Genes, with each gene representing a parameter of the encoded problem solution.
- Promising chromosomes are used, through the use of genetic operators such as crossover and mutation, to create new chromosomes, causing each successive population to move closer to the optimal solution in a process similar to biological evolution.

GAs possess the following advantages over conventional, iterative searches:

- They are efficient, possessing the ability to solve certain combinatorial problems many orders of magnitude faster than iterative searches;
- They are problem independent, caring nothing of the problem being solved, asking only that an attempt at a solution be rated according to how well it solves the problem;
- They are implicitly parallel, investigating many alternative solutions simultaneously.

An attempt at solving a problem by GA commonly proceeds along the following seven steps (Goldberg, 1989):

1. Select an encoding schema
2. Randomly initialise a population of individuals
3. Evaluate each individual in the population
4. Select fit individuals to breed a new population
5. Create a new, child population from the parent breeding population
6. Replace some or all of the parent population with the child population
7. Repeat steps 3 - 6 until a terminating condition is reached

Each of these steps are described below.

Selection of an encoding schema for GAs has been, and still is, a matter of some debate. The two competing schools of thought are known as the Principle of Minimal Alphabets and the Principle of Meaningful Alphabets. Briefly, the Principle of Minimal alphabets states that the representation used should use the smallest possible number of characters. This is somewhat supported by nature, which uses only a four character alphabet for DNA. The Principle of Meaningful Alphabets states that the characters used should be directly relevant to the problem being solved. This tends to lead to shorter chromosomes, but complicates operations such as mutation.

Initialisation of a population of individuals is generally a simple operation. It may sometimes be necessary to insert individuals whose gene values have been set externally to the GA into the initial, random population, as a means of incorporating problem specific knowledge into the GA and assisting it’s search. While the initialisation of the population may not be a difficult problem, the selection of the size of the population may be more problematic, and is often a matter of compromise. Small population will be faster to process, due to the smaller number of individuals to be evaluated and reproduced each generation. However, small populations run the risk of stagnating genetically, i.e. the population does not contain sufficient genetic diversity to properly explore the search space. This problem is known as *premature convergence* and is similar to the effects of inbreeding in biological populations.

Evaluating each individual within a population involves quantifying the fitness of each individual. The method used to do this is highly dependent upon the problem being explored, as the fitness of each individual depends upon how well it solves the problem. The only requirements for the evaluation of each individual is that the fitness value assigned accurately reflect the degree to which the individual solves the problem, and that the values be calculated in a consistent manner across the generations. No matter how the fitness is assigned, some form of normalisation of fitness values is usually required, so that the genes of early ‘super fit’ individuals do not overwhelm those of their brethren.

Selection involves choosing individuals to form a breeding population. While there are many methods of selection in use, they are all related in using the fitness of the individual as the basis of their operation. With roulette wheel selection, for example, each individual has a slice of a virtual roulette wheel assigned to it, with the size of the slice determined by the individual’s relative fitness. Every time an individual is required, the wheel is spun and the individual it lands on is selected. With tournament selection, two individuals are selected by roulette wheel selection and their fitness values compared, the one with the higher fitness being the one selected.

Creating a new population from a breeding population involves two operators, crossover and mutation. Crossover involves two chromosomes joining at one or more points and exchanging genes. One method of crossover is one-point, where the chromosomes join at one-point and exchange genes before and after the crossover point. An example of crossover is shown in fig. 1. Here the crossover occurs at the first locus. Another method is uniform crossover, where a ‘coin toss’ is performed at each locus, the result of the coin toss determining whether or not an exchange of genes takes place at that locus. An example of uniform crossover is shown in fig. 2. Here crossover is determined to occur at the second and third loci.

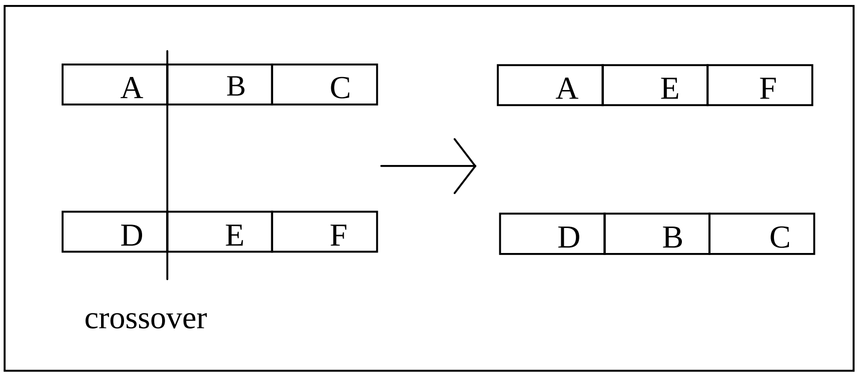

Figure 1: Example of One point crossover between two chromosomes

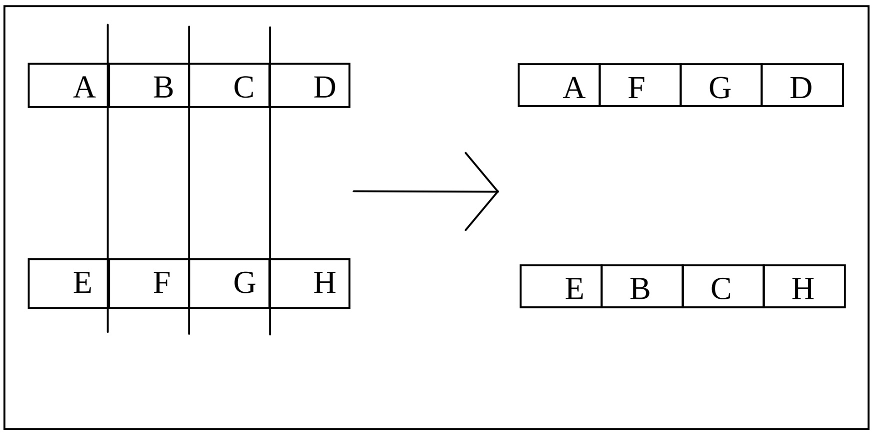

Figure 2: Example of uniform crossover

The exact crossover technique used may have a significant effect upon the performance of the GA. For some problems some alleles may only realise their full effect when in the company of other alleles. Too high a number of crossover points runs the risk of separating these alleles and hence degrading the performance of future generations. Too few crossover points may be detrimental when the loci in question are greatly separated on the chromosome. Mutation is given a high degree of importance by biologists as “the original source of genetic variation” (Campbell, 1996, pg. 424). While many GA researchers contend that it is not as important as the biologists believe (Goldberg, 1989, pg. 14), the principle of mutation remains unchanged: A gene is randomly selected with a certain probability, and its value is altered. There are two methods used to make this change. In the first the value of the gene is changed to an entirely new value. This method is less effective with low order encoding schema, as there is a higher chance of the new value being identical to the old. In the second method the existing value is altered. This method is more effective for low order encoding schema, as there is no chance of the mutation resulting in the same value. The rate of mutation is also much debated. Too high a mutation rate may cause the GA to degenerate to a random search. Too low a rate of mutation may not allow the GA to escape from a premature convergence trap.

There are many strategies for the replacement of the parent population by the child population. Some simply replace the entire parent population with the children, while others replace parents only if the children are fit. The rationale behind most replacement strategies is to purge the weaker genes while preserving the stronger genes, although some strategies also maintain a small number of less fit individuals in an attempt to preserve some genetic diversity and hence avoid premature convergence.

One tactic that is often added to a selection strategy is elitism. In elitism, the fittest individual passes unchanged from the parent population to the child. This is to ensure that the fittest genes are not lost to the next generation, which is a possibility with a stochastic system like a GA.

Selection of the initial parameters for a GA is a problem. One of the disadvantages of GAs is their sensitivity to the initial parameters used to start the search. As mentioned above, parameters such as population size, mutation rate and the crossover technique used, all have a significant effect on the search ability of the GA. Some attempts to find the optimal combination of parameters have used combinatorial approaches, while others have used a GA to find the optimal parameters for another GA (Freisleben and Härtfelder, 1993).

Random initialisation of a GA population may not always be suitable. If some knowledge exists about the problem solution, then it may be more effective to incorporate this information into the initial population. One way in which this can be done is encoding some possible solutions into a chromosome structure and inserting them into the initial population. This approach was used in solving the Travelling Salesman Problem in (Grefenstette, 1987) and proved to be effective at improving the initial search and increasing the efficiency of the search. Another widely used method for incorporating heuristic knowledge into GAs is constraining the range of values a particular gene may take. This is useful for directing a particular parameter of the GA towards a specific area of the search space.

3. Neurogenetic Systems

3.1. ANN Topology Determination by GA

The number of layers within a MLP and the number of nodes within each layer can often have a significant effect upon the performance of the network. Too many nodes in the hidden layers of the network may cause the network to overfit the training data, while too few may reduce it’s ability to generalise. Selection of the optimal number of layers and nodes is a difficult problem that is widely solved by a ‘rule of thumb’ approach, the effectiveness of which is directly proportional to the experience of the network designer. In recent years many researchers have investigated using GAs for solving this problem.

In (Arena, 1993) each chromosome was treated as a two dimensional matrix, with each cell row representing the presence or absence of one node of the network. During evaluation, the number of nodes present in each layer was counted and an appropriate MLP created. If no nodes were present in a row, then that layer was not included in the phenotype. Crossover was implemented as the exchange of a sub matrix from each parent. This method was tested on the optimisation of a complex non-linear system and was shown to be effective at determining a near optimal topology for the MLP.

A different approach was taken in (Schiffman, Joost and Werner, 1993). Here the length of the chromosome determined the number of nodes present, as well as the connectivity of the nodes. This approach to ANN design was tested on a medical classification problem, that of identifying thyroid disorders and provided networks that were both smaller and more accurate than manually designed ANNs were.

3.2. Selection of Control Parameters by GA

In addition to the selection of an ANN’s topology, it is also possible to select the training parameters for the network. This has been investigated in e.g. (Choi and Bluff, 1995).

In (Choi and Bluff, 1995) a GA was used to select the training parameters for the backpropagation training of an MLP. In this work the chromosome encoded the learning rate, momentum, sigmoid parameter and number of training epochs to be used for the backpropagation training of the network. This technique was tested with several different data sets, including bottle classification data, where a glass bottle is classified as either being suitable for reuse or suitable for recycling and breast cancer data, which classifies tissue samples as malignant or benign. For each of the test data sets, the genetically derived network outperformed those networks whose control parameters were manually set, often by a significant margin.

3.3. Training of ANNs via GA

For some problems, it is actually more efficient to abandon the more conventional training algorithms, such as backpropagation, and train the ANN via GA. For some problems GA training may be the only way in which to guarantee convergence.

GA based training of ANNs has been extensively investigated by Hugo de Garis (de Garis, 1990, de Garis, 1992). The networks used in these experiments were not of the MLP variety, but were instead fully self connected, synchronous networks. These “GenNets” were used to attempt tasks that were time dependent, such as controlling the walking action of a pair of stick legs. With this problem the inputs to the network were the current angle and angular velocity of the “hip’ and “knee” joints of each leg. The outputs were the future angular acceleration of each of the joints. Both the inputs and outputs for this problem were time dependent. Conventional training methods proved to be incapable of solving this problem, while GenNets solved it very easily. Other applications of GenNets involved creating “production rule” GenNets, that duplicated the function of a production rule system. These were then inserted into a simulated artificial insect and used to process inputs from sensor GenNets. The outputs of the production rule GenNets sent signals to other GenNets to execute various actions, i.e. eat, flee, mate etc.

A similarly structured recurrent network was used in (Fukuda, Komata and Arakawa, 1997), to attempt a similar problem. The application area in this research was using the genetically trained network to control a physical biped robot. The results gained from this approach were quite impressive. Not only was the robot able to walk along flat and sloped surfaces, it was able to generalise it’s behaviour to deal with surfaces it had not encountered in training. Comparison of the results gained from the genetically trained network to those trained by other methods showed that not only did the genetically trained network train more efficiently than the others, it was also able to perform much better than the others.

4. Neurofuzzy Genetic Systems

4.1. The FuNN Architecture

The FuNN fuzzy neural network is a very flexible model (Kasabov, 1997a). It is essentially a fuzzy system that is trained and handled like a neural network, having nodes and weights that represent input and output variables, fuzzy membership functions, and fuzzy rules.

Tuning the membership functions in FuNNs is a technique intended to slightly improve an already trained network. By slightly shifting the centres of the MF the overall performance of the network can be improved. However, because of the number of MFs in even a moderately sized network, and the degree of variation in the magnitude of the changes each MF may require, a GA is the most efficient means of achieving the optimisation.

Much of the flexibility of the FuNN model is due to the large number of design parameters available in creating a FuNN. Each input and output may have an arbitrary number of membership functions attached. The number of rule nodes may vary considerably. Also, not all of the inputs used for the network may be necessary. This is especially true for networks with a large number of inputs, for example those used for speech recognition systems (Kasabov, 1997b). The number of combinations these options yield is huge, making it quite impractical to search for the optimal configuration of the FuNN combinatorially. Using a GA is one method of solving this difficult problem and is the one that will be concentrated upon here.

4.2. Optimisation Of FuNN Membership Functions By GA

Optimisation of FuNN MFs involves applying small deltas to each of the input and output membership functions. Optimisation of conventional fuzzy systems by encoding these deltas into a GA structure has been investigated in (Gan, Lan and Zhang, 1995) and has been shown to be more effective than manual tuning. Applying these same techniques to tuning FuNN MFs can be seen as a logical extension of this previous work.

The initial GA population is randomly initialised, except for one chromosome, which has all of the encoded delta values set to zero, to represent the initial network. This, along with elitism, ensures that the network can only either improve in performance or stay the same, never degrade in performance. To evaluate each individual, the encoded delta values are added to the centre of each membership function and the recall error over the training data sets is calculated. In situations where a small number of examples of one class could be overwhelmed by large numbers of other classes, the average recall error is taken over several data sets, with each data set containing examples from one class. Fitness of the individual is calculated by the following formula:

$$f = \frac{1}{e_r}$$

where $f =$ the fitness of the individual, and $e_r =$ the average overall recall error of the test data sets.

The above methodology was used to optimise a FuNN that had been trained for single phoneme recognition. The data was taken from the Otago Speech Corpus of New Zealand English. (Sinclair and Watson, 1995). The data used was taken from one male and one female speaker and consisted of seventy-eight melscale vectors, representing three timesteps of twenty-six vectors. The target phoneme was /p/. The bootstrapped backpropagation training algorithm was used for training the network. Bootstrap training is useful in situations where there are large amounts of data for some classes but relatively few examples of others. With Bootstrapped training, each class is dipped into at regular intervals and a new training set built with examples taken from each class in the specified proportions. The network is then trained with ordinary backpropagation training on the generated training set. In the case of this experiment, there were two classes of data, one positive, the other negative and the training set was rebuilt every ten epochs. The network was trained for a total of 1000 epochs, with the learning rate and momentum both set at 0.5. The FuNN was then submitted to the GA optimiser. The parameters for the GA were a population size of one hundred, with one point crossover and tournament selection being used and a mutation rate of one in one thousand. The GA was run for one hundred generations, at the end of which the resulting FuNN was tested and the results compared with those of the original network. These results are tabulated below in table one.

Table 1: comparison of recall accuracy of a FuNN before and after optimisation of MF by GA

	Before Optimisation	After Optimisation
Positive Examples Correct (%)	36.84211	85.96491
Negative Examples Correct (%)	99.83667	94.40588
Overall Examples Correct (%)	99.11201	94.30878

As can be seen from the table, the network's ability to recognise the target phoneme has increased dramatically, while it's ability to reject the other phonemes, and hence it's overall performance, has only marginally decreased.

4.3. Design Of FuNNs By GA

Given a set of training and recall data, it is a relatively simple matter for a GA to evolve an optimal topology for a FuNN, including those features selected as most significant. Encoded within each chromosome is a sequence of values that indicate whether or not a particular input feature is to be used by the network. Also encoded in the chromosome is a sequence of values, one for each input, which describes the number of membership functions to attach to each input. A similar sequence of values describes the number of membership functions attached to the outputs, and the number of rule nodes in the structure. To evaluate each individual, the chromosome is decoded into a FuNN structure and the resulting FuNN trained for a specific number of epochs with the training data set. When training is complete, the recall error over the recall dataset is calculated and the fitness of the individual determined according to the following formula

$$f = w_i \times \tanh\frac{1}{i} + w_t \times \tanh\frac{1}{e_t} + w_r \times \tanh\frac{1}{e_r}$$

where:

$f =$ the fitness of the individual

$w_i =$ the weighting factor applied to the input component of the evaluation function

$i =$ the number of inputs in the network being evaluated

$w_t =$ the weighting factor applied to the training error component of the evaluation function

$e_t =$ the training error of the network

$w_r =$ the weighting factor applied to the recall error component of the evaluation function

$e_r =$ the recall error of the network

The *tanh* function is applied to each component of the evaluation function to give each component the same significance in evaluation. The three weighting factors are adjustable to allow the user to direct the GA towards a particular fitness measure, e.g. to create a network with a minimal number of inputs, rather than good training error, or a network with good generalisation capability and a larger number of inputs.

In the example described here, the above methodology was used to design a FuNN optimised for single phoneme recognition. For this experiment, New Zealand English phoneme data was again used, and the target phoneme was /t/. An initial population size of fifty individuals was used, with a mutation rate of one in one thousand, one point crossover and tournament selection being used. The GA was run over fifty generations. Each individual chromosome was decoded into a FuNN as described above and trained for five epochs on a subset of the total data available, consisting of one hundred examples of the target phoneme and three hundred examples selected from all other phonemes. Recall was over a similarly structured dataset while the learning rate and momentum for the training phase was set to 0.5 each. In this experiment, each of the weighting factors of the evaluation function was set to one, giving each factor equal importance. At the termination of the GA, the most fit individual of the final generation was trained for 1000 epochs using the bootstrapped training algorithm across the entire data set, with the learning rate and momentum again set to 0.5 each and the dataset being rebuilt every ten epochs.

The structure of the network is presented below in table two.

Table 2: structure of the evolved FuNN

Input Nodes	27
Condition Nodes	193 (ranging from two to ten per input)
Rule Nodes	6
Action Nodes	3
Output Nodes	1

In order to compare the performance of the GA designed FuNN against a manually designed FuNN, a network with the architecture shown in table three was created and trained with the same settings across the same dataset.

Table 3: structure of the manually designed FuNN

Input Nodes	78
Condition Nodes	234 (three per input)
Rule Nodes	10
Action Nodes	2
Output Nodes	1

The results of testing these two networks are compared in table four. It can be seen that even though the genetically designed network has far fewer input nodes than the manually designed FuNN, it’s ability to accurately classify both the positive and negative examples of the test data set is superior to that of the manually designed FuNN.

Table 4: comparison of recall accuracy for manually and genetically designed FuNNs

	Manually Designed	Genetically Designed
Positive Examples Correct (%)	70	96
Negative Examples Correct (%)	91	93
Overall Examples Correct (%)	91	96

5. Evolutionary Neurogenetic Systems

5.1. The Brainbuilders at ATR

At the ATR laboratory in Kyoto, Japan, Hugo de Garis is leading an effort to evolve an artificial brain. This research combines genetic algorithms and cellular automata to "grow" simplified artificial neurons (Gers, de Garis and Korkin, 1997) The ultimate goal of the CAM-Brain project is, by the year 2001, to create a billion neuron artificial brain consisting of millions of neural network modules controlling thousands of behaviours. CAM is an abbreviation of Cellular Automata Machine, which refers to the specialised cellular automata hardware used in the experiments. A cellular automaton (CA) (Tofoli and Margolus, 1987) is a simplified simulation of a biological cell, with a finite number of states, whose behaviour is governed by rules that determine it's future state based upon it's current state and the current state of it's neighbours. Cellular automata have been used to model such phenomena as the flocking behaviour of birds and the dynamics of gas molecules. The advantage of CA is their ability to produce seemingly complex dynamic behaviours from a few simple rules.

In the CAM-Brain project, a GA is used to evolve a sequence of control signals, which are used to control the growth of "trails" of cells within a three dimensional space. When trails collide, a synapse is formed. Neural signals are sent down the trails of cells and are processed by the neurons formed at the synapses. Each module is evolved to perform one specific task, with the connectivity of the trails providing the functionality of each module. The modules are then assembled into complete systems.

This is an interesting approach, and early simulation results show that the approach is capable of producing useful modules. However, each module must be evolved to perform a specific task, as it is not possible to create a generic module that is then trained to perform a specific task. Also, should a module's requirements change, an entirely new module must be created, as there is no facility for adaptation. Finally, there is the question of the amount of time it would take to define and assemble millions of modules into a functioning brain. With the development of the CAM-Brain machine (CBM) it is now possible for the Brain Builder group to evolve a module in less than a second. However, the goals, and hence the GA fitness definition, of each module must be defined manually. Clearly, the goals of a million modules would take a long time to define, not to mention integrate into a working brain. These issues must still be addressed by the Brain Builder group before the CAM-Brain system is widely accepted.

5.2. A General Framework of an Evolutionary Neuro-genetic System

The neo-computational models developed and used so far are mainly based on using neurons and connections between them, the neurons being simple processing elements. Interestingly enough, the research in neuro-physiology explores the activation of the brain when performing different tasks and then creates models of it without necessarily accounting for the genetically defined information and behaviour of these parts of the brain. More precise models of the brain and therefore more adequate computational models will be created if the neurons do have memory represented as genetically embodied information. This is the motivation for and the rationale behind the development of the following framework of a neuro-genetic system. The system has the following features:

- neurons are defined by: inputs; outputs; a chromosome of genes; an input function; an activation function; an output function;
- connections between neurons are created dynamically as needed which is based on the genetic information in the chromosomes attached to the neurons;
- the genes may change through experience and self-analysis. The system interacts with the environment, observes its behaviour and changes its genes, if necessary, for a better performance. This is also a process of adaptation of the system to the environment in which it operates
- each neuron contains the same chromosome but may only use some particular genes for its operation, so the neurons may be specialised in that way
- the chromosomes contain all the necessary information for the functioning of the single neurons as well for the functioning of the system as a whole

The approach taken here is based on the assumption that genetic information is incorporated in each individual neuron. This genetic information has the function of constraining and guiding the learning process. The learning process results in knowledge accumulated as long-term memory in the system. Initially the system starts with a set of neurons with few connections between them mainly defined genetically. Gradually the neurons become connected to each other based on the Hebbian learning or on the simultaneous activation of them when the same stimuli are presented to the neural network system. The process of learning is continuous, incremental and goes on as long as new stimuli are entered.

This approach has its biological motivation. As pointed out by Horace Barlow (Barlow, 1994), "the neocortex gives an useful knowledge of the world in two ways: not only does it discover the structure of its world by experience during its lifetime, but it has mechanisms, adapted through the process of genetic selection, the confer skills for doing this…the neocortex acquires knowledge of the world by nature as well as by nurture…"

6. Conclusions And Future Work

The goal of this chapter has been to present some principles and examples of combining the genetic algorithm and neural network paradigms. It has covered the use of GAs to design, train and optimise neural networks, as well as outline an entirely new, biologically motivated approach to neurogenetic computation.

This new approach is based on the following principles:

- two modes of learning
    1. a genetically designed and slowly modified way, and
    2. a fast stimulous-recation based learning
- redundancy of neurons in the neuronal space
- connections are created and removed dynamically as the system learns

The neuro-genetic approach to designing intelligent, adaptive systems would make possible their applications for solving challenging problems that have not been successfully solved so far. Such problems are: the predictive analysis of genome information; creating a general AI learning theory which accommodates both the natural evolution of species and brain development; building brain-like computing systems.

Future research in these areas will likely include further integration of genetic and neural systems, following the principles of artificial life and software agents to create systems that can grow and evolve in an on-line, incremental mode. Such a framework is the ECOS (Evolving Connectionist Systems) framework (Kasabov, 98, this volume). ECOS and its realisation EFuNN need to be optimised and the evolutionary programming paradigm seems to be appropriate for this purpose. Current and previous research into neurogenetic systems has so far only scratched the surface of what is possible by marrying these two powerful techniques.

REFERENCES

1. Amari, S. and Kasabov, N. (1997) eds. *Brain-like Computing and Intelligent Information Systems*, Singapore, Springer Verlag.
2. Arena, P., Caponetto, R., Fortuna, L., Xibilia, M.G.. M.L.P. (1993) *Optimal Topology via Genetic Algorithms*. In Albrecht, R.F., Reeves, C. R., Steele, N. C. (Eds.), *Artificial Neural Nets and Genetic Algorithms*, Spring-Verlag Wien, New York.
3. Barlow, (1994) "What is the computational goal of the Neocortex?". In Koch, C., and Davis, J. *Large Scale Neuronal Theories of the Brain*, MIT Press.
4. Campbell N.A. (1996) *Biology*, Fourth Edition, Benjamin / Cummings.
5. Choi and Bluff (1995) *Genetic Optimisation of Control Parameters of a Neural Network* . In Kasabov, N. and Coghill, G. (Eds.), *Proceedings of ANNES '95*, Los Alamitos, CA: IEEE Computer Society Press.
6. de Garis, H. (1990) *Genetic Programming : Evolution of a Time Dependent Neural Network Module Which Teaches a Pair of Stick Legs to Walk* In *Proceedings of the 9th.European Conf. on Artificial Intelligence*, Stockholm, Sweden
7. de Garis, H. (1992) *Artificial Nervous Systems: The Genetic Programming of Production-Rule-GenNet Circuits. In Proceedings of the International Joint Conference on Neural Networks*, Beijing, China.
8. Freisleben, B. and Härtfelder, M. (1993) *Optimisation of Genetic Algorithms by Genetic Algorithms*. In Albrecht, R.F., Reeves C. R., Steele, N. C. (Eds.), *Artificial Neural Nets and Genetic Algorithms*, Spring-Verlag Wien, New York.

9. Fukuda, T., Komata, Y., and Arakawa, T. (1997) *Recurrent Neural Networks with Self-Adaptive GAs for Biped Locomotion Robot*. In *Proceedings of the International Conference on Neural Networks ICNN'97*. IEEE Press, Houston.
10. Gan, M., Lan, H. and Zhang, L. (1995) *A Genetic-based Method of Generating Fuzzy Rules and Membership Functions by Learning from Examples*. In *Proceedings of International Conference on Neural Information Processing (ICONIP '95)*, Publishing House of Electronics Industry, Beijing, China.
11. Gers, F., de Garis, H., and Korkin, M. (1997) *CoDi-1Bit: A Simplified Cellular Automata Based Neuron Model*. In *AE97, Artificial Evolution Conference,* Nimes, France.
12. Goldberg, D.E., (1989) *Genetic Algorithms in Search, Optimisation and Machine Learning*, Addison-Wesley.
13. Grefenstette, J. J. (1987) *Incorporating Problem Specific Knowledge into Genetic Algorithms*. In Davis, L. (Ed.) *Genetic Algorithms and Simulated Annealing*, Morgan Kaufmann Publishers, Inc., Los Altos, California.
14. Holland, J. H., (1975) *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, Massachusetts.
15. Kasabov, N., Kozma, R., Watts, M. (1997) *Optimisation and adaptation of fuzzy neural networks through genetic algorithms and learning- with-forgetting*. *Information Sciences*.
16. Kasabov, N., Kim, JS, Watts, M. and Gray, A. (1997a) *FuNN/2 - A fuzzy neural network architecture for adaptive learning and knowledge acquisition*. *Information Sciences: Applications*.
17. Kasabov, N., Kozma, R., Kilgour, R., Laws, M., Taylor, J., Watts, M. and Gray, A. (1997b) A *Methodology for Speech Data Analysis and a Framework for Adaptive Speech Recognition Using Fuzzy Neural Networks*. In Kasabov, N., Kozma, R., Ko, K., O'Shea. R., Coghill, G. and Gedeon, T. (Eds.) *Progress in Connectionist-Based Information Systems*, Springer, Singapore.
18. Kasabov, N., Watts, M. (1997) Genetic algorithms for structural optimisation, dynamic adaptation and automated design of fuzzy neural networks. In Proceedings of the International Conference on Neural Networks ICNN'97. IEEE Press, Houston.
19. Schiffman, W., Joost, M. and Werner. R. (1993) Application of Genetic Algorithms to the Construction of Topologies for Multilayer Perceptrons. In Albrecht, R.F., Reeves C. R., Steele, N. C. (Eds.), Artificial Neural Nets and Genetic Algorithms, Spring-Verlag Wien, New York.
20. Sinclair, S., and Watson, C. (1995) The Development of the Otago Speech Database. In Kasabov, N. and Coghill, G. (Eds.), Proceedings of ANNES '95, Los Alamitos, CA: IEEE Computer Society Press.
21. Tofoli, T. and Margolus, N. (1997) Cellular Automata Machines: a New Environment for Modelling. The MIT Press, Cambridge, MA, 1997.