Grammatical Swarm and Particle Swarm Optimization models
applied to Neural Network learning and topology definition

NURIA GÓMEZ, LUIS F. MINGO
JUAN GARITAGOITIA, VICTOR MARTINEZ
Universidad Politécnica de Madrid
Escuela Universitaria de Informática
Ctra. de Valencia km. 7, 28031 Madrid
SPAIN
{ngomez,lfmingo,jmgmartin,victormh}@eui.upm.es

JOSE A. CALVO MANZANO
Universidad Politécnica de Madrid
Facultad de Informática
Boadilla del Monte, 28660 Madrid
SPAIN
jacalvo@fi.upm.es

Abstract: There exists a clear difference between cooperative and competitive strategies. The former ones are based on the swarm colonies, in which all individuals share its knowledge about the goal in order to pass such information to other individuals to get optimum solution. The latter ones are based on genetic models, that is, individuals can die and new individuals are created combining information of alive one; or are based on molecular/celular behaviour passing information from one structure to another. A Grammatical Swarm model is applied to obtain the Neural Network topology of a given problem, training the net with a Particle Swarm algorithm. This paper just shows some ideas in order to obtain an automatic way to define the most suitable neural network topology for a given patter set.

Key-Words: Social Intelligence, Neural Networks, Grammatical Swarm, Particle Swarm Optimization.

1 Introduction
Natural sciences, and especially biology, represented a rich source of modelling paradigms. Well-defined areas of artificial intelligence (genetic algorithms, neural networks), mathematics, and theoretical computer science (L systems, DNA computing) are massively influenced by the behaviour of various biological entities and phenomena. In the last decades or so, new emerging fields of so-called natural computing identify new (unconventional) computational paradigms in different forms. There are attempts to define and investigate new mathematical or theoretical models inspired by nature, as well as investigations into defining programming paradigms that implement computational approaches suggested by biochemical phenomena. Especially since Adleman’s experiment, these investigations received a new perspective. One hopes that global system-level behaviour may be translated into interactions of a myriad of components with simple behaviour and limited computing and communication capabilities that are able to express and solve, via various optimizations, complex problems otherwise hard to approach.

A number of computational paradigms, inspired or gleaned from biochemical phenomena, are becoming of growing interest building a wealth of models, called generically Molecular Computing. New advances in, on the one hand, molecular and theoretical biology, and on the other hand, mathematical and computational sciences promise to make it possible in the near future to have accurate systemic models of complex biological phenomena.

2 Social Intelligence

This section shows some new computational paradigms that are based on the co-operation of individuals instead on the competition of individuals (typically modelled by genetic algorithms). Such social intelligence makes individuals to evolve towards the best solution using information from other individuals but none of them disappear. This is a new approach taken from the biology, in essence, social behaviour helps individuals to adapt to their environment, as it ensures that they obtain access to more information than that captured by their own senses.

Two popular variants of swarm models exist, those inspired by studies of social insects such as ant colonies, and those inspired by studies of the flocking behaviour of birds and fish.

2.1 Ant Colony Optimization

Ant colony optimization (ACO) is a class of optimization algorithms modelled on the actions of an ant
colony. ACO methods are useful in problems that need to find paths to goals. Artificial 'ants' - simulation agents - locate optimal solutions by moving through a parameter space representing all possible solutions. Real ants lay down pheromones directing each other to resources while exploring their environment. The simulated 'ants' similarly record their positions and the quality of their solutions, so that in later simulation iterations more ants locate better solutions.[2] One variation on this approach is the bees algorithm, which is more analogous to the foraging patterns of the honey bee.

2.2 Particle Swarm Optimization

Particle swarm optimization (PSO) is a global optimization algorithm for dealing with problems in which a best solution can be represented as a point or surface in an n-dimensional space. Hypotheses are plotted in this space and seeded with an initial velocity, as well as a communication channel between the particles [5, 6]. Particles then move through the solution space, and are evaluated according to some fitness criterion after each timestep. Over time, particles are accelerated towards those particles within their communication grouping which have better fitness values. The main advantage of such an approach over other global minimization strategies such as simulated annealing is that the large number of members that make up the particle swarm make the technique impressively resilient to the problem of local minima.

3 Grammatical Swarm

Grammatical Swarm (GS) adopts a Particle Swarm learning algorithm coupled to a Grammatical Evolution (GE) genotype-phenotype mapping to generate programs in an arbitrary language. Grammatical Evolution (GE) is an evolutionary algorithm that can evolve computer programs in any language [1, 2], and can be considered a form of grammar-based genetic programming. Rather than representing the programs as parse trees, as in GP [3, 4], a linear genome representation is used. A genotype-phenotype mapping is employed such that each individual's variable length binary string, contains in its codons (groups of 8 bits) the information to select production rules from a Backus Naur Form (BNF) grammar. The grammar allows the generation of programs in an arbitrary language that are guaranteed to be syntactically correct, and as such it is used as a generative grammar, as opposed to the classical use of grammars in compilers to check syntactic correctness of sentences. The user can tailor the grammar to produce solutions that are purely syntactically constrained, or they may incorporate domain knowledge by biasing the grammar to produce very specific forms of sentences. BNF is a notation that represents a language in the form of production rules. Let us suppose the following BNF grammar:

```
<expr> ::= - <expr><op><expr>
    | <var>
<op> ::= +
    | -
    | *
<var> ::= - x
    | y
```

And the following genotype:

```
14 8 27 254 5 17
```

In the example individual (see figure 3), the left-most <expr> in <expr> <op> <expr> is mapped by reading the next codon integer value 240 and used in 240%2 = 0 to become another <expr> <op> <expr>. The developing program now looks like <expr> <op> <expr> <op> <expr>. Continuing to read subsequent codons and always mapping the left-most non-terminal the individual finally generates the expression y * x - x - x + x, leaving a number of unused codons at the end of the individual, which are deemed to be introns and simply ignored.

This is the classic benchmark problem in which evolution attempts to find the five input even-parity boolean function [7]. The grammar adopted here is:

```
<prog> ::= <expr>
<expr> ::= <expr> <op> <expr>
    | ( <expr> <op> <expr> )
    | <var> | <pre-op> ( <var> )
<pre-op> ::= not
<op> ::= "|" | 
<var> ::= d0 | d1 | d2 | d3 | d4
```

The result is given by the best individual, see transcript bellow. Figure 3 shows a graphic with the best, average and variance of the swarm population. This figure has been obtained using the GEVA simulator [7].

```
{ not { d1 } | d2 ~ d4 } &
not ( d3 ) " { not ( d1 ) } &
{ not { d2 } } &
{ not { d2 } } |
{ d1 ~ not ( d3 ) ~ not ( d1 ) ~
( not { d1 } ~ ( d0 | not ( d4 ) ) ) ) } ~ d4 )
~ not ( d0 ) ~ d1
```
Figure 1: Grammatical Swarm Concepts.

Figure 2: Results of even-5-parity problem simulated with GEVA.
4 Neural Networks and Grammatical Swarm

Neural networks are non-linear systems whose structure is based on principles observed in biological neuronal systems. A neural network could be seen as a system that can be able to answer a query or give an output as answer to a specific input. The in/out combination, i.e. the transfer function of the network is not programmed, but obtained through a "training" process on empiric datasets. In practice the network learns the function that links input together with output by processing correct input/output couples. Actually, for each given input, within the learning process, the network gives a certain output which is not exactly the desired output, so the training algorithm modifies some parameters of the network in the desired direction. Hence, every time an example is input, the algorithm adjusts its network parameters to the optimal values for the given solution: in this way the algorithm tries to reach the best solution for all the examples. These parameters we are speaking about are essentially the weights or linking factors between each neuron that forms our network.

Neural Networks application fields are typically those where classic algorithms fail because of their unflexibility (they need precise input datasets). Usually problems with unprecise input datasets are those whose number of possible input datasets is so big that they cant be classified. For example in image recognition are used probabilistic algorithms whose efficiency is lower than neural networks and whose characteristics are low flexibility and high development complexity. Another field where classic algorithms are in troubles is the analysis of those phenomena whose mathematical rules are unknown. There are indeed rather complex algorithms which can analyze these phenomena but, from comparisons on the results, it comes out that neural networks result far more efficient [9, 8]: these algorithms use Fourier's trasmorm to decompose phenomena in frequent components and for this reason they result highly complex and they can only extract a limited number of harmonics generating a big number of approximations. A neural network trained with complex phenomena data is able to estimate also frequent components, this means that it realizes in its inside a Fourier's transformer even if it was not trained for that! One of the most important neural networks applications is undoubtedly the estimation of complex phenomena such as meteorological, financial, socio-economic or urban events.

Thanks to a neural network its possible to predict, analyzing hystorical series of datasets just as with these systems but there is no need to restrict the problem or use Fourier's trasmorm. A defect common to all those methods its to restrict the problem setting certain hypothesis that can turn out to be wrong. We just have to train the neural network with hystorical series of data given by the phenomenon we are studying. Calibrating a neural network means to determinate the parameters of the connections (synapsis) through the training process. Once calibrated there is need to test the network efficiency with known datasets, which has not been used in the learning process. There is a great number of Neural Networks which are substantially distinguished by: type of use, learning model (supervised/non-supervised), learning algorithm, architecture, etc.

But the most common trouble consists on what architecture must be used in order to get better results. That is the reason this paper proposes a Grammatical Swarm algorithm to get a right architecture/topology. Moreover, the training process can be done using Particle Swarm Optimization. With these two models the whole neural network is obtained (topology and weights) using ideas from social intelligence. Next sections describe how to implement a model in order to get a neural network topology and how to train this topology.

4.1 Training using Particle Swarm Optimization

Given a fixed neural network architecture, all weights in connections can be coded as a genotype and apply the particle swarm optimization algorithm in order to train the network where the fitness function must be the mean squared error of the net with the training set. Some variations can be done just using validation and testing sets to get better fitness values with more generalization properties.

Equations used in the particle swarm optimization training process are the following ones, where $c_1$ and $c_2$ are two positive constants, $R_1$ and $R_2$ are two random numbers belonging to [0, 1] and $w$ is the inertia weight. This equations define how the genotype values are changing along iterations, in our case, how neural network weights are changing.

\[
\begin{align*}
v_{in}(t + 1) &= wv_{in}(t) + c_1 R_1 (p_{in} - x_{in}(t)) + c_2 R_2 (p_{gn} - x_{in}(t)) \\
x_{in}(t + 1) &= x_{in}(t) + v_{in}(t + 1)
\end{align*}
\] (1)

Previous equations will modified the network weights till a stop conditions is achieved, that is, a lower mean squared error or a maximum number of iterations is reached.
Here it is the algorithm to train the network with the particle swarm optimization model:

Initialize population with random values $[-1,1]$

WHILE not finish condition is satisfied

FOR $i=1$ TO $n$ DO

Compute fitness particle $J_i$ (MSE of the net)
IF $J_i < p_{id}$
$p_{id} = J_i$
END IF
IF $J_i < p_{ig}$
$p_{ig} = J_i$
END IF
Compute new velocity of $i$ (equation 2)
Compute new position of $i$ (equation 1)
END FOR

END WHILE

This particle swarm optimization model is similar to genetic algorithms model but it uses a collaborative approach instead a competitive one.

Figure 4 shows different neural network architectures and their learning evolution according to the number of PSO iteration (maximum 30 iterations). We can see that any multilayer perceptron, with at least 2 hidden neurons, can successfully solve the XOR problem. While a multilayer perceptron with only one hidden neuron will never achieve a mean squared error lower than 2.0.

4.2 Grammatical Topology

Previous PSO model applied to a fixed neural network is a good solution to train a net, but it does not define any kind or topology properties it only obtains the best weights configuration. Next grammars can be used with Grammatical Swarm algorithms in order to obtain a network topology for a given problem.

This grammar can specify a feed-forward neural network topology just with consecutive layers, that is, a classical Multilayer Perceptron, see figure 3–a).

Next grammar is able to generate feed-forward connections not only with one consecutive layer but also with more than one consecutive layers, see figure 3–b). Such connections are defined by the <connections> non terminal, where the <digit> means the $n$-consecutive layer.

The whole algorithm, could be summarize as follows:

1. Create an initial population of genotypes.
2. For genotype $i$

   (a) Using genotype and grammar to obtain a neural architecture.

   (b) Compute Fitness of genotype.

   • Apply previous PSO algorithm to train the genotype network.

   (c) Modified the best individual if appropriate.

3. Update velocity of genotype $i$.
4. Update position of genotype $i$.
5. If stop condition is not satisfied go to step 2.

This neural network model is a powerful one since only with the input and output pattern sets a network topology is chosen and also trained. Both tools, topology and training, are based on grammatical swarm and particle swarm optimization respectively.

Figure 4 shows different multilayer perceptrons obtained with previous grammars. Depending on the topology that each individual codes the mean squared error will be lower or greater.
5 Conclusions

This paper has reviewed some natural computation strategies as a survey concerning optimization strategies. Some competitive and collaborative models have been exposed in order to understand the ability to extract some biological concepts and apply them in computational models as described along the paper. Such bio-inspired models have proved to be a powerful tool in order to solve non common problems in a collaborative/competitive way.

As a powerful application, neural networks can take advantage of such swarm optimization models. This paper has proposed a grammatical definition in order to choose the better network topology using grammatical swarm and the training of such networks (to compute the fitness function) is done with the PSO approach.

Acknowledgements: The research was supported by the Spanish Research Agency projects CCG08-UAM TIC-4425-2009 and TEC2007-68065-C03-02.

References:


