Optimising a Hierarchical Neural Clusterer applied to large Gene Sequence Data Sets

Rod Adams *Member, IEEE,* Neil Davey, Paul Kaye and Wanida Pensuwon

Abstract— **Evolutionary Algorithms have been used to optimise the performance of neural network models before. This paper uses a hybrid approach by permanently attaching a Genetic Algorithm (GA) to a hierarchical clusterer to investigate appropriate parameter values for producing specific tree shaped representations for some gene sequence data. It addresses a particular problem where the size of the data set makes the direct use of a GA too time consuming. We show by using a data set nearly two orders of magnitude smaller in the GA investigation that the results can be usefully translated across to the real, much larger data sets. The data sets in question are gene sequences and the aim of the analysis was to cluster short sub-sequences that could represent binding sites that regulate the expression of genes.**

***Index Terms*— clustering, competitive learning, genetic algorithms, neural tree networks.**

I. Introduction

Evolutionary Algorithms have been used to optimise the performance of neural network models before [1], [2]. This paper uses a hybrid approach by permanently attaching a Genetic Algorithm (GA) to a variant of the standard neural network model of the competitive learning algorithm. The neural model considered here is a dynamic node creating algorithm that attempts to produce a hierarchical classification for the given data set. The GA is used to investigate appropriate parameter values for producing specific tree shaped representations for some gene sequence data. It addresses a particular problem where the size of the data set makes the direct use of a GA problematic due to the time needed to evaluate multiple configurations of the network on the data. We show by using a data set nearly two orders of magnitude smaller in the GA investigation that the results found can be usefully translated across to the real, much larger data sets.

The standard Neural Network Competitive Learning algorithm [3] may be modified by the addition of dynamic node creation and the imposition of a tree structure on the classificatory ordering of the nodes. Two of the earliest and most influential attempts are [4] and [5]. Both papers introduced the concept of dynamic tree growth in response to exposure to the data set. The final tree represented the "natural groupings" of the data, with the top level of the tree grouping data into large clusters and at successive levels down the tree the data being clustered more finely. A comparison and critique of these papers is given in [6] and [7]. The dynamic tree growth modifications to the standard competitive learning algorithm bring two main advantages: the number of clusters that the neural network will identify does not need to be predefined, and the hierarchical tree structure improves the interpretability of the results. In addition, the use of a tree structure allows a more efficient search for the classifying node so increasing the speed of the model. An improved dynamic neural network hierarchical clusterer has been introduced by us in [7], and a more robust, stochastic version, the Stochastic Competitive Evolutionary Neural Tree (SCENT), in [8] and [9].

The stochastic version of the model is able to produce a suitable classification over a large variety of data sets. Changing the parameters associated with the model makes it possible to adjust the type of tree structure produced (for example flatter or deeper) [9]. However the parameters interact with each other in complicated ways so that it is very difficult to select appropriate parameters by hand. By incorporating a Genetic Algorithm with the basic SCENT code to produce a hybrid model, containing the dynamic neural tree and an optimiser, it is possible to search the parameter space in order to meet the requirements for a specific task [1], [2].

In this paper we investigate producing parameters for SCENT that will construct two quite different hierarchical tree clustering for some extremely large data sets. The data in question are a complete collection of small sequence windows taken from some genetic sequence data of up to 120000 base pairs. The direct use of the GA on the data sets was ruled out due to the time needed to construct and evaluate the tree on the data for the large number of iterations required by a GA. The approach considered here was to investigate the parameter settings by running the hybrid model on a similarly constructed collection of small sequence windows taken from a completely different, and much shorter, genetic sequence and then apply these values to the large data sets. It has not previously been established that parameters that work for small data sets would also work for large data sets since the model makes use of frequency based information. However from these experiments we conclude that once the general values of the parameters were found for the small genetic sequence they were sufficiently robust that they could successfully produce the appropriate hierarchical clusterings on the large data sets.

The basic stochastic competitive evolutionary neural tree model is described in Section 2. The experiments

Rod Adams, Neil Davey and Paul Kaye are in the Science and Technology Research Centre, University of Hertfordshire, Hatfield, Herts, AL10 9AB, UK. Emails: r.g.adams, n.davey, p.h.kaye@herts.ac.uk,

Wanida Pensuwon is at the Dept. of Electrical & Electronics Engineering, Ubonratchathani University, Ubonratchathani, 34190, Thailand. Email: wanida.pe@ubu.ac.th

performed and the Genetic Algorithm used are described in Section 3, and the results are reported in Section 4. Finally, some discussion and conclusions are given.

II. The SCENT Algorithm

In SCENT, the tree structure is created dynamically in response to structure in the data set. The neural tree starts with a root node with its *tolerance* (the radius of its classificatory hypersphere) set to the standard deviation of input vectors and its position set to the mean of input vectors. It has 2 randomly positioned children. Each node has two counters, called *inner* and *outer*, which count the number of occasions that a classified input vector is within or outside *tolerance*, respectively. These counters are used to determine whether the tree should grow children or siblings once it has been determined that growth is to be allowed.

A. Top-Level Algorithm

At each input presentation, a recursive search through the tree is made for a winning branch of the tree. Each node on this branch is moved towards the input using the standard competitive neural network update rule [3].

Any winning node is allowed to grow if it satisfies 2 conditions. It should be mature (have existed for an epoch), and the number of times it has won compared to the number of times its parent has won needs to exceed a *threshold*. Since any new growth may get pruned a finite limit is put on the number of times a node attempts growth..

When a node is allowed to grow, if it represents a dense cluster, then its *inner* counter will be greater than its *outer* counter and it creates two children. Otherwise, it produces a sibling node. The process of growth is illustrated in Figure 1.

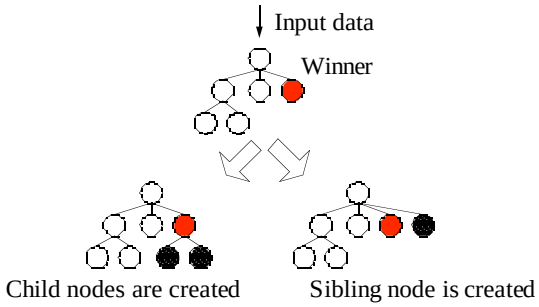
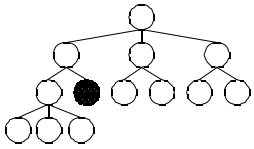
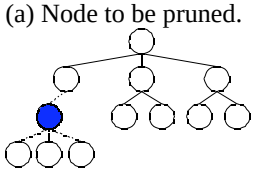


Figure 1. Process of growing a tree. Child node creation is shown on the left whereas sibling node creation is shown on the right.

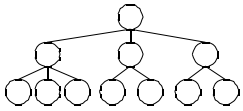
To improve the tree two pruning algorithms, short and long term, are applied to delete the insufficiently useful nodes. The short-term pruning procedure deletes nodes early in their life, if their existence does not improve the classificatory error. The long-term pruning procedure removes a leaf when its *activity* (the rate of classifying input) is not greater than a *threshold*. See Figure 2 for the pruning process. More details may be found in [9].



(a) Node to be pruned.



(b) Singleton is removed, the tree is reconstructed.



(c) Final tree after pruning process.

Figure 2. Pruning process of an inactive node from the tree. The final tree is restructured so that a singleton is removed.

B.Stochasticity

There are two different ways in which stochasticity has been added to the model [9]. Firstly the deterministic decisions relating to growth and pruning have been made probabilistic (we call this Decision Based Stochasticity), and secondly the attributes inherited by nodes when they are created have been calculated with a stochastic element (we call this Generative Stochasticity). To both of these approaches a *simulated annealing* process can be added to mediate the amount of non-determinism in a controlled way, so that a decreasing *temperature* allows for less randomness later in the life of the network.

1) Decision Based Stochasticity

There are three crucial decision making points in the model: the selection for growth, the type of growth and selection for pruning. These decisions are made deterministically in the basic model, a relevant scalar value is calculated and compared to the appropriate *threshold*. Decision Based Stochasticity is generalised in the normal way to a stochastic decision, where the sharp change of decision, depending on some input, is made softer by the addition of some randomness.

Figure 3 illustrates the heaviside *threshold* function softened to a sigmoid. In the deterministic version (on the left) the decision is made at a precise value of the decision variable plotted on the horizontal axis. However, in the stochastic version (on the right) the value obtained by the sigmoid function is compared to a random number between 0 and 1, and if larger, the decision is accepted. In this way values of the decision variable less than the original *threshold* can lead to positive decisions and values greater than the precise one can lead to negative decisions.

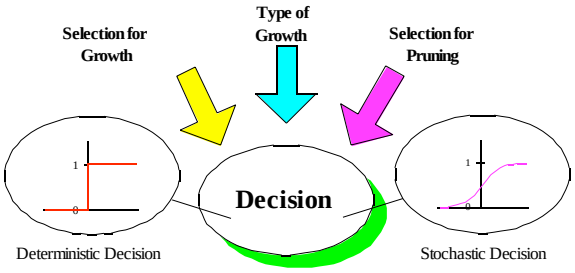


Figure 3. Decision Based Stochasticity. The probability of accepting a decision produced in the left ellipse is crisp whereas the probability of accepting a decision in the right ellipse is fuzzy.

The reason for adding stochasticity is that it may be useful for the network to create more tentative new growth and therefore for the pruning process to be more common. The stochasticity softens the strong decision making and allows the possibility of more chances at growth and of keeping that growth, in the hope that more correct decisions will be made for the different data sets.

2) Generative Stochasticity

This type of stochasticity adds noise to a generated value in the model. The major occurrence of a generated value in SCENT is during sibling creation and child creation.

The key property of a newly created node, calculated from its parent, is its *tolerance* size. Here, some randomness is added to this calculation. To achieve this, a Gaussian centred on the deterministic value gives the probability distribution of the new value. So that in child creation, for instance, the two new nodes can have different *tolerances* based around the original deterministic value. Since the network is sensitive to the value of *tolerance* a stochastic element added here could be beneficial.

3) Control of Stochasticity

The degree of randomness in the stochasticity can be controlled in two different ways. The first method has a *fixed temperature* (degree of randomness) during the whole run whereas for the second method the *temperature* is reduced every epoch by a *temperature* decrease factor. The second method is known as *Simulated Annealing*, as in the standard *simulated annealing* approach. A high *temperature* corresponds to a large amount of randomness, and this is reduced over time. When the *temperature* is reduced to zero, the decision will become deterministic.

An example of SCENT used on a 27 cluster data set is shown in Figure 4.

III. Experiments

A. Genetic Algorithm & Parameter Setting

The behaviour of SCENT is determined by a set of parameters, that specify, for example, the growth and pruning *thresholds* and amount of stochasticity to use. For a large selection of data sets these parameters have been adjusted to produce an acceptable hierarchical clustering without the need to change the parameters. However if a specific type of hierarchy is required (maximising breadth or depth of the tree for instance) or if a difficult data set is encountered, such as an extremely large data set or a high dimensional data set, then a search of the parameter space is required. While most parameters specify the mechanisms that directly affect the growth and pruning of the tree, other parameters specify whether certain parts of the algorithm should execute or not and, if so, by how much. Such parameters control things such as whether to use simulated annealing or not and whether to use stochasticity and, if so, what places in the code to use it. Looked at from a genetic developmental viewpoint with the parameters represented by genes this mimics gene-gene interaction and gene regulation during growth. The parameters therefore interact with each other considerably so that finding suitable parameter values is a non trivial task. Consequently to facilitate the search for appropriate parameter values the permanent addition of a Genetic Algorithm (GA) to the SCENT model was made. Any Evolutionary algorithm would have been suitable and we picked a GA due to our familiarity with it.

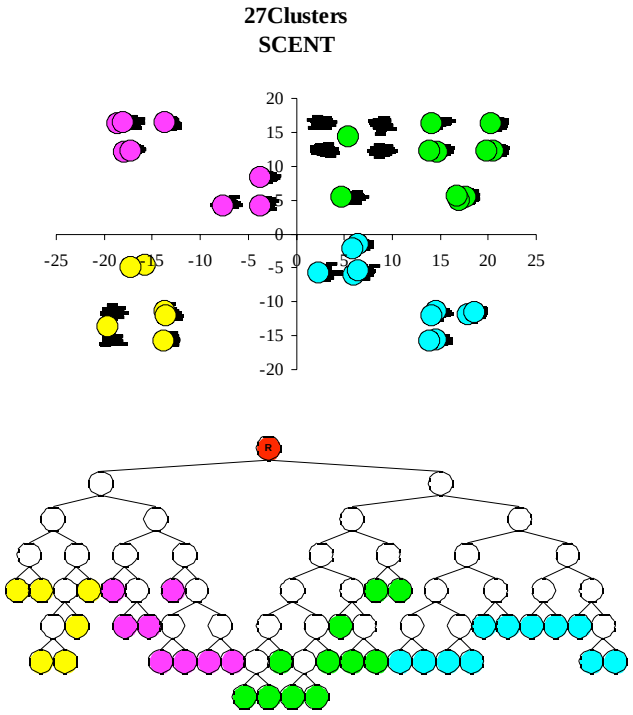


Figure 4. An example of the tree structure produced by SCENT. The original data set and the final leaf nodes are shown in the top half of the figure, the full tree structure is shown in the bottom half of the figure. The leaf nodes are shaded in the tree. The data in each of the 4 quadrants of the top picture is clustered at the second level of the tree, with the left hand sub-tree representing the 3rd quadrant, the next representing the 2nd quadrant, the next representing the 1st quadrant and the right hand sub-tree representing the 4th quadrant.

The GA used is an adapted version of the GENESIS code from John J Grefenstette [10]. The original GA, as is most common, represents parameters in a binary coding. In some circumstances a binary coding is a sufficiently accurate representational method, such as when the parameter being encoded can only take a fixed number of different values. However some of the parameters in our model have a continuous domain and would benefit from a real (floating point) encoding so that we do not arbitrarily restrict their accuracy by imposing a small fixed number of values, as is often done. Both continuous and discrete parameter can be given the most appropriate natural representation using real and binary encoding respectively.

Our GA has been adjusted so that it can cope with a

selection of both binary coded genes and real valued genes at the same time. Both types of parameter are restricted to remain within a fixed range of values, but the real coded ones can take all floating point values within that range. The genes in our GA are on two chromosomes, one binary coded the other real coded. Crossover and mutation on the binary string are 2 point crossover and standard mutation as given in the original GENESIS.

Various methods have been proposed for dealing with crossover and mutation for real valued genes [11]. The methods we use are selected from the ones found in [11]. For the real part of the genome we use 2 point crossover which takes place at two random points. The real chromosome consists of a vector of real numbers. The format for crossover of two parent vectors is:

$$Parent1 = x_1 ... x_i ..... x_j ..... x_n$$
$$Parent2 = y_1 ... y_i ..... y_j ..... y_n$$
$$Child1 = x_1 ... x_{i-1} u_i y_{i+1} ..... y_{j-1} u_j x_{j+1} ..... x_n$$
$$Child2 = y_1 ... y_{i-1} v_i x_{i+1} ..... x_{j-1} v_j y_{j+1} ..... y_n$$

where

$$u_{i,j} = \alpha_{i,j} x_{i,j} + (1 - \alpha_{i,j}) y_{i,j} \text{ and}$$
$$v_{i,j} = (1 - \alpha_{i,j}) x_{i,j} + \alpha_{i,j} y_{i,j} \text{ with}$$
$$0 \le \alpha_{i,j} \le 1$$

That is, the positions of two real numbers are randomly chosen as crossover points. The real numbers are swapped between the crossover points. At the crossover points the two real numbers are mixed using a randomly chosen factor α, in line with binary representation crossover which allows swapping to occur within the boundary of a parameter. Note that this effectively introduces a degree of mutation.

The mutation operator takes one of the real genes and changes it by creating a new value that is calculated by selecting a value from a Gaussian distribution centred on the current value and with variance equal to one quarter of the possible range of values for that gene. If this value is outside the range then it is wrapped round to the other end.

Figure 5 illustrates the process of using a Genetic Algorithm to find appropriate parameter values. Each gene in the two chromosomes is converted into a parameter value and these are used in the SCENT model which generates a tree structure to represent the data set. This is repeated for each member of the population. Fitness is assigned to the resulting tree structures using appropriate methods depending on factors such as its depth or the size of its leaf nodes (see section IIIC). These fitness values are used to create a new population of parameter values using standard GA selection, crossover and mutation operators. This is repeated each time round the loop.

B.Data Sets

The data used were genetic sequences of up to 120000 base pairs thought to contain several genes. The aim of the exercise was to locate binding sites that regulate the expression of the genes [12], [13]. Such binding sites are not guaranteed to be identical sequences. So initially it was necessary to cluster short sequences of the base pairs in order to find closely related patterns that were repeated. Identical patterns would automatically be in the same cluster, but depending on the size of cluster obtained then the nearness of the similarly clustered patterns would be varied. Subsequently these clusters were to be analysed for such things as groups of clustered sequences that were close together in the original full sequence.

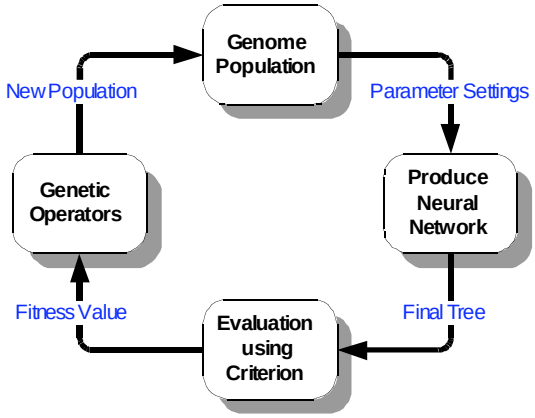


Figure 5. Process of finding appropriate parameter values for the neural clusterer when producing specific tree structures using a Genetic Algorithm.

The nucleotides a, c, g and t s were coded as independent vectors using (1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0) and (0, 0, 0, 1) respectively to maintain their mutual equidistance. Any unknown bases (n s) were coded as (1, 1, 1, 1). The sequences of a, c, g, t and n s that were to be clustered were defined by a variable *window* size, usually between 5 and 15. For a window size of 10, as used in this paper, each vector sequence was therefore of dimension 40. This window was then allowed to slide down the genetic sequence, one nucleotide at a time producing a large set of 40 dimensional vectors. With 120000 base pairs there are 119991 vectors each of arity 40. So the data set consisted of ~4800000 integers. This was nearly a couple of orders of magnitude larger than anything we had tried before. There were two such large data sets, both being Bacterial Artificial Chromosomes (BACs) from different chromosomes of Drosophila melanogaster. The second data set was then halved in size to provide a different sized test data set.

We wanted to produce two types of tree structure. One with leaf nodes as small as possible, so that each leaf node should contain closely related short genetic sequences. The second was a tree with a high degree of hierarchical structure, but still with moderately small clusters. Here selections at different depths in the tree could be used to analyse which sequences were put together at each level.

The largest data set, constructed from the 120000 base pairs sequence, takes more than two days to be clustered by a single run of SCENT. Although the algorithm just scales linearly with the data set size the extremely large size of these data sets means that they are therefore too large to use with a GA, which would require multiple runs, when investigating suitable parameter values. Consequently a shorter sequence was used instead. To attempt to ensure

that the parameters could be suitable for any large gene sequence a totally different short sequence was taken (a gene plus promoter region from the human genome). The sequence used was of length 1519 base pairs, which with a window size of 10 gave 1510 vectors (still of dimension 40). The tree clustering for this small data set could be produced in about one minute.

So the data used was a set of vectors formed by using a 10 nucleotide window from:

a 120000 base pairs gene sequence from Drosophila melanogaster - designated *Sequence1*

a 60000 base pairs gene sequence from Drosophila melanogaster - designated *Sequence2*

a small, 1519 base pairs sequence from the human genome - designated *Small*

The GA was run with a population of 50, a crossover rate of 0.6, a mutation rate of 0.001 (plus that introduced by the crossover operation), a roulette wheel rank based selection mechanism and an elitist replacement strategy. It was thought that the precise values for the parameters would not be significant, since they have appeared robust previously, however having each parameter value in the correct region was significant so we decided that running for 10 generations would be sufficient. When set to run for 10 generations the results were produced in approximately 6 hours.

C. Measurement of Clustering Performances

There were two separate goals in these experiments. In the first experiment small leaf nodes were required so the fitness criteria was that the size of the leaf nodes should be minimised.

In the second experiment a maximum depth was required along with reasonably small clusters. A fitness criteria that took the tree depth and the inverse of leaf size was maximised. The addition of the two factors was weighted by a *weighting_factor* (the value 1/5 being used here) so that they were of more similar importance, with depth remaining as the more significant factor.

$$fitness = depth + \frac{weighting\_factor}{size\_of\_leaf}$$

In each case 3 tree structures were built using the parameter values by using 3 different sets of random initialisation values. The final fitness was averaged from these 3 in order to eliminate random special cases.

IV. Results

A .Experiment 1

In this section, we present the results when trying to produce small leaf sizes, that is small clusters. Interestingly the fitness function used automatically produced a flat clustering with very few nodes below the second level of the tree. Table I shows the results.

As can be seen the parameter values produced leaves of size 3.9 for 1510 sequences with 100% of them at level 1 and 2. In fact 97% of the leaves were at level 1 so the tree was virtually flat. For the large (and real data sets) this translated into trees with leaf nodes of average size of 5.4 and 4.4 for the 120000 sequence and 60000 sequence data sets respectively. The leaf size for the largest data set is surprisingly small and is much smaller than expected prior to running the experiment. Again the hierarchy is mainly flat with over 99% of the nodes at level 1 and 2.

TABLE I: Results for the First Experiment. Here the Criterion was that the Leaf Size should be Minimised. As can be seen the Depth is also Reduced, with Most of the Nodes at Level 1 and 2. The Results have Translated from the Small Data Set to the Large Ones Very Successfully.

Data Set	Small	Sequence1	Sequence2
Size	1510	119991	59991
Nodes	394	30386	16984
Leaves	388	22197	13583
Leaf Size	**3.9**	**5.4**	**4.4**
Top Level	383	13051	9900
Depth	2	4	3
% Level1&2	**100%**	**99.3%**	**99.9%**

B. Experiment 2

In this section, we present the results of using a small data set when trying to produce a deep tree with reasonably small leaves, though not as small as in the first experiment. Table II shows the results.

A maximum depth of 5 was found for the small test data set with only 14% of the nodes at level 1 and 2. The leaf size was 5.2. This translated well into the larger data sets with a depth of 7 being produced with only about 2% of the nodes at level 1 or 2. A reasonable leaf size in the 30s was achieved for the largest data set.

TABLE II: Results for the Second Experiment. Here the Criterion was that the Depth should be Maximised, Together with Reducing the Leaf Size. As can be seen the Number of Levels is Quite High with Most of the Nodes at a Greater Depth than 2. The Leaf Size is Reasonably Small. The Results have Translated from the Small Data Set to the Large Ones Very Successfully.

Data Set	Small	Sequence1	Sequence2
Size	1510	119991	59991
Nodes	428	4550	3694
Leaves	291	3392	2728
Leave Size	5.2	35	22
Top Level	8	8	8
Depth	**5**	**7**	**7**
% Level1&2	**14%**	**1.4%**	**1.6%**

V. Discussion and Conclusion

The addition of the Genetic Algorithm to our previously described hierarchical clusterer has enabled two specific tree structured representations, of some specialised large data sets, to be obtained efficiently. The GA only took on average 6 hours to do 10 generations with a population of 50 each evaluated three times to find its average fitness. A GA run was used to successfully evolve a suitable set of

parameters for each of the types of final tree structure required, which were then used on the larger data sets. Having found the parameter values the large data sets then required a considerable time to find their final clustering (the 120000 base pair data sets took about 2 days to be clustered). These time differences illustrate clearly the difficulty of using the GA on the large data set directly and the advantage of being able to use the parameters determined via the small data set.

The two parameter sets for the two sets of requirements were obviously different. The first set of parameters produced trees with small clusters which also gave a flat clustering with very little child growth and lots of sibling growth, the second set produced deep trees with reasonably small clusters and lots of child growth with little sibling growth. The major differences between the two sets of parameters were those parameters that determined the *tolerance* size for child growth and those that determined if long term pruning should occur. Long term pruning occurs if the new node does not classify enough input vectors.

As far as growth is concerned the second set of parameters produced much larger *tolerances* for children nodes which would then encouraged further child production and deeper trees, as required. Surprisingly there was no major differences in the parameters that determined new *tolerances* formed after sibling growth (the increased *tolerance* was meant to inhibit too much sibling growth). However, the trees produced by the first set of parameters had lots of sibling growth while the other trees did not. It therefore appears that, for these data sets, the main contribution to encouraging sibling growth is just to inhibit too much child growth.

On the pruning side the parameters that determine whether to do short term pruning were similar for both sets (short term pruning occurs if the error is not improved soon after growth). The second set did make it slightly harder to do short term pruning but it was moderated by a stochastic element. However the main difference between the parameters was that for the second set it was very much harder to do long term pruning; the first set of parameters would prune more easily. This was surprising since the first set of parameters were specifically aimed at producing small clusters and therefore did produce quite a lot more nodes that obviously did not get pruned. The second set of parameters did not produce such small clusters despite the difficulty of pruning new nodes based on classification size.

In the two experiments described here the performance of the parameters, found approximately with only 10 generations of a GA and using a small data set from a different genome, were evaluated on much larger data sets. The key result of the paper is that the large data sets performed in a similar manner to the small, test data set. Hence the parameters found by the use of a GA were robust in terms of an increase of scale by 80 times. The two different tree structures were quite different - one was essentially flat and the other was of a considerable depth. This shows that the hybrid SCENT model and GA is capable of producing different sorts of hierarchy to order.

The co-occurrence of short gene sequences, which may represent binding sites for the regulation of genes, is of interest and clustering, as described in this paper, may help to identify such sequences.

REFERENCES

[1] A. G. Rust, R. Adams, S. George & H. Bolouri, "Designing Development Rules for Artificial Evolution", *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, Norwich, April 97, pp509-513, Springer Verlag.

[2] X. Yao, "Evolving Artificial Neural Networks", *Proceedings of the IEEE*, vol. 87 (9), pp. 1423-1447, Sept. 1999.

[3] J. Hertz, A. Krogh. and R. G. Palmer, *An Introduction to the Theory of Neural Computation*, Addision Wesley, USA, 1991.

[4] Li T, Tan Y, Suen S and Fang L, 1992. "A Structurally Adaptive Neural Tree for Recognition of a Large Character Set", *Proceedings of the 11th IAPR International Joint Conference on Pattern Recognition*, pp. 187-190.

[5] Racz J and Klotz T, 1991. "Knowledge Representation by Dynamic Competitive Learning Techniques", *SPIE Applications of Artificial Neural Networks II*, 1469, pp. 778-783.

[6] Butchart K, Davey N and Adams R, 1995. "A Comparative Study of two Self Organising Structurally Adaptive Neural Tree Networks". In: *Neural Networks and their Applications*, Taylor J G (Ed.), John Wiley.

[7] R. Adams. K. Butchart. and N. Davey, "Classification with a Competitive Evolutionary Neural Tree," *Neural Networks*, vol. 12, pp. 541-551, 1999.

[8] N. Davey, R. Adams and S. George, "The Architecture and Performance of a Stochastic Competitive Evolutionary Neural Tree," *Applied Intelligence*, vol. 12, No. 1/2, pp. 75-93, 2000.

[9] W. Pensuwon, *Stochastic Hierarchical Dynamic Neural Networks*, Ph.D. Thesis. University of Hertfordshire, 2001.

[10] J. J. Grefenstette, (1995) Genesis 5.0, ftp://www.aic.nrl.navy.mil/pub/galist/source-code/ga-source

[11] T. Baeck, D. B. Fogel and T. Michalewicz, *Evolutionary Computation 1*, Inst. of Physics, 2000.

[12] B. Alberts et al, *Molecular Biology of the Cell*, Garland Publishing, New York.

[13] J. W. Fickett and W.W. Wasserman, "Discovery and Modelling of Transcriptional Regulatory Regions", *Current Opinion in Biotechnology*, vol. 11, pp. 19-24, 2000